Patterns

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Patterns

Table of Contents

AWS Prescriptive Guidance patterns ... 1
Cloud foundations ... 3
Automate account creation using the Landing Zone Accelerator ... 4
Summary ... 4
Prerequisites and limitations ... 4
Architecture ... 5
Tools ... 6
Best practices ... 9
Epics ... 10
Related resources ... 36
Additional information ... 36
Automatically inventory AWS resources ... 39
Summary ... 39
Prerequisites and limitations ... 39
Architecture ... 41
Tools ... 42
Best practices ... 43
Epics ... 43
Troubleshooting ... 50
Related resources ... 51
Configure VPC Flow Logs across accounts ... 52
Summary ... 52
Prerequisites and limitations ... 52
Architecture ... 53
Tools ... 54
Best practices ... 55
Epics ... 57
Related resources ... 58
Additional information ... 58
Manage AWS permission sets using Terraform ... 61
Summary ... 61
Prerequisites and limitations ... 61
Architecture ... 62
Tools ... 65

Best practices ... 66
Epics ... 67
Troubleshooting ... 78
Related resources ... 79
Additional information ... 79
Tag Transit Gateway attachments automatically ... 81
Summary ... 81
Prerequisites and limitations ... 81
Architecture ... 81
Tools ... 84
Epics ... 85
Related resources ... 92
More patterns ... 93
AI & machine learning ... 94
Aggregate DynamoDB data for ML forecasting in Athena ... 96
Summary ... 96
Prerequisites and limitations ... 96
Architecture ... 96
Tools ... 98
Epics ... 99
Related resources ... 109
Associate an AWS CodeCommit repository with Amazon SageMaker AI Studio Classic across accounts ... 110
Summary ... 110
Prerequisites and limitations ... 110
Architecture ... 110
Tools ... 111
Epics ... 112
Additional information ... 117
Automatically extract content from PDF files ... 120
Summary ... 120
Prerequisites and limitations ... 120
Architecture ... 121
Tools ... 122
Epics ... 123
Related resources ... 127

- Attachments 128
- Build a cold start forecasting model using SageMaker AI DeepAR 129
  - Summary 129
  - Prerequisites and limitations 129
  - Architecture 130
  - Tools 132
  - Best practices 132
  - Epics 133
  - Related resources 135
- Build an MLOps workflow using SageMaker AI and Azure DevOps 136
  - Summary 136
  - Prerequisites and limitations 136
  - Architecture 137
  - Tools 140
  - Best practices 141
  - Epics 141
  - Troubleshooting 150
  - Related resources 151
- Configure logging in Amazon Bedrock using CloudFormation 153
  - Summary 153
  - Prerequisites and limitations 153
  - Architecture 154
  - Tools 156
  - Epics 157
  - Related resources 159
- Create Docker containers in SageMaker for model training in Step Functions 161
  - Summary 161
  - Prerequisites and limitations 161
  - Architecture 162
  - Tools 162
  - Epics 163
  - Related resources 175
- Create access controls in Amazon EKS with Amazon Bedrock agents 176
  - Summary 176
  - Prerequisites and limitations 176
  - Architecture 177

- Tools 179
- Best practices 180
- Epics 180
- Troubleshooting 197
- Related resources 198

Deploy a RAG use case on AWS 199
- Summary 199
- Prerequisites and limitations 199
- Architecture 200
- Tools 202
- Best practices 204
- Epics 204
- Related resources 208
- Additional information 208

Deploy multiple pipeline model objects in a single SageMaker endpoint 210
- Summary 210
- Prerequisites and limitations 210
- Architecture 211
- Tools 212
- Epics 213
- Related resources 223

Develop AI chat-based assistants by using RAG and ReAct prompting 224
- Summary 224
- Prerequisites and limitations 225
- Architecture 226
- Tools 230
- Best practices 232
- Epics 232
- Troubleshooting 238
- Related resources 238
- Additional information 239

Develop a chat-based assistant using Amazon Bedrock 240
- Summary 240
- Prerequisites and limitations 240
- Architecture 241
- Tools 244

- Best practices ... 245
- Epics ... 246
- Related resources ... 249
- Additional information ... 250
- Document institutional knowledge from voice inputs ... 253
  - Summary ... 253
  - Prerequisites and limitations ... 253
  - Architecture ... 254
  - Tools ... 256
  - Best practices ... 257
  - Epics ... 258
  - Related resources ... 263
- Generate personalized recommendations using Amazon Personalize ... 265
  - Summary ... 265
  - Prerequisites and limitations ... 265
  - Architecture ... 265
  - Tools ... 267
  - Epics ... 268
  - Related resources ... 271
  - Additional information ... 272
- Streamline ML workflows using SageMaker AI and Hydra ... 276
  - Summary ... 276
  - Prerequisites and limitations ... 276
  - Architecture ... 277
  - Tools ... 279
  - Best practices ... 280
  - Epics ... 280
  - Troubleshooting ... 287
  - Related resources ... 287
  - Additional information ... 287
- Train and deploy a custom GPU-supported ML model ... 289
  - Summary ... 289
  - Prerequisites and limitations ... 289
  - Architecture ... 290
  - Tools ... 290
  - Epics ... 291

Related resources ... 306
Additional information ... 306
... 309
Summary ... 309
Prerequisites and limitations ... 310
Architecture ... 313
Tools ... 315
Best practices ... 316
Epics ... 316
Related resources ... 323
Additional information ... 323
Use Amazon Q Developer as a coding assistant ... 328
Summary ... 328
Prerequisites and limitations ... 328
Tools ... 329
Best practices ... 329
Epics ... 330
Troubleshooting ... 338
Related resources ... 339
Use SageMaker Processing for distributed feature engineering of terabyte-scale ML datasets ... 340
Summary ... 340
Prerequisites and limitations ... 340
Architecture ... 341
Tools ... 345
Epics ... 345
Related resources ... 356
Attachments ... 357
Visualize AI/ML model results using Flask and Elastic Beanstalk ... 358
Summary ... 358
Prerequisites and limitations ... 358
Architecture ... 359
Tools ... 362
Epics ... 362
Related resources ... 371
Additional information ... 371

More patterns 376
Analytics 377
Analyze Amazon Redshift data in Microsoft SQL Server Analysis Services 379
Summary 379
Prerequisites and limitations 379
Architecture 379
Tools 380
Epics 380
Related resources 382
.... 383
Summary 383
Prerequisites and limitations 383
Architecture 383
Tools 384
Epics 385
Related resources 390
Automate data ingestion from AWS Data Exchange into Amazon S3 391
Summary 391
Prerequisites and limitations 391
Architecture 391
Tools 392
Epics 393
Related resources 395
Attachments 395
Automate encryption enforcement in AWS Glue 396
Summary 396
Prerequisites and limitations 396
Architecture 396
Tools 398
Best practices 398
Epics 399
Related resources 401
Build a data pipeline to process Google Analytics data using the AWS DataOps Development Kit 402
Summary 402
Prerequisites and limitations 402

- Architecture ... 402
- Tools ... 404
- Epics ... 405
- Troubleshooting ... 407
- Related resources ... 407
- Additional information ... 407

Build a video processing pipeline ... 410
- Summary ... 410
- Prerequisites and limitations ... 410
- Architecture ... 411
- Tools ... 412
- Epics ... 412
- Related resources ... 420
- Additional information ... 420
- Attachments ... 421

Build an ETL pipeline from Amazon S3 to Amazon Redshift using AWS Glue ... 422
- Summary ... 422
- Prerequisites and limitations ... 422
- Architecture ... 423
- Tools ... 424
- Epics ... 425
- Related resources ... 431
- Additional information ... 432

Build an enterprise data mesh with Amazon DataZone ... 433
- Summary ... 433
- Prerequisites and limitations ... 433
- Architecture ... 434
- Tools ... 436
- Epics ... 437
- Related resources ... 449
- Additional information ... 450

Calculate value at risk (VaR) by using AWS services ... 452
- Summary ... 452
- Prerequisites and limitations ... 452
- Architecture ... 453
- Tools ... 455

Best practices 456
Epics 456
Related resources 459
Configure cross-account access to a shared AWS Glue Data Catalog 460
Summary 460
Prerequisites and limitations 460
Architecture 460
Tools 462
Epics 462
Related resources 475
Additional information 475
Convert NORMALIZE to Amazon Redshift SQL 476
Summary 476
Prerequisites and limitations 476
Architecture 476
Tools 477
Epics 482
Related resources 482
Convert RESET WHEN to Amazon Redshift SQL 483
Summary 483
Prerequisites and limitations 483
Architecture 483
Tools 484
Epics 488
Related resources 488
Deploy and manage a serverless data lake on AWS 490
Summary 490
Prerequisites and limitations 490
Architecture 491
Tools 493
Epics 494
Related resources 496
..... 498
Summary 498
Prerequisites and limitations 498
Architecture 499

Tools ... 499
Epics ... 500
Related resources ... 503
Attachments ... 503
Ensure Amazon EMR logging to Amazon S3 ... 504
Summary ... 504
Prerequisites and limitations ... 504
Architecture ... 505
Tools ... 506
Epics ... 506
Related resources ... 509
Attachments ... 509
Generate test data using AWS Glue ... 510
Summary ... 510
Prerequisites and limitations ... 510
Architecture ... 511
Tools ... 511
Best practices ... 512
Epics ... 512
Related resources ... 523
Additional information ... 524
Ingest IoT data directly into Amazon S3 ... 528
Summary ... 528
Prerequisites and limitations ... 528
Architecture ... 529
Tools ... 530
Best practices ... 530
Epics ... 531
Troubleshooting ... 538
Related resources ... 539
Additional information ... 539
Launch a Spark job in Amazon EMR using a Lambda function ... 544
Summary ... 544
Prerequisites and limitations ... 544
Architecture ... 545
Tools ... 545

Epics ... 546
Related resources ... 549
Additional information ... 550
Attachments ... 552
Migrate Apache Cassandra workloads to Amazon Keyspaces ... 553
Summary ... 553
Prerequisites and limitations ... 553
Architecture ... 553
Tools ... 554
Best practices ... 555
Epics ... 555
Troubleshooting ... 568
Related resources ... 568
Additional information ... 568
Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator ... 570
Summary ... 570
Prerequisites and limitations ... 570
Architecture ... 571
Epics ... 572
Related resources ... 577
Additional information ... 577
Migrate Oracle Business Intelligence 12C to the AWS Cloud ... 579
Summary ... 579
Prerequisites and limitations ... 579
Architecture ... 580
Tools ... 582
Epics ... 583
Related resources ... 596
Additional information ... 596
Migrate an ELK Stack to the AWS Cloud ... 601
Summary ... 601
Prerequisites and limitations ... 601
Architecture ... 603
Tools ... 606
Epics ... 607
Related resources ... 614

Additional information ... 616
Migrate data to AWS using Starburst ... 617
Summary ... 617
Prerequisites and limitations ... 617
Architecture ... 617
Tools ... 621
Epics ... 622
Related resources ... 625
Optimize the ETL ingestion of input file size ... 626
Summary ... 626
Prerequisites and limitations ... 626
Architecture ... 626
Tools ... 628
Epics ... 628
Related resources ... 631
Additional information ... 631
Orchestrate an ETL pipeline with Step Functions ... 632
Summary ... 632
Prerequisites and limitations ... 632
Architecture ... 633
Tools ... 634
Epics ... 635
Troubleshooting ... 642
Related resources ... 643
Additional information ... 643
Perform ML analytics using Amazon Redshift ML ... 646
Summary ... 646
Prerequisites and limitations ... 646
Architecture ... 647
Tools ... 648
Epics ... 649
Related resources ... 653
Query Amazon DynamoDB by using Amazon Athena ... 654
Summary ... 654
Prerequisites and limitations ... 654
Architecture ... 655

- Tools ... 656
- Epics ... 657
- Troubleshooting ... 660
- Related resources ... 661

Query DynamoDB tables using Athena ... 662
- Summary ... 662
- Prerequisites and limitations ... 662
- Architecture ... 663
- Tools ... 664
- Epics ... 664
- Related resources ... 673
- Additional information ... 673

Set up a minimum viable data space ... 675
- Summary ... 675
- Prerequisites and limitations ... 676
- Architecture ... 677
- Tools ... 679
- Best practices ... 680
- Epics ... 681
- Troubleshooting ... 735
- Related resources ... 735
- Additional information ... 735

Set up language-specific sorting for Amazon Redshift query results ... 740
- Summary ... 740
- Prerequisites and limitations ... 740
- Architecture ... 740
- Tools ... 741
- Epics ... 741
- Related resources ... 745
- Additional information ... 746

Subscribe a Lambda function to event notifications from cross-Region S3 buckets ... 750
- Summary ... 750
- Prerequisites and limitations ... 750
- Architecture ... 750
- Tools ... 752
- Epics ... 752

Related resources ... 756
Three AWS Glue job types for converting data ... 758
Summary ... 758
Prerequisites and limitations ... 758
Architecture ... 758
Tools ... 759
Epics ... 760
Related resources ... 763
Additional information ... 763
Attachments ... 769
Visualize Amazon Redshift audit logs using Athena and QuickSight ... 770
Summary ... 770
Prerequisites and limitations ... 770
Architecture ... 771
Tools ... 771
Epics ... 772
Related resources ... 775
Attachments ... 775
Visualize IAM credential reports using Amazon QuickSight ... 776
Summary ... 776
Prerequisites and limitations ... 776
Architecture ... 777
Tools ... 778
Epics ... 779
Additional information ... 785
More patterns ... 787
Compute ... 789
Containers & microservices ... 790
Access Amazon Neptune from Amazon EKS ... 792
Access container applications on Amazon ECS ... 804
Access container applications on Amazon ECS with an AWS Fargate launch type ... 822
Access container applications privately on Amazon EKS ... 838
Activate mTLS in App Mesh on Amazon EKS ... 846
Automate backups for Amazon RDS for PostgreSQL DB instances ... 854
Automate deployment of Node Termination Handler ... 867
Automatically build and deploy a Java application to Amazon EKS ... 882

Create an Amazon ECS task definition on EC2 instances using Amazon EFS 906
Deploy Lambda functions with container images 912
Deploy Java microservices on Amazon ECS using AWS Fargate 921
Deploy Kubernetes packages using Amazon EKS and Helm 928
Deploy a Java microservice on Amazon EKS and expose it with an Application Load Balancer 939
Deploy a clustered application to Amazon ECS by using AWS Copilot 952
Deploy a gRPC-based application on Amazon EKS 963
Deploy and debug Amazon EKS clusters 977
Deploy containers by using Elastic Beanstalk 1006
Generate a static outbound IP address using Lambda and Amazon VPC 1013
Identify duplicate container images automatically 1028
Install SSM Agent on Amazon EKS worker nodes 1058
Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands 1064
Migrate NGINX Ingress Controllers for EKS Auto Mode 1071
Migrate container workloads from ARO to ROSA 1091
Place Kubernetes Pods on compatible nodes in Amazon EKS 1109
Replicate filtered Amazon ECR container images across accounts or Regions 1127
Rotate credentials without restarting containers 1144
Run Amazon ECS tasks on Amazon WorkSpaces 1151
Run an ASP.NET web API Docker container on AWS 1162
Run message-driven workloads using AWS Fargate 1175
Run stateful workloads with persistent data storage 1185
Set up auto scaling using Amazon EKS Pod Identity and KEDA 1209
.... 1233
Use mutual TLS in Amazon ECS with ALB 1255
More patterns 1285
Serverless 1287
Build a React Native app using AWS Amplify 1288
Centrally manage tenants across multiple SaaS products 1307
Consolidate Amazon S3 presigned URL generation and object downloads 1320
Create a cross-account Amazon EventBridge connection 1337
Deliver DynamoDB records to Amazon S3 1352
Implement path-based API versioning by using API Gateway 1359
Import the psycopg2 library to AWS Lambda 1371

Integrate API Gateway with Amazon SQS 1380
Process APIs asynchronously with AWS Lambda 1396
Process APIs asynchronously with Amazon DynamoDB Streams 1406
Process APIs asynchronously with Amazon SQS 1416
Run Systems Manager Automation tasks synchronously from Step Functions 1427
Run parallel reads of S3 objects with AWS Lambda 1442
Send telemetry data from Lambda to OpenSearch 1457
Set up a serverless cell router 1474
Set up private access to an Amazon S3 bucket 1493
Troubleshoot Step Functions states 1500
Use a serverless approach to chain AWS services together 1510
More patterns 1516
Networking 1518
Automate peering for AWS Transit Gateway 1519
Centralize network connectivity using AWS Transit Gateway 1526
Configure HTTPS encryption for Oracle JD Edwards EnterpriseOne by using an Application Load Balancer 1533
Connect to Application Migration Service data and control planes over a private network 1544
Create Infoblox objects using AWS CloudFormation custom resources 1559
Customize CloudWatch alerts for Network Firewall 1573
Deploy resources in a Wavelength Zone using Terraform 1593
Migrate DNS records in bulk to a Route 53 private hosted zone 1603
Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS 1613
Report Network Access Analyzer findings in multiple AWS accounts 1619
Set up DNS resolution for hybrid networks in a multi-account AWS environment 1651
..... 1669
View AWS Network Firewall logs and metrics using Splunk 1675
More patterns 1686
Content delivery 1687
Send AWS WAF logs to Splunk using Firehose 1688
Serve static content in an S3 bucket through a VPC by using CloudFront 1697
More patterns 1707
Databases & storage 1708
Databases 1709

Access on-premises SQL Server data using linked servers ... 1711
Add HA to Oracle PeopleSoft on AWS ... 1717
Assess query performance for migrating SQL Server databases to MongoDB Atlas on AWS ... 1741
Automate blue/green deployments of Aurora global databases with IaC ... 1752
Automate cross-account replication of Amazon RDS instances ... 1786
Automate database tasks with AWS Lambda and Task Scheduler ... 1801
Automate failover and failback with DR Orchestrator Framework ... 1813
Automatically back up SAP HANA databases ... 1842
Automatically generate PynamoDB models and CRUD functions for DynamoDB ... 1850
Block public access to Amazon RDS ... 1859
Configure cross-account Amazon DynamoDB access ... 1867
Configure read-only routing in an Always On availability group ... 1882
Connect by using an SSH tunnel in pgAdmin ... 1892
Convert JSON Oracle queries into PostgreSQL database SQL ... 1897
Copy Amazon DynamoDB tables across accounts ... 1928
Copy Amazon DynamoDB tables across accounts ... 1935
Create cost and usage reports for Amazon RDS and Amazon Aurora ... 1948
Deploy SQL Server FCIs on Amazon EC2 and Amazon FSx ... 1956
Emulate Oracle PL/SQL associative arrays in Amazon Aurora and Amazon RDS ... 1973
Enable encrypted connections for PostgreSQL DB instances ... 1984
Encrypt an existing Amazon RDS for PostgreSQL DB instance ... 1993
Enforce automatic tagging of Amazon RDS databases at launch ... 2001
Estimate DynamoDB costs ... 2007
Estimate storage costs for an Amazon DynamoDB table ... 2019
Estimate the Amazon RDS engine size for an Oracle database using AWR reports ... 2024
Export Amazon RDS for SQL Server tables to an S3 bucket ... 2054
Handle anonymous blocks in Dynamic SQL statements ... 2066
Handle overloaded Oracle functions in Aurora PostgreSQL-Compatible ... 2074
Help enforce DynamoDB tagging ... 2081
Implement cross-Region DR ... 2087
Migrate 100+ argument Oracle functions to PostgreSQL ... 2104
Migrate SAP HANA to AWS using HSR ... 2110
Migrate SQL Server Always On availability group using AWS MGN ... 2124
Migrate SQL Server to AWS using distributed availability groups ... 2135
Migrate from Oracle 8i or 9i to Amazon RDS for Oracle using SharePlex and AWS DMS ... 2146

Monitor Amazon Aurora for encryption ... 2155
Monitor GoldenGate logs by using Amazon CloudWatch ... 2161
Replatform Oracle Database EE to Amazon RDS for Oracle SE2 ... 2174
Replicate mainframe databases to AWS by using Precisely Connect ... 2185
Schedule jobs for Amazon RDS and Aurora PostgreSQL ... 2207
Send notifications for RDS for SQL Server using an SMTP server ... 2213
Set up DR for SAP on IBM Db2 on AWS ... 2226
Set up a CI/CD pipeline for database migration by using Terraform ... 2248
Set up an HA/DR architecture for Oracle E-Business Suite on Amazon RDS Custom ... 2258
Set up data replication between RDS for MySQL and MySQL on Amazon EC2 ... 2267
Transition roles for an Oracle PeopleSoft application ... 2273
Unload data from Amazon Redshift across accounts ... 2307
Database migration patterns by workload ... 2337
More patterns ... 2348
Storage & backup ... 2353
Allow EC2 instances write access to S3 buckets in AMS ... 2354
Automate data stream ingestion into a Snowflake database ... 2360
Automatically encrypt EBS volumes ... 2371
Back up Sun SPARC servers in the Charon-SSP emulator on AWS ... 2383
Back up and archive data to Amazon S3 with Veeam ... 2405
Configure NetBackup for VMware Cloud on AWS ... 2431
Copy S3 objects between accounts and Regions by using AWS CLI ... 2439
Copy S3 objects between accounts and Regions by using S3 Batch Replication ... 2454
Migrate Hadoop data to Amazon S3 using DistCp and AWS PrivateLink for Amazon S3 .. 2467
More patterns ... 2482
Developer tools ... 2484
DevOps ... 2485
Automate AWS infrastructure operations ... 2488
Automate CloudFront updates for changes in load balancer endpoints ... 2498
Automate CodeGuru reviews for AWS CDK Python applications ... 2508
Automate AWS resource assessment ... 2520
Automate SAP systems installation ... 2535
Automate Service Catalog portfolio and product deployment using AWS CDK ... 2547
Automate deletion of CloudFormation stacks ... 2567
Automate dynamic pipeline management to deploy hotfix solutions ... 2580
Automate ingestion of Amazon MWAA custom metrics ... 2609

Automate management of Amazon Lex resources using the AWS CDK and CloudFormation 2626
Automate stack set deployment by using AWS CodePipeline and AWS CodeBuild 2646
Automatically attach a managed policy for Systems Manager to EC2 instance profiles 2677
Automatically build CI/CD pipelines and Amazon ECS clusters for microservices 2693
Build a loosely coupled architecture with microservices 2706
Build and push Docker images to Amazon ECR 2718
Build and test iOS apps with AWS services 2724
Check AWS CDK applications or CloudFormation templates for best practices by using rule packs 2731
Configure mutual TLS for applications on Amazon EKS 2736
Create AppStream 2.0 resources using AWS CloudFormation 2748
Create a custom log parser for Amazon ECS using Firelens 2754
Create a pipeline and AMI using CodePipeline and HashiCorp Packer 2764
Create dynamic CI pipelines for Java and Python projects 2771
Deploy CloudWatch Synthetics canaries 2787
Deploy a CI/CD pipeline for Java microservices on Amazon ECS 2794
Deploy a ChatOps solution to manage SAST scan results 2804
Deploy agentic systems on Amazon Bedrock with CrewAI 2823
.... 2841
Deploy an Amazon EKS cluster from AWS Cloud9 using an EC2 instance profile 2846
Deploy code in multiple AWS Regions 2858
Deploy updates to on-premises EC2 instances using CodePipeline 2870
Deploy workloads from Azure DevOps pipelines to private Amazon EKS clusters 2879
Execute Amazon Redshift SQL queries 2912
Export AWS Backup reports as a CSV file 2925
Export Amazon EC2 instance tags to a CSV file 2934
Generate an AWS CloudFormation template containing AWS Config managed rules 2941
Give SageMaker notebook instances cross-account access to a CodeCommit repository 2948
Implement a GitHub Flow branching strategy 2959
Implement a Gitflow branching strategy 2969
Implement a Trunk branching strategy 2983
Implement centralized custom Checkov scanning 2990
Implement diagnostics with K8sGPT and Amazon Bedrock 3001
Initiate different CI/CD pipelines after detecting changes in a monorepo 3016
Integrate a Bitbucket repository with AWS Amplify 3034

Launch a CodeBuild project across AWS accounts using Lambda 3043
Manage Multi-AZ failover for EMR clusters by using Application Recovery Controller 3058
Manage blue/green deployments of microservices to multiple accounts and Regions 3073
Monitor Amazon ECR repositories for wildcard permissions 3108
Optimize multi-account serverless deployments 3113
Provision AWS Service Catalog products using GitHub Actions 3133
Provision least-privilege IAM roles 3145
Publish Amazon CloudWatch metrics to a CSV file 3159
Remove Amazon EC2 entries across AWS accounts from AWS Managed Microsoft AD 3165
Remove Amazon EC2 entries in same AWS account from AWS Managed Microsoft AD 3183
Run unit tests for Python ETL jobs in AWS Glue 3194
Set up Helm v3 charts in Amazon S3 3204
Set up a CI/CD pipeline with CodePipeline 3213
Set up centralized logging at scale using Terraform 3228
Set up end-to-end encryption for applications on Amazon EKS 3253
Simplify Amazon EKS multi-tenant application deployment 3267
Subscribe multiple email endpoints to an SNS topic 3289
Use the AWS Fargate WaitCondition hook construct 3294
Use third-party Git repos in AWS CodePipeline 3306
Validate Terraform configurations by using AWS CodePipeline 3316
More patterns 3333
Infrastructure 3336
Access a bastion host using Session Manager and Instance Connect 3338
Centralize DNS resolution by using AWS Managed Microsoft AD 3355
Centralize monitoring by using Observability Access Manager 3367
Check EC2 instances for mandatory tags at launch 3382
Clean up AFT resources safely after state file loss 3388
Connect to an EC2 instance using Session Manager 3402
Create a pipeline in AWS Regions that don't support AWS CodePipeline 3411
Customize default role names by using AWS CDK aspects and escape hatches 3419
Deploy a Cassandra cluster on Amazon EC2 with private static IPs 3428
Extend VRFs to AWS using Transit Gateway Connect 3436
Get Amazon SNS notifications for state changes to AWS KMS keys 3454
Preserve routable IP space in multi-account VPC designs for non-workload subnets 3461
Provision a Terraform product in Service Catalog from a repo 3467
Register multiple AWS accounts with a single email address 3487

Set up DNS resolution for hybrid networks in a single-account AWS environment 3505
Set up UiPath RPA bots automatically on Amazon EC2 3511
Set up a highly available PeopleSoft architecture on AWS 3529
Set up disaster recovery for Oracle JD Edwards EnterpriseOne 3558
Set up drift detection and reporting 3585
Successfully import an S3 bucket as a CloudFormation stack 3591
Synchronize Amazon EFS file systems in different Regions 3602
Test AWS infrastructure using LocalStack and Terraform Tests 3610
Upgrade SAP Pacemaker clusters from ENSA1 to ENSA2 3619
Use consistent Availability Zones in VPCs across different accounts 3642
Use user IDs in IAM policies 3649
Validate Account Factory for Terraform code locally 3660
More patterns 3680
Web & mobile apps 3684
Authenticate React web application users 3685
Continuously deploy an Amplify web application 3696
Create a React app by using AWS Amplify and Amazon Cognito 3705
Create a portal for micro-frontends 3721
Deploy a React-based SPA to Amazon S3 and CloudFront 3753
Deploy an Amazon API Gateway API using private endpoints 3761
Embed an Amazon QuickSight dashboard in a local Angular application 3769
Explore web app development with Green Boost 3790
Run unit tests by using AWS CodeBuild 3818
Structure a Python project in hexagonal architecture 3827
More patterns 3853
IoT 3855
Extract and query AWS IoT SiteWise metadata attributes 3856
Summary 3856
Prerequisites and limitations 3856
Architecture 3857
Tools 3858
Epics 3858
Related resources 3861
Additional information 3861
Monitor for security events in an AWS IoT environment 3864
Summary 3864

Prerequisites and limitations 3864
Architecture 3865
Tools 3867
Epics 3868
Related resources 3872
..... 3873
Summary 3873
Prerequisites and limitations 3874
Architecture 3874
Tools 3876
Best practices 3876
Epics 3877
Troubleshooting 3892
Related resources 3894
Additional information 3894
More patterns 3899
Migration & modernization 3900
Migration 3901
Create AWS CloudFormation templates for AWS DMS 3902
Get started with automated portfolio discovery 3906
Migrate on-premises Cloudera workloads to AWS 3913
Resolve connection errors after migrating SQL Server to AWS 3930
Restart the AWS Replication Agent automatically without disabling SELinux 3933
Re-architect 3940
Rehost 4340
Relocate 4615
Replatform 4671
Migration patterns by workload 5209
More patterns 5218
Modernization 5220
Analyze and visualize software architecture in CAST Imaging 5221
Assess application readiness before migrating to AWS by using CAST Highlight 5233
Automatically archive expired DynamoDB data to Amazon S3 5257
Build a multi-tenant serverless architecture in Amazon OpenSearch Service 5274
Deploy multiple-stack applications 5327
Deploy nested applications using AWS SAM 5338

- Implement the serverless saga pattern by using AWS Step Functions ... 5348
- Manage on-premises container applications with Amazon ECS Anywhere ... 5364
- Modernize ASP.NET Web Forms applications on AWS ... 5375
- Run event-driven workloads with AWS Fargate ... 5391
- Tenant onboarding in SaaS architecture ... 5401
- Use CQRS and event sourcing ... 5426
- More patterns ... 5455

Mainframes ... 5457

- Access AWS services from IBM z/OS by installing the AWS CLI ... 5459
- Back up and archive mainframe data to Amazon S3 ... 5480
- Build COBOL Db2 programs by using AWS Mainframe Modernization and AWS CodeBuild ... 5506
- Build a Micro Focus Enterprise Server PAC ... 5530
- Build a mainframe file viewer in the AWS Cloud ... 5551
- Containerize modernized Blu Age applications ... 5568
- Convert EBCDIC data to ASCII on AWS ... 5578
- Convert mainframe EBCDIC files to ASCII files using AWS Lambda ... 5597
- Convert mainframe data files with complex record layouts ... 5618
- Deploy an environment for containerized apps ... 5635
- Generate Db2 z/OS data insights ... 5646
- Generate insights by using AWS Mainframe Modernization and Amazon Q in QuickSight 5701
- Implement Microsoft Entra ID authentication for AWS Blu Age ... 5723
- Integrate Stonebranch Universal Controller with AWS ... 5741
- Migrate and replicate VSAM files to the AWS Cloud using Precisely ... 5777
- Modernize a mainframe application using AWS Transform ... 5796
- Modernize mainframe output management on AWS ... 5815
- Modernize your mainframe batch printing workloads on AWS ... 5866
- Modernize your mainframe environment with Rocket Software Enterprise Suite ... 5897
- Modernize your mainframe online printing workloads on AWS ... 5918
- Move mainframe files to Amazon S3 using Transfer Family ... 5952
- Secure user access in a Db2 federation database ... 5966
- Transfer Db2 z/OS data to AWS ... 5975
- More patterns ... 6003

Management ... 6004

Cost management ... 6005

- Create detailed cost and usage reports for AWS Glue jobs ... 6006

Create detailed cost and usage reports for Amazon EMR clusters 6012
More patterns 6017
End-user computing 6018
Implement SAML 2.0 authentication for Amazon WorkSpaces 6019
More patterns 6029
High-performance computing 6030
Deploy a Lustre file system by using Terraform and DRA 6031
Set up a Grafana monitoring dashboard for AWS ParallelCluster 6040
More patterns 6055
Hybrid cloud 6056
Configure a data center extension to VMware Cloud on AWS 6057
Configure vRealize Automation to provision VMs on VMware Cloud on AWS 6063
Integrate VMware vRealize Network Insight with VMware Cloud on AWS 6074
Migrate VMs to VMware Cloud on AWS by using HCX OSAM 6080
Send logs from VMware Cloud on AWS to Splunk 6086
Set up a pipeline for hybrid workloads on Amazon ECS Anywhere 6093
More patterns 6112
Management & governance 6113
Alert when Amazon Data Firehose resources are not encrypted 6114
Automate adding or updating Windows registry entries 6119
Automatically create an RFC using Python 6124
Automatically stop and start an Amazon RDS DB instance 6131
Centralize software package distribution in AWS Organizations by using Terraform 6146
Configure logging for .NET applications in CloudWatch Logs 6159
Copy AWS Service Catalog products across AWS accounts and Regions 6168
Create a RACI matrix for cloud operations 6179
Create alarms for custom metrics using CloudWatch 6185
Create an AWS Cloud9 IDE with default encrypted EBS volumes 6192
Create tag-based CloudWatch dashboards automatically 6198
Document your landing zone design 6208
Enable Amazon DevOps Guru across an organization with the AWS CDK 6211
Govern permission sets for multiple accounts using AFT 6238
Implement AFT by using a bootstrap pipeline 6250
Manage AWS Service Catalog products in multiple AWS accounts and Regions 6275
Migrate an AWS account from AWS Organizations to AWS Control Tower 6283
Monitor SAP RHEL Pacemaker clusters 6298

Monitor application activity using CloudWatch Logs Insights ... 6317
Monitor use of an AMI across AWS accounts ... 6327
Set up alerts for programmatic account closures in AWS Organizations ... 6345
View EBS snapshot details for your AWS account or organization ... 6355
More patterns ... 6362
Messaging & communications ... 6364
Automate RabbitMQ configuration in Amazon MQ ... 6365
Improve call quality on agent workstations in Amazon Connect ... 6374
More patterns ... 6389
Security, identity & compliance ... 6390
Access AWS services from ASP.NET using Amazon Cognito ... 6393
Summary ... 6393
Prerequisites and limitations ... 6393
Architecture ... 6394
Tools ... 6394
Epics ... 6395
Troubleshooting ... 6399
Related resources ... 6400
Attachments ... 6400
Authenticate SQL Server using AWS Directory Service ... 6401
Summary ... 6401
Prerequisites and limitations ... 6401
Architecture ... 6401
Tools ... 6402
Epics ... 6402
Related resources ... 6406
Automate incident response and forensics ... 6407
Summary ... 6407
Prerequisites and limitations ... 6407
Architecture ... 6408
Tools ... 6414
Epics ... 6415
Related resources ... 6419
Additional information ... 6419
Attachments ... 6419
Automate remediation for Security Hub standard findings ... 6420

Summary ... 6420
Prerequisites and limitations ... 6420
Architecture ... 6421
Tools ... 6422
Best practices ... 6422
Epics ... 6423
Related resources ... 6426
Attachments ... 6426
Automate security scans for cross-account workloads using Amazon Inspector ... 6427
Summary ... 6427
Prerequisites and limitations ... 6427
Architecture ... 6428
Tools ... 6430
Epics ... 6430
Related resources ... 6434
Attachments ... 6434
Automatically audit access from public IP addresses ... 6435
Summary ... 6435
Prerequisites and limitations ... 6435
Architecture ... 6436
Tools ... 6439
Best practices ... 6440
Epics ... 6440
Troubleshooting ... 6444
Related resources ... 6445
Automatically re-enable AWS CloudTrail using security best practices ... 6446
Summary ... 6446
Prerequisites and limitations ... 6446
Architecture ... 6447
Tools ... 6448
Epics ... 6448
Related resources ... 6454
Attachments ... 6455
Automatically remediate unencrypted Amazon RDS DB instances and clusters ... 6456
Summary ... 6456
Prerequisites and limitations ... 6456

Architecture ... 6458
Tools ... 6458
Best practices ... 6460
Epics ... 6460
Related resources ... 6466
Additional information ... 6467
Automatically rotate IAM user access keys ... 6468
Summary ... 6468
Prerequisites and limitations ... 6469
Architecture ... 6470
Tools ... 6472
Epics ... 6474
Related resources ... 6484
Automatically validate and deploy IAM policies and roles ... 6486
Summary ... 6486
Prerequisites and limitations ... 6487
Architecture ... 6488
Tools ... 6489
Epics ... 6489
Related resources ... 6493
Bidirectionally integrate Security Hub and Jira ... 6495
Summary ... 6495
Prerequisites and limitations ... 6496
Architecture ... 6497
Tools ... 6499
Epics ... 6500
Related resources ... 6510
Additional information ... 6510
Build a pipeline for hardened container images ... 6512
Summary ... 6512
Prerequisites and limitations ... 6512
Architecture ... 6513
Tools ... 6517
Epics ... 6518
Troubleshooting ... 6526
Related resources ... 6527

Centralize IAM access key management in AWS Organizations by using Terraform ... 6528
Summary ... 6528
Prerequisites and limitations ... 6529
Architecture ... 6529
Tools ... 6531
Best practices ... 6532
Epics ... 6532
Troubleshooting ... 6541
Related resources ... 6542
Check an Amazon CloudFront distribution for access logging, HTTPS, and TLS version ... 6543
Summary ... 6543
Prerequisites and limitations ... 6543
Architecture ... 6544
Tools ... 6545
Epics ... 6545
Related resources ... 6548
Attachments ... 6548
Check for single-host network entries in security group ingress rules for IPv4 and IPv6 ... 6549
Summary ... 6549
Prerequisites and limitations ... 6549
Architecture ... 6549
Tools ... 6550
Epics ... 6551
Related resources ... 6554
Attachments ... 6554
Choose an Amazon Cognito authentication flow ... 6555
Summary ... 6555
Prerequisites and limitations ... 6555
Architecture ... 6556
Tools ... 6562
Epics ... 6562
Related resources ... 6565
Additional information ... 6566
Create AWS Config custom rules using Guard ... 6567
Summary ... 6567
Prerequisites and limitations ... 6568

- Architecture ... 6568
- Tools ... 6573
- Epics ... 6573
- Troubleshooting ... 6575
- Related resources ... 6576

Create a report of Prowler findings from multiple AWS accounts ... 6577
- Summary ... 6577
- Prerequisites and limitations ... 6578
- Architecture ... 6579
- Tools ... 6580
- Epics ... 6582
- Troubleshooting ... 6605
- Related resources ... 6605
- Additional information ... 6606

Delete unused Amazon EBS volumes using AWS Config ... 6609
- Summary ... 6609
- Prerequisites and limitations ... 6609
- Architecture ... 6610
- Tools ... 6612
- Epics ... 6612
- Troubleshooting ... 6615
- Related resources ... 6616

Deploy AWS Control Tower controls using AWS CDK ... 6617
- Summary ... 6617
- Prerequisites and limitations ... 6618
- Architecture ... 6619
- Tools ... 6620
- Best practices ... 6621
- Epics ... 6622
- Related resources ... 6629
- Additional information ... 6630

Deploy AWS Control Tower controls using Terraform ... 6633
- Summary ... 6633
- Prerequisites and limitations ... 6633
- Architecture ... 6635
- Tools ... 6636

Best practices ... 6637
Epics ... 6637
Troubleshooting ... 6642
Related resources ... 6644
Additional information ... 6644
Deploy Security Automations for AWS WAF using Terraform ... 6647
Summary ... 6647
Prerequisites and limitations ... 6647
Architecture ... 6647
Tools ... 6648
Best practices ... 6649
Epics ... 6649
Troubleshooting ... 6652
Related resources ... 6652
Deploy a pipeline that detects security issues in code ... 6654
Summary ... 6654
Prerequisites and limitations ... 6654
Architecture ... 6654
Tools ... 6656
Epics ... 6657
Troubleshooting ... 6659
Related resources ... 6660
Additional information ... 6660
Deploy detective controls for public subnets ... 6662
Summary ... 6662
Prerequisites and limitations ... 6663
Architecture ... 6663
Tools ... 6667
Best practices ... 6668
Epics ... 6668
Related resources ... 6676
Additional information ... 6677
Deploy preventative controls for public subnets ... 6679
Summary ... 6679
Prerequisites and limitations ... 6679
Architecture ... 6680

Tools ... 6682
Epics ... 6682
Related resources ... 6689
Additional information ... 6689
Detect Amazon RDS instances with expiring CA certificates ... 6692
Summary ... 6692
Prerequisites and limitations ... 6692
Architecture ... 6693
Tools ... 6696
Best practices ... 6697
Epics ... 6697
Troubleshooting ... 6701
Related resources ... 6702
Dynamically generate an IAM policy with IAM Access Analyzer ... 6703
Summary ... 6703
Prerequisites and limitations ... 6703
Architecture ... 6704
Tools ... 6706
Epics ... 6707
Related resources ... 6713
Enable GuardDuty using CloudFormation templates ... 6714
Summary ... 6714
Prerequisites and limitations ... 6714
Architecture ... 6714
Tools ... 6715
Epics ... 6716
Related resources ... 6718
Additional information ... 6718
Enable transparent data encryption in Amazon RDS for SQL Server ... 6722
Summary ... 6722
Prerequisites and limitations ... 6722
Architecture ... 6722
Tools ... 6723
Epics ... 6724
Related resources ... 6726
Ensure AWS load balancers use secure listener protocols ... 6728

- Summary ... 6728
- Prerequisites and limitations ... 6728
- Architecture ... 6729
- Tools ... 6729
- Best practices ... 6730
- Epics ... 6730
- Troubleshooting ... 6733
- Related resources ... 6734
- Attachments ... 6734

Ensure encryption for Amazon EMR data at rest ... 6735
- Summary ... 6735
- Prerequisites and limitations ... 6735
- Architecture ... 6736
- Tools ... 6737
- Epics ... 6737
- Related resources ... 6740
- Attachments ... 6740

Ensure that an IAM profile is associated with an EC2 instance ... 6741
- Summary ... 6741
- Prerequisites and limitations ... 6741
- Architecture ... 6742
- Tools ... 6742
- Epics ... 6743
- Related resources ... 6746
- Attachments ... 6746

Ensure that new Amazon Redshift clusters are encrypted ... 6747
- Summary ... 6747
- Prerequisites and limitations ... 6747
- Architecture ... 6747
- Tools ... 6748
- Epics ... 6749
- Related resources ... 6751
- Attachments ... 6751

Help prevent scheduled KMS key deletion ... 6752
- Summary ... 6752
- Prerequisites and limitations ... 6752

- Architecture ... 6753
- Tools ... 6754
- Epics ... 6755
- Related resources ... 6758
- Additional information ... 6759
- Attachments ... 6759

Identify public Amazon S3 buckets in AWS Organizations ... 6760
- Summary ... 6760
- Prerequisites and limitations ... 6760
- Architecture ... 6761
- Tools ... 6762
- Epics ... 6763
- Troubleshooting ... 6767
- Related resources ... 6768
- Additional information ... 6768

Ingest AWS security logs into Microsoft Sentinel ... 6769
- Summary ... 6769
- Prerequisites and limitations ... 6769
- Architecture ... 6771
- Tools ... 6772
- Best practices ... 6773
- Epics ... 6774
- Related resources ... 6784

Manage AWS Organizations policies as code ... 6785
- Summary ... 6785
- Prerequisites and limitations ... 6785
- Architecture ... 6786
- Tools ... 6788
- Best practices ... 6790
- Epics ... 6790
- Troubleshooting ... 6807
- Related resources ... 6809
- Additional information ... 6810

Manage IAM Identity Center permission sets using CodePipeline ... 6812
- Summary ... 6812
- Prerequisites and limitations ... 6812

Architecture 6814
Tools 6815
Best practices 6816
Epics 6817
Troubleshooting 6827
Related resources 6827
Manage credentials with AWS Secrets Manager 6828
Summary 6828
Prerequisites and limitations 6828
Architecture 6828
Tools 6829
Epics 6829
Related resources 6831
Additional information 6831
.... 6834
Summary 6834
Prerequisites and limitations 6834
Architecture 6835
Tools 6835
Epics 6836
Related resources 6839
Attachments 6839
Monitor Amazon EMR clusters for in-transit encryption at launch 6840
Summary 6840
Prerequisites and limitations 6841
Architecture 6841
Tools 6842
Epics 6843
Related resources 6845
Attachments 6845
Monitor Amazon ElastiCache clusters for at-rest encryption 6846
Summary 6846
Prerequisites and limitations 6847
Architecture 6847
Tools 6848
Epics 6849

Related resources 6851
Attachments 6852
Monitor EC2 instance key pairs 6853
Summary 6853
Prerequisites and limitations 6853
Architecture 6853
Tools 6854
Epics 6855
Related resources 6858
Attachments 6859
Monitor IAM root user activity 6860
Summary 6860
Prerequisites and limitations 6860
Architecture 6861
Tools 6862
Epics 6863
Related resources 6868
Additional information 6868
Notify when an IAM user is created 6869
Summary 6869
Prerequisites and limitations 6869
Architecture 6869
Tools 6870
Epics 6871
Related resources 6873
Attachments 6874
Prevent internet access by using an SCP 6875
Summary 6875
Prerequisites and limitations 6875
Tools 6876
Best practices 6876
Epics 6876
Related resources 6879
Report IAM Identity Center identities and their assignments 6880
Summary 6880
Prerequisites and limitations 6880

Architecture ... 6882
Tools ... 6883
Epics ... 6883
Troubleshooting ... 6885
Related resources ... 6886
Additional information ... 6886
Restrict access based on IP address or geolocation ... 6889
Summary ... 6889
Prerequisites and limitations ... 6889
Tools ... 6890
Epics ... 6891
Related resources ... 6896
Scan Git repositories for sensitive information ... 6897
Summary ... 6897
Prerequisites and limitations ... 6897
Architecture ... 6897
Tools ... 6898
Best practices ... 6898
Epics ... 6899
Related resources ... 6903
Secure file transfers using Transfer Family and GuardDuty ... 6905
Summary ... 6905
Prerequisites and limitations ... 6905
Architecture ... 6906
Tools ... 6908
Best practices ... 6909
Epics ... 6910
Troubleshooting ... 6914
Related resources ... 6917
Additional information ... 6917
Send alerts from AWS Network Firewall to a Slack channel ... 6919
Summary ... 6919
Prerequisites and limitations ... 6919
Architecture ... 6920
Tools ... 6921
Epics ... 6922

Related resources 6928
Additional information 6928
Send custom attributes to Amazon Cognito tokens 6933
Summary 6933
Prerequisites and limitations 6933
Architecture 6934
Tools 6936
Best practices 6937
Epics 6938
Troubleshooting 6947
Related resources 6947
Additional information 6947
Simplify private certificate management by using AWS Private CA and AWS RAM 6951
Summary 6951
Prerequisites and limitations 6951
Architecture 6952
Tools 6954
Epics 6956
Related resources 6962
Additional information 6963
Streamline Amazon EC2 compliance management with Amazon Bedrock agents and AWS Config 6964
Summary 6964
Prerequisites and limitations 6964
Architecture 6965
Tools 6967
Best practices 6968
Epics 6968
Troubleshooting 6970
Related resources 6971
Turn off security standard controls in Security Hub member accounts 6972
Summary 6972
Prerequisites and limitations 6972
Architecture 6973
Tools 6974
Epics 6975

Related resources 6979
Update credentials from IAM Identity Center using PowerShell 6980
Summary 6980
Prerequisites and limitations 6980
Architecture 6981
Tools 6982
Best practices 6982
Epics 6982
Troubleshooting 6985
Related resources 6985
Additional information 6985
Use AWS Config to monitor Amazon Redshift 6988
Summary 6988
Prerequisites and limitations 6988
Architecture 6989
Tools 6990
Epics 6991
Related resources 6993
Additional information 6994
Use Network Firewall to capture DNS names from outbound traffic 6995
Summary 6995
Prerequisites and limitations 6995
Architecture 6996
Tools 6997
Epics 6997
Use Terraform to automatically enable GuardDuty 7012
Summary 7012
Prerequisites and limitations 7013
Architecture 7014
Tools 7016
Epics 7017
Related resources 7026
Additional information 7027
Verify best practices for PCI DSS 4.0 7028
Summary 7028
Prerequisites and limitations 7028

Tools ... 7030
Epics ... 7030
Related resources ... 7033
Additional information ... 7033
Attachments ... 7034
... 7035
Summary ... 7035
Prerequisites and limitations ... 7035
Architecture ... 7036
Tools ... 7036
Epics ... 7037
Related resources ... 7040
Attachments ... 7040
... 7041
Summary ... 7041
Prerequisites and limitations ... 7041
Architecture ... 7042
Tools ... 7042
Epics ... 7043
Related resources ... 7046
Attachments ... 7046
More patterns ... 7047

AWS Prescriptive Guidance Patterns

Amazon Web Services (AWS) Prescriptive Guidance patterns provide step-by-step instructions, architecture, tools, and code for implementing specific cloud migration, modernization, and deployment scenarios. These patterns, which are vetted by subject matter experts at AWS, are meant for builders and hands-on users who are planning to, or are in the process of, migrating to AWS. They also support users who are already on AWS and are looking for ways to optimize or modernize their cloud operations.

You can use these patterns to move your on-premises or cloud workloads of varying complexity to AWS and to accelerate your cloud adoption, optimization, and modernization efforts, regardless of whether you're in the proof of concept, planning, or implementation phase of your project. For example, for a cloud migration project:

- In the planning phase, you can evaluate the different options available to migrate to AWS. You can choose the right pattern that fits your needs, depending on whether you want to relocate, rehost, replatform, or rearchitect. You can also understand the different tools available for migration, and start planning to procure licenses or start initial conversations with vendors.
- In the proof of concept and implementation phases, you can follow the step-by-step instructions provided in the pattern to migrate your workload to AWS. Each pattern includes details such as prerequisites, target reference architectures, tools, step-by-step tasks, best practices, troubleshooting, and code.
- If you're already using the AWS Cloud, you can find patterns that will help you modernize, optimize, scale, and secure your use of cloud resources.

To view lists of patterns by technical domain, use the following links or the filtering and search options on the AWS Prescriptive Guidance home page.

- Cloud foundations
- AI & machine learning
- Analytics
- Compute
- Databases & storage
- Developer tools
- IoT

- Migration & modernization
- Management
- Security, identity & compliance

To view all publications, including guides, strategies, and patterns, see the AWS Prescriptive Guidance home page.

Cloud foundations

Topics

- Automate account creation by using the Landing Zone Accelerator on AWS
- Automatically inventory AWS resources across multiple accounts and Regions
- Configure VPC Flow Logs for centralization across AWS accounts
- Manage AWS permission sets dynamically by using Terraform
- Tag Transit Gateway attachments automatically using AWS Organizations
- More patterns

Automate account creation by using the Landing Zone Accelerator on AWS

Justin Kuskowski, Joe Behrens, and Nathan Scott, Amazon Web Services

Summary

This pattern explains how to use the Landing Zone Accelerator on AWS solution to automatically deploy a new AWS account when an authorized user submits a request. It uses AWS Step Functions to orchestrate a number of AWS Lambda functions. The Lambda functions add the account information to a Git repository, initiate an AWS CodePipeline pipeline, and validate that the necessary AWS resources are provisioned. When the process is complete, the user receives a notification that the account has been created.

Optionally, you can integrate Microsoft Entra ID groups and assign AWS IAM Identity Center permission sets during the account creation process. If your organization is using Microsoft Entra ID as an identity source, this optional feature helps you automatically manage and configure access to the new account.

Prerequisites and limitations

Prerequisites

- Access to the management account in AWS Organizations
- AWS Cloud Development Kit (AWS CDK) version 2.118.0 or later, installed and configured
- Python version 3.9 or later, installed
- AWS Command Line Interface (AWS CLI) version 2.13.19 or later, installed
- Docker version 24.0.6 or later, installed
- Landing Zone Accelerator on AWS solution, deployed in the management account
- (Optional) Microsoft Entra ID and IAM Identity Center, integrated

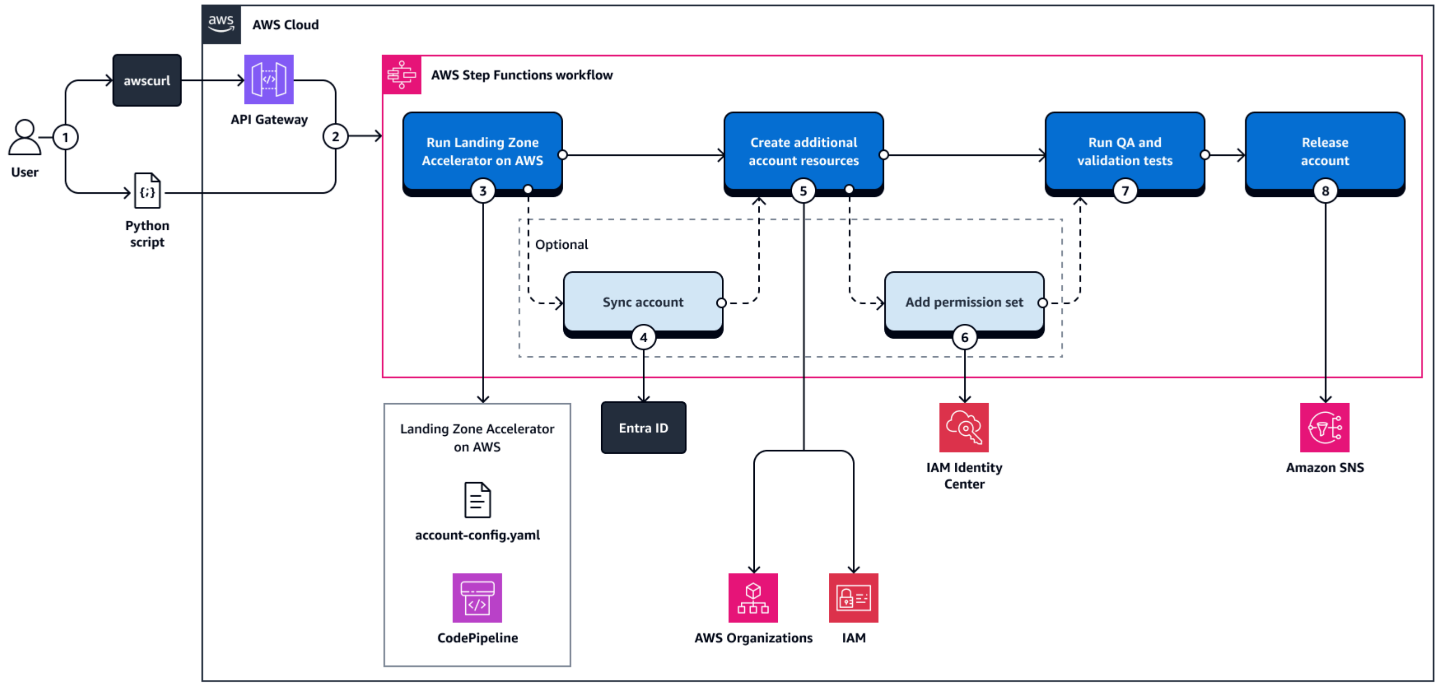
Limitations

The account creation workflow supports sequential executions to deploy a single AWS account. This limitation makes sure that the account creation workflow has successfully completed without having to compete for resources during a parallel run.

Architecture

Target architecture

The following image shows the high-level architecture of automating creation of a new AWS account by using the Landing Zone Accelerator on AWS. AWS Step Functions orchestrates the automation. Each task in the Step Functions workflow is performed by one or more AWS Lambda functions.



The diagram shows the following workflow:

1. The user requests an account by either running a Python script or by using Amazon API Gateway.
2. The Account Creation Orchestrator workflow starts in AWS Step Functions.
3. The workflow updates the `account-config.yaml` file in the source code repository. It also initiates the Landing Zone Accelerator on AWS pipeline and checks the status of the pipeline. This pipeline creates and sets up the new account. For more information about how this works, see the [Architecture overview](#) for the Landing Zone Accelerator on AWS.
4. (Optional) When the pipeline is complete, the workflow checks whether the group exists in Microsoft Entra ID. If the group doesn't exist in Microsoft Entra ID, the workflow adds the group to Microsoft Entra ID.
5. The workflow performs additional steps that the Landing Zone Accelerator on AWS solution can't do. The default steps include:

- Creating an account alias in AWS Identity and Access Management (IAM)
- Attaching tags to the account in AWS Organizations
- Creating parameters in AWS Systems Manager Parameter Store based on the tags assigned to the account

6. (Optional) The workflow assigns one or more permission sets to the Microsoft Entra ID group that you previously specified. The permission sets allow users in the group to access the new account and allow them to perform the actions you configure.
7. An AWS Lambda function runs QA and validation tests. It validates resource creation, checks that tags have been created, and validates that security resources have been deployed.
8. The workflow releases the account and uses Amazon Simple Email Service (Amazon SES) to notify the user that the process has completed successfully.

For more information about the Step Functions workflow, see *Step Functions workflow diagram* in the Additional information section of this pattern.

Microsoft Entra ID applications

If you choose to integrate with Microsoft Entra ID, you create the following two applications when you deploy this pattern:

- An application that is linked to IAM Identity Center and makes sure that Microsoft Entra ID groups are available in IAM Identity Center. In this example, this Microsoft Entra ID application is named `LZA2`.
- An application that allows a Lambda function to communicate with Microsoft Entra ID and call the Microsoft Graph APIs. In this pattern, this application is named `create_aws_account`.

These applications collect data that is used to sync the Microsoft Entra ID group and assign permission sets.

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale. In this pattern, you use API Gateway to check the availability of AWS account names, initiate the AWS Step Functions workflow, and check the status of the Step Functions execution.

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts. This solution uses an EventBridge rule that starts a Lambda function if the Step Functions workflow state changes to `Failed`, `Timed-out`, or `Aborted`.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to all of your AWS accounts and cloud applications.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data. In this pattern, AWS KMS keys are used to encrypt data, such as data stored in Amazon Simple Storage Service (Amazon S3), Lambda environment variables, and data in Step Functions.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon Simple Email Service (Amazon SES) helps you send and receive emails by using your own email addresses and domains. After a new account is successfully created, you receive a notification through Amazon SES.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses. If an error occurs during the account creation process, Amazon SNS sends a notification to the email addresses that you configure.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management.

Other tools

- awscurl automates the process of signing AWS API requests and helps you make requests as a standard curl command.
- Microsoft Entra ID, formerly known as *Azure Active Directory*, is a cloud-based identity and access management service.
- Microsoft Graph APIs help you access data and intelligence in Microsoft cloud services, such as Microsoft Entra and Microsoft 365.

Code repository

The code for this pattern is available in the GitHub lza-account-creation-workflow repository.

The lambda_layer directory contains the following layers, which are referenced in more than one Lambda function:

- account_creation_helper – This layer includes the modules for assuming roles and checking the progress in AWS Service Catalog.
- boto3 – This layer includes the AWS SDK for Python (Boto3) module to make sure that AWS Lambda has the latest version.
- identity_center_helper – This layer supports calls to IAM Identity Center.

The lambda_src directory contains the following Lambda functions:

- AccountTagToSsmParameter – This function uses the tags attached to the account in AWS Organizations in order to create parameters in Parameter Store. Each parameter starts with an `/account/tags/` prefix.
- AttachPermissionSet – This function adds a permission set to an IAM Identity Center group.
- AzureADGroupSync – This function syncs the target Microsoft Entra ID group to IAM Identity Center.
- CheckForRunningProcesses – This function checks whether the `AWSAccelerator-Pipeline` pipeline is currently running. If the pipeline is running, the function delays the AWS Step Functions workflow.
- CreateAccount – This function uses AWS Service Catalog and AWS Control Tower to create the new AWS account.

- CreateAdditionalResources – This function creates the AWS resources that aren't managed by Landing Zone Accelerator or AWS CloudFormation, such as the account alias and AWS Service Catalog tags.
- GetAccountStatus – This function scans for the provisioned product in AWS Service Catalog to determine if the account creation process has completed.
- GetExecutionStatus – This function retrieves the status of a running or completed AWS Step Functions execution.
- NameAvailability – This function checks if an AWS account name already exists in AWS Organizations.
- ReturnResponse – If the account creation was successful, this function returns the ID of the new account. If the account creation was not successful, it returns an error message.
- RunStepFunction – This function runs the AWS Step Functions workflow that creates the account.
- SendEmailWithSES – This function sends emails to the users who are waiting for the account creation to finish.
- ValidateADGroupSyncToSSO – This function checks if the specified Microsoft Entra ID groups are synced with IAM Identity Center.
- ValidateResources – This function validates that all AWS Control Tower customizations have run successfully.

Best practices

We recommend the following naming conventions for the AWS CDK:

- Start all parameters with a `p` prefix.
- Start all conditions with a `c` prefix.
- Start all resources with an `r` prefix.
- Start all outputs with an `o` prefix.

Epics

Deploy the IAM roles for validation and tagging

Task	Description	Skills required
Prepare the Landing Zone Accelerator on AWS for customization.	1. In the Landing Zone Accelerator on AWS code repository, create a file named `customizations-config.yaml`. You use this file to define customizations for the core solution. For more information, see [Customizing the solution](). 2. In the `customizations-config.yaml` file, create a section called `cloudFormationStacks`.	AWS DevOps
Prepare to deploy the `lza-account-creation-validation` role.	Now, you customize the solution to deploy the `lza-account-creation-validation` IAM role in all accounts other than the management account. This role provides the `ValidateResources` Lambda function with read-only access to the new accounts. 1. Download the [account-creation-validation-role.yaml]() file from GitHub.	AWS DevOps

Task	Description	Skills required

	2. Save the file to the location that is indicated in the templates section of the `customizations-config.yaml` file.
	3. Open the `customizations-config.yaml` file.
	4. In the `cloudFormationStacks` section, add the following code. Update the target AWS Region as needed for your landing zone:
	<pre> - deploymentTargets:
	        organizationalUnits:
	          - Root
	        excludedAccounts:
	          - Management
	
	      description: IAM Role to allow Account Validation
	      name: lza-account-creation-validation
	      regions:
	        - us-east-1
	      template: cloudformation/account-creation-validation-role.yaml</pre>	

Task	Description	Skills required

	``
	          runOrder: 1
	          terminati
	onProtection: true
	          parameters:
	            - name:
	 pManagementAccount
	Id
	              value:
	 "{{ account Managemen
	t }}"
	``
	5. Save and close the `customizations-con fig.yaml` file.	

Task	Description	Skills required

Prepare to deploy the `account-tagging-to-ssm-parameter-role` role.	Now, you customize the solution to deploy the `account-tagging-to-ssm-parameter-role` IAM role in all accounts other than the management account. This role is used to create the parameters in AWS Systems Manager Parameter Store.
	
	1. Download the account-tagging-to-ssm-parameter-role.yaml file from GitHub.
	2. Save the file to the location that is indicated in the templates section of the `customizations-config.yaml` file.
	3. Open the `customizations-config.yaml` file.
	4. In the `cloudFormationStacks` section, add the following code. Update the target AWS Region as needed for your landing zone:
	
	`      - deploymentTargets:`
	`          organizationalUnits:`
	`            - Root`
		AWS DevOps

Task	Description	Skills required

	``
	            excludedA
	ccounts:
	              -
	 Management
	          descripti
	on: IAM Role to
	 create SSM Parameter
	s based on Account
	 Tagging
	          name: lza-
	account-tagging-to-
	ssm-parameter
	          regions:
	            - us-
	east-1
	          template:
	 cloudformation/acc
	ount-tagging-to-ss
	m-parameter-role.y
	aml
	          runOrder: 1
	          terminati
	onProtection: true
	
	          parameters:
	            - name:
	 pManagementAccount
	Id
	              value:
	 "{{ account Managemen
	t }}"
	``
	5. Save and close the `customizations-config.yaml` file.	

Task	Description	Skills required

Prepare to deploy the `config-log-validation-role` role.	Now, you customize the solution to deploy the `config-log-validation-role` IAM role in the log archive account. This role allows the `ValidateResources` Lambda function to access the Amazon S3 bucket for logging and access AWS Config rules.
	
	1. Download the config-log-validation-role.yaml file from GitHub.
	2. Save the file to the location that is indicated in the templates section of the `customizations-config.yaml` file.
	3. Open the `customizations-config.yaml` file.
	4. In the `cloudFormationStacks` section, add the following code. Update the target AWS Region as needed for your landing zone:
	
	`- deploymentTargets:`
	`accounts:`
	`- LogArchive`
		AWS DevOps

Task	Description	Skills required

	``
	          descripti
	on: IAM Role to
	 validate Config and
	 Logs
	          name:
	 lza-config-log-val
	idation-role
	          regions:
	            - us-
	east-1
	          template:
	 cloudformation/con
	fig-log-validation-
	role.yaml
	          runOrder: 1
	          terminati
	onProtection: true
	
	          parameters:
	            - name:
	 pManagementAccount
	Id
	              value:
	 "{{ account Managemen
	t }}"
	``
	5. Save, close, and commit the changes made to the `customizations-config.yaml` file.	

(Optional) Get data from Microsoft Entra ID

Task	Description	Skills required
Create the application that that allows a Lambda function to communicate with Microsoft Entra ID.	1. In the Microsoft Entra ID admin center, register the `create_aws_account` application. For instructi	Microsoft Entra ID

Task	Description	Skills required

	ons, see Register an application in the Microsoft documentation.
	2. Follow the instructions in Update an app's requested permissions in the Microsoft documentation to configure the following Microsoft Graph permissions for the `create_aws_account` application:
	• Application.Read.All
	  • Type: Application
	  • Admin consent required: Yes
	• AppRoleAssignment.ReadWrite.All
	  • Type: Application
	  • Admin consent required: Yes
	• Group.ReadWrite.All
	  • Type: Application
	  • Admin consent required: Yes
	• ServicePrincipalEndpoint.Read.All
	  • Type: Application
	  • Admin consent required: Yes
	• Synchronization.ReadWrite.All	

Task	Description	Skills required

	• Type: Application
	• Admin consent required: Yes
	• User.Read
	  • Type: Delegated
	  • Admin consent required: No	

Task	Description	Skills required
Retrieve values for the `create_aws_account` application.	Now, you retrieve the values that you need for the `create_aws_account` application. 1. In the Microsoft Entra ID admin center, navigate to **App registrations**, and then choose `create_aws_account` . 2. In the left pane, choose **Overview**. 3. On the **Overview** page, make note of the following values: • Application (client) ID • Directory (tenant) ID 4. In the left pane, under **Manage**, choose **Certificates & secrets**. 5. On the **Certificates & secrets** page, choose the **Client secrets** tab, and then make note of the following values: • Client secret value • Client secret ID	Microsoft Entra ID

Task	Description	Skills required
Create the application that integrates Microsoft Entra ID with IAM Identity Center.	In the Microsoft Entra ID admin center, register the `LZA2` application. For instructions, see [Register an application](#) in the Microsoft documentation.	Microsoft Entra ID
Retrieve values for the `LZA2` application.	Now, you retrieve the values that you need for the `LZA2` application. 1. In the Microsoft Entra ID admin center, navigate to **Enterprise applications**, and then choose `LZA2`. 2. In the left pane, choose **Overview**. 3. On the **Overview** page, make note of the following values: • Name • Object ID 4. In the left pane, under **Manage**, choose **Manifest**. 5. In the JSON file, in the `appRoles` section, locate the app role named `User`. 6. Make note of the `id` value for this app role.	Microsoft Entra ID

Task	Description	Skills required

Create a secret.	1. In the AWS CLI, enter the following command to create the variables . Use the values that you retrieved for the `create_aws_account` and LZA2 applications:
	`# Variables for create_aws_account app TENANT_ID='<Directory ID>' CLIENT_ID='<Application ID>' SECRET_ID='<Client secret ID>' SECRET_VALUE='<Client secret value>'`
	`# Variables for LZA2 app OBJECT_ID='<Object ID>' APP_ROLE_ID='<App role ID>' ENTERPRISE_APP_NAME='<Name>'`
	2. Enter the following command to create a secret named `GraphApiSecret` in AWS Secrets Manager:
	`aws secretsmanager create-secret \ --name GraphApiSecret \`
		AWS DevOps

Task	Description	Skills required
	``` --secret-string "{\"client_id\": \"${CLIENT_ID}\", \"tenant_id\": \"${TENANT_ID}\", \"object_id\": \"${OBJECT_ID}\", \"app_role_id\": \"${APP_ROLE_ID}\" , \"secret_value\": \"${SECRET_VALUE}\ ", \"secret_id\": \"${SECRET_ID}\"}" ``` If you need to update the secret in the future, you can update the variables and run the following command: ``` aws secretsmanager update-secret \ --secret-id GraphApiSecret \ --secret-string "{\"client_id\": \"${CLIENT_ID}\", \"tenant_id\": \"${TENANT_ID}\", \"object_id\": \"${OBJECT_ID}\", \"app_role_id\": \"${APP_ROLE_ID}\" , \"secret_value\": \"${SECRET_VALUE}\ ", \"secret_id\": \"${SECRET_ID}\"}" ```	

Deploy the solution

Task	Description	Skills required

Clone the source code.	1. Enter the following command to clone the [lza-account-creation-workflow](https://github.com/aws-samples/lza-account-creation-workflow) repository:
	`git clone https://github.com/aws-samples/lza-account-creation-workflow`
	2. Enter the following command to navigate into the `config` directory of the cloned repository:
	`cd lza-account-creation-workflow/config`
		DevOps engineer

Update the `deploy-config.yaml` file.	1. Open the `deploy-config.yaml` file. 2. Review the template and update the values as needed for deployment in your AWS environment. For example, update values for the following: • `accountCreationFailure` • `accountCompletionFromEmail` • `ssoLoginUrl` • `rootEmailPrefix`	AWS DevOps

Task	Description	Skills required
	• `rootEmailDomain` 3. If you're integrating with Microsoft Entra ID, do the following: • Set `enableAzureADIntegration` to `true`. • For `graphApiSecretName` value, enter the secret that you created previously (`GraphApiSecret`). 4. Save and close the `deploy-config.yaml` file.	

Task	Description	Skills required

Deploy the solution in your AWS environment.	1. Enter the following command:
	`cdk bootstrap <account-number>/<Region>`
	For more information, see [Bootstrapping](#) in the AWS CDK documentation.
	2. Enter the following command to synthesize the CloudFormation template:
	`cdk synth`
	For more information, see [Configure and perform AWS CDK stack synthesis](#) in the AWS CDK documenta tion.
	3. Enter the following command to deploy the solution.
	`cdk deploy`
	For more information, see [Deploy AWS CDK applications](#) in the AWS CDK documentation.
		AWS DevOps

Task	Description	Skills required
	Note This solution uses an Amazon S3 bucket to store the source code for this solution. You can use the upload_to_source_bucket.py script to create an archive of the source code and upload an updated version.	

Option 1 – Create an account using Python

Task	Description	Skills required
Identify which arguments to use.	Choose which arguments to use when you run the Python script that initiates the Step Functions workflow. For a complete list of arguments, see the [Additional information](#additional-information) section of this pattern.	AWS DevOps, Python
Start the Python script.	1. Enter the following command to navigate into the cloned repository: `cd lza-account-creation-workflow` 2. Run the Python script that initiates the Step Functions workflow. The following is	DevOps engineer, Python

Task	Description	Skills required
	an example command that includes sample arguments and values: `python ./run-stepfunction.py \ --account-name "lza-test-01" \ --support-dl "johnsmith@example.com" \ --managed-org-unit "Workloads/Workload-1" \ --purpose "Testing new micro service" \ --force-update true \ --ad-integration "{\"CustomerAccountAdmin\": \"platform-admin\", \"CustomerAccountDev\": \"workload1-app1\"}" \ --bypass-creation true \ --tags APPLICATION=TestingMicroService` 3. Wait until you receive a notification that the account has been successfully created. **Note** If the account creation process	

Task	Description	Skills required
	fails, Amazon SNS sends a notification to the email addresses that you define in `accountCreationFailure` in the `deploy-config.yaml` file. The account requester is not notified.	

Option 2 – Create an account using API Gateway and awscurl

Task	Description	Skills required
Set up the variables for awscurl.	1. Enter the following command to navigate into the source code directory: `cd lza-account-creation-workflow` 2. Enter the following command to set the AWS access key variables. You can copy the variables from the [AWS access portal](#) and then paste them into a shell. The following is an example:	AWS DevOps

Task	Description	Skills required

```
export AWS_ACCESS_KEY_ID="<id>"
export AWS_SECRET_ACCESS_KEY="<key>"
export AWS_SESSION_TOKEN="<token>"
```

3. Enter the following command to set the AWS Region for API calls:

```
export AWS_REGION=$(aws configure get region)
```

4. Enter the following command to retrieve the API Gateway endpoint from the `lza-account-creation-workflow-application` CloudFormation output:

```
export API_GATEWAY_ENDPOINT=$(aws cloudformation describe-stacks --stack-name "lza-account-creation-workflow-application" --query 'Stacks[*].Outputs[?OutputKey==`oApiGatewayCreateAccountEndpoint`].OutputValue' --output text)
```

Task	Description	Skills required
Check name availability.	Enter the following command to verify that the name is available for the AWS account. Replace <AWS_ACCOUNT_NAME> with the name of the target account:	AWS DevOps

```
awscurl --service execute-api \
    --region ${AWS_REGION} \
    --access_key ${AWS_ACCESS_KEY_ID} \
    --secret_key ${AWS_SECRET_ACCESS_KEY} \
    --security_token ${AWS_SESSION_TOKEN} \
    -X POST ${API_GATEWAY_ENDPOINT}check_name?account_name=<AWS_ACCOUNT_NAME>
```

Task	Description	Skills required

Run the account creation workflow.	1. In the root folder of the cloned repository, open the `api_example.json` file.
	2. Update the parameters with your configuration values:
	<pre>{
	    "account_name":
	 "lza-test-01",
	    "account_email":
	 "johnsmith+lzatest
	01@example.com",
	    "support_dl":
	 "johnsmith@example
	.com",
	    "managed_
	org_unit": "Workload
	s/Workload-1",
	    "ad_integration":
	 [
	        {
	            "Permissi
	onSetName":
	 "CustomerAccountAd
	min",
	            "ActiveDi
	rectoryGroupName":
	 "platform-admin"
	        },
	        {
	            "Permissi
	onSetName":
	 "CustomerAccountDe
	v",
	            "ActiveDi
	rectoryGroupName":
	 "workload1-app1"
	        }
	    ],</pre>
		AWS DevOps

Task	Description	Skills required

	``
	    "force_update": "true",
	    "bypass_creation": "false",
	    "account_tags": [
	        {
	            "Key": "Environment",
	            "Value": "Dev"
	        }, {
	            "Key": "DeploymentMethod",
	            "Value": "ApiGateway"
	        }
	    ]
	}
	``
	3. Save and close the `api_example.json` file.
	4. Enter the following command to initiate the Step Functions workflow:
	``
	awscurl --service execute-api \
	    --data @api-example.json \
	    --region ${AWS_REGION} \
	    --access_key ${AWS_ACCESS_KEY_ID} \
	    --secret_key ${AWS_SECRET_ACCESS_KEY} \
	    --security_token ${AWS_SESSION_TOKEN} \
	``	

Task	Description	Skills required

	``` -X POST ${API_GATEWAY_ENDPOINT}execute ```
	5. In the output from the previous command, make note of the Step Functions execution Amazon Resource Name (ARN).
	6. If you want to check the status of the Step Functions workflow, enter the following command. Replace `<STEP_FUNCTION_EXECUTION_NAME>` with the Step Functions execution ARN or name:
	``` awscurl --service execute-api \ --region ${AWS_REGION} \ --access_key ${AWS_ACCESS_KEY_ID} \ --secret_key ${AWS_SECRET_ACCESS_KEY} \ --security_token ${AWS_SESSION_TOKEN} \ -X GET ${API_GATEWAY_ENDPOINT}get_execution_status?execution=<STEP_FUN ```	

Task	Description	Skills required
	`CTION_EXECUTION_NAME>` 7. Wait until you receive a notification that the account has been successfully created. **Note** If the account creation process fails, Amazon SNS sends a notification to the email addresses that you define in `accountCreationFailure` in the `deploy-config.yaml` file. The account requester is not notified.	

(Optional) Clean up the solution

Task	Description	Skills required
Remove objects from the Amazon S3 buckets.	Remove any objects in the following Amazon S3 buckets: • `lza-account-creation-work-<CDK_UNIQUE_ID>`	AWS DevOps

Task	Description	Skills required
	• `lza-account-creation-workflow-src-<AWS_REGION>-<AWS_ACCOUNT>` • `lza-account-creation-workflow-<AWS_REGION>-<AWS_ACCOUNT>`	

Delete the CloudFormation stack.	Enter the following command to delete the CloudFormation stack:
	`aws cloudformation delete-stack \ --stack-name lza-account-creation-workflow-application`
	`aws cloudformation wait stack-delete-complete \ --stack-name lza-account-creation-workflow-application`
		AWS DevOps

Delete the pipeline.	Enter the following command to delete the `lza-account-creation-workflow-pipeline` pipeline: `cdk destroy lza-account-creation-workflow-pipeline --force`	AWS DevOps

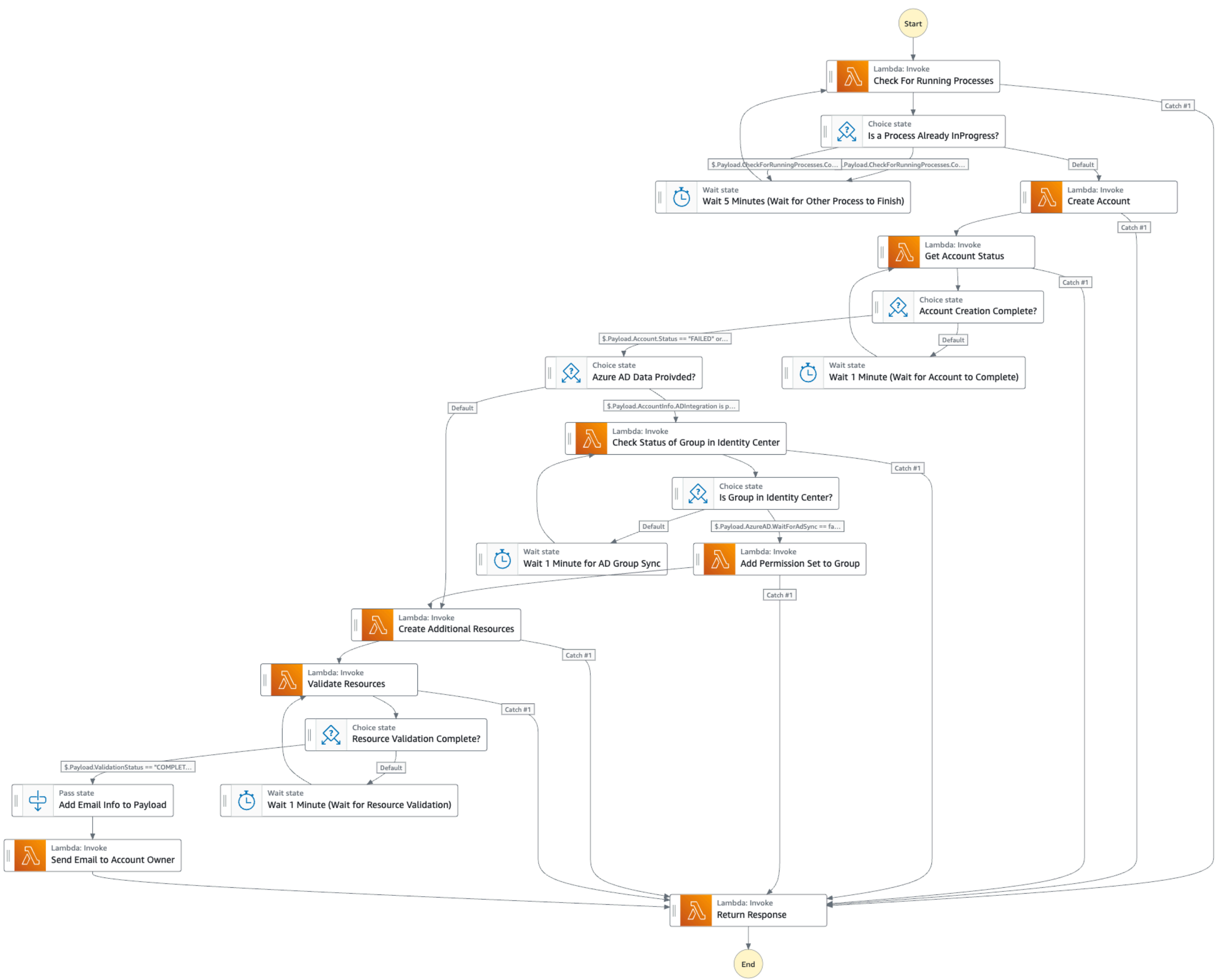
Related resources

- Landing Zone Accelerator on AWS (AWS Solutions Library)
- Troubleshooting common AWS CDK issues (AWS CDK documentation)

Additional information

Step Functions workflow diagram

The following image shows the states in the Step Functions workflow.



Arguments

The following are the arguments that you can use when you run the Python script that initiates the Step Functions workflow.

The following arguments are required:

- `account-name (-a)` (string) – The name of the new AWS account.
- `support-dl (-s)` (string) – The email address that receives notification when the account creation process is complete.
- `managed-org-unit (-m)` (string) – The managed organizational unit (OU) that will contain the new account.

The following arguments are optional:

- `ad-integration (-ad)` (string dictionary) – The Microsoft Entra ID group and assigned permission set. The following is an example of how to use this argument:

```
--ad-integration "{\"<PermissionSetName>\": \"<EntraIdGroupName>\"}"
```

- `account-email (-e)` (string) – The email address for the root user of the new AWS account.

> **Note**
>
> If this argument isn't used, an email address will be generated by using the values `rootEmailPrefix` and `rootEmailDomain` from the `configs/deploy-config.yaml` file. If an email address isn't provided, an email address is generated by using the following format: `rootEmailPrefix+accountName@rootEmailDomain`.

- `region (-r)` (string) – The AWS Region where the Step Functions workflow was deployed. The default value is `us-east-1`.
- `force-update (-f)` (string Boolean) – Enter `true` to force AWS Service Catalog to update the provisioned product.
- `bypass-creation (-b)` (string Boolean) – Enter `true` to bypass adding the account to the `accounts-config.yaml` file and bypass running the `AWSAccelerator-Pipeline` pipeline. This argument is typically used to test the account creation workflow process or to run the rest of the Step Functions steps if an error occurs in the `Landing Zone Accelerator` pipeline.

- `tags (-t)` (string) – Additional tags that you want to add to the AWS account. By default, the following tags are added: `account-name`, `support-dl`, and `purpose`. The following is an example of how to use this argument:

```
--tags TEST1=VALUE1 TEST2=VALUE2
```

Automatically inventory AWS resources across multiple accounts and Regions

Matej Macek, Amazon Web Services

Summary

This pattern outlines an automated approach to maintaining a comprehensive inventory of AWS resources across multiple accounts and AWS Regions. It is designed to help infrastructure and security engineers improve their resource management practices. It uses AWS Config to track resource changes, Amazon Athena for querying, and Amazon QuickSight for interactive dashboards. You implement this solution by deploying an AWS CloudFormation stack.

This solution is similar to the one presented in Visualizing AWS Config data using Amazon Athena and Amazon QuickSight (AWS blog post). This pattern expands on that solution to address the following common requirements and provide the following key benefits:

- **Compliance-focused –** This approach can help you meet regulatory requirements such as PCI DSS, NIST SP 800-53, ISO/IEC 27001, HIPAA, GDPR, and others that mandate accurate asset inventories.
- **Customization framework –** It provides a foundation for creating QuickSight dashboards for various AWS resources, so that you can customize the solution to your specific requirements.
- **User-driven enhancements –** This approach incorporates feedback from real-world use cases and addresses requests for a more comprehensive solution.

Infrastructure, security, and finance teams often face visibility and collaboration challenges in dynamic, multi-account or multi-Region environments. This solution is designed to address those challenges and significantly reduce the time and effort required to create and maintain a resource inventory. The result is a centralized view of resources that helps you improve resource allocation decisions, identify and mitigate risks, optimize costs, and improve overall visibility and collaboration. This approach bridges the gap between conceptual solutions and real-world implementation needs for security, compliance, and operational purposes.

Prerequisites and limitations

Prerequisites

- The following active AWS accounts:

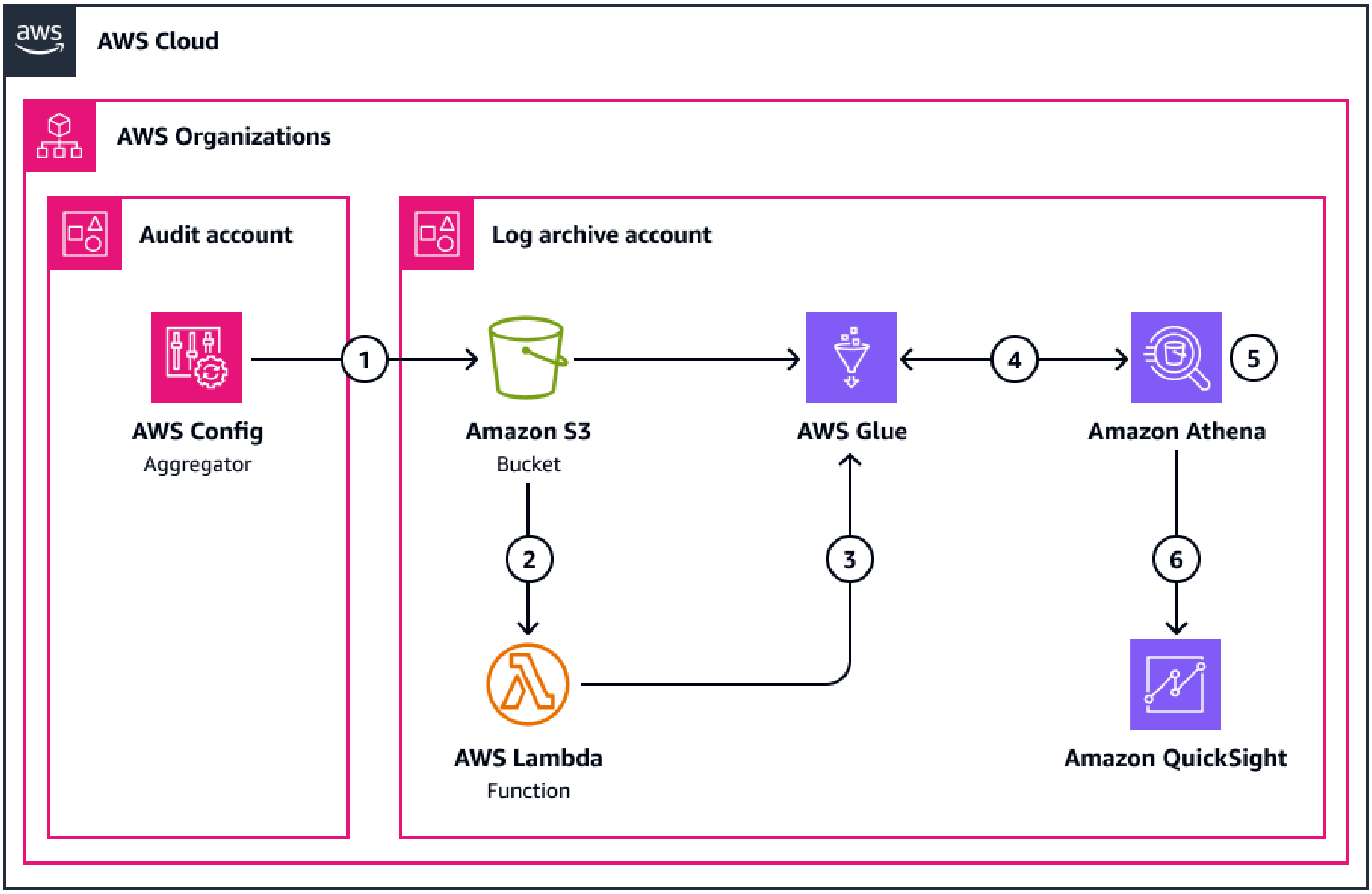
  - *Management account* - A centralized account for billing, creating accounts, and controlling access across the organization
  - *Audit account* – A centralized hub for security monitoring, compliance checks, and drift notifications
  - *Log archive account* – A centralized account for storing and analyzing the collected data
- In the audit account, an AWS Config aggregator that collects and aggregates configuration data from your target accounts and Regions
- In the log archive account, set up the following:
  - An Amazon Simple Storage Service (Amazon S3) bucket where you store the data from the AWS Config aggregator
  - An Amazon QuickSight subscription
  - An authorized connection between QuickSight and Amazon Athena
  - Permissions to access the Amazon S3 bucket through an Athena query
- AWS Command Line Interface (AWS CLI), installed and configured
- Permissions to deploy a CloudFormation stack that provisions the following resources:
  - An AWS Lambda function
  - An Amazon S3 notification configuration
  - Athena database, tables, and views
  - QuickSight datasets and data sources
- Permissions to run automations in AWS Systems Manager
- Permissions to access QuickSight

Limitations

- The solution relies on AWS Config. AWS Config usually records configuration changes to your resources right after a change is detected, or at the frequency that you specify. However, this is on a best-effort basis and can take longer at times.
- This solution tracks only resource types that AWS Config supports.
- The solution does not track resource inventory across other cloud providers or on-premises environments.
- Some AWS services aren't available in all AWS Regions. For Region availability, see the Service endpoints and quotas page in the AWS documentation, and choose the link for the service.

Architecture

The following diagram shows a streamlined process for collecting, organizing, analyzing, and visualizing configuration and compliance data across multiple accounts in an AWS organization.



The diagram shows the following workflow:

1. On a periodic schedule, the AWS Config aggregator collects configuration and compliance data about the resources in the target accounts and Regions and then delivers the data to the Amazon S3 bucket in the log archive account.
2. Adding new AWS Config data to the Amazon S3 bucket invokes an AWS Lambda function.
3. The Lambda function partitions the data by configuring keys with values that correspond to the Region and date of each snapshot file. This helps AWS Glue efficiently query and process the configuration and compliance data.
4. Amazon Athena uses an AWS Glue schema to run SQL queries against the data stored in the Amazon S3 bucket. It utilizes the schema metadata from AWS Glue to understand the structure of the data.

5. Views in Athena define and extract the target datasets.
6. Dashboards in Amazon QuickSight help you to visualize and analyze the datasets.

Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time. An AWS Config aggregator collects AWS Config configuration and compliance data from multiple AWS accounts and Regions.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams. This pattern uses an AWS Glue Data Catalog and Schema registry.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data in a single dashboard.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale. AWS Systems Manager Automation simplifies common maintenance, deployment, and remediation tasks for many AWS services.

Code repository

The AWS CloudFormation template for this pattern is available in the AWS Config visualization GitHub repository. This CloudFormation template deploys an AWS Systems Manager automation runbook that sets up AWS Config for use with Amazon Athena. This automation prepares AWS Glue to connect with the designated Amazon S3 bucket, creates views in Amazon Athena, and configures Amazon QuickSight for dashboard visualization.

Best practices

- We recommend that you follow the best practices in Set up and govern a secure, multi-account AWS environment with AWS Control Tower on AWS Prescriptive Guidance.
- We recommend that you create an AWS Config aggregator that collects configuration and compliance data for the entire AWS organization. For more information, see Multi-Account Multi-Region Data Aggregation in the AWS Config documentation.
- Before deploying this solution, we recommend that you review the current pricing information for Amazon S3, AWS Config, Athena, and QuickSight.

Epics

Deploy the CloudFormation stack

Task	Description	Skills required
Download the CloudFormation template.	Download the Config-QuickSight-Visualization-SSM-Automation.yaml CloudFormation template.	AWS administrator, Cloud administrator, DevOps engineer
Modify the CloudFormation template.	Complete this step only if you're using AWS Control Tower and AWS Config is managed by AWS Control Tower. You need to modify the CloudFormation template. 1. Sign in to the management account.	DevOps engineer, AWS administrator

Task	Description	Skills required
	2. Open the [AWS Organizations console](#). 3. Navigate to the **Settings** page. This page displays details about the organization, including the organization ID. 4. Copy the organization ID. 5. In your preferred text editor, open the **Config-QuickSight-Visualization-SSM-Automation.yaml** file. 6. Find the following line: `return re.match( '^AWSLogs/(\d+)/Config/([\w-]+)/(\d+)/(\d+)/(\d+)/ConfigSnapshot/[^\\\]+$', object_key)` 7. Replace this line with the following, where `<ORGANIZATION_ID>` is the ID that you previously copied: `return re.match( '^<ORGANIZATION_ID>/AWSLogs/(\d+)/Config/([\w-]+)/(\d+)/(\d+)/(\d+)/ConfigSnapshot/[^\\\]+$', object_key)`	

Task	Description	Skills required
	8. Save and close the **Config-QuickSight-Visualization-SSM-Automation.yaml** file.	
Create a CloudFormation stack.	Follow the instructions in [Create a stack from the CloudFormation console](). Note the following: 1. Choose **Upload a template file**, and then choose the YAML file you downloaded. 2. For **Stack name**, enter `Config-QuickSight-Visualization-SSM-Automation` . 3. Choose **Submit**.	AWS administrator, Cloud administrator, DevOps engineer

Run the automation in Systems Manager

Task	Description	Skills required
Find your QuickSight user name.	1. Open the [QuickSight console](). 2. Open the profile menu. 3. Make note of the user name. You need this value later.	AWS administrator, Cloud administrator, DevOps engineer
Find the delivery channel name and Amazon S3 bucket name.	1. In the AWS CLI, enter the following command:	AWS administrator, Cloud administrator, DevOps engineer

Task	Description	Skills required

	<pre>aws configservice
	 describe-delivery-
	channels</pre>2. Make note of the Amazon S3 bucket name and the name of your the AWS Config delivery channel. You need these values later.	

Task	Description	Skills required
Run the automation in Systems Manager.	1. Open the AWS Systems Manager console. 2. In the navigation pane, choose **Documents**. 3. Choose **Owned by me**. 4. Choose **Config-QuickSight-Visualization**. 5. Choose **Execute automation**. 6. In the **Input parameters** section, enter your values for the following parameters: • `ConfigDeliveryChannelName` – Enter the name of your AWS Config delivery channel. This parameter is required. • `ConfigS3BucketLocation` – Enter the name of the Amazon S3 bucket where you store AWS Config configuration data. This parameter is required. • `QuickSightUserName` – Enter a user name that has administrative access to QuickSight. This parameter is required.	AWS administrator, Cloud administrator, DevOps engineer

Task	Description	Skills required
	• `AutomationAssumeRole` – The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that allows Systems Manager Automation to perform the actions on your behalf. This parameter is optional. Leave this parameter blank. • `DeleteConfigVisualization` – Choose `false`. 7. Choose **Execute**.	

Visualize data in Amazon QuickSight

Task	Description	Skills required
Refresh data.	To schedule dataset refreshes according to your specific requirements, follow the instructions in [Refreshing SPICE data]().	AWS administrator, DevOps engineer, Cloud administrator
Create an analysis.	To create a dashboard in QuickSight that helps you visualize the resources, follow the instructions in [Starting an analysis in Amazon QuickSight]().	QuickSight administrator

Task	Description	Skills required
Create a dashboard.	1. After you finish modifying your QuickSight analysis, follow the instructions in Publishing dashboards to create a dashboard. A *dashboard* is an analysis that you can share with other QuickSight users. 2. Follow the instructions in Granting access to a dashboard to share the dashboard with your target QuickSight users.	QuickSight administrator

(Optional) Clean up

Task	Description	Skills required
Delete the resources created by the Systems Manager automation.	1. Open the AWS Systems Manager console. 2. In the navigation pane, choose **Documents**. 3. Choose **Owned by me**. 4. Choose **Config-QuickSight-Visualization**. 5. Choose **Execute automation**. 6. In the **Input parameters** section, for the `DeleteConfigVisualization` parameter, enter `true`. 7. Choose **Execute**.	AWS administrator, Cloud administrator, DevOps engineer

Task	Description	Skills required
Delete the CloudFormation stack.	To delete the resources in the `Config-QuickSight-Visualization-SSM-Automation` stack, follow the instructions in [Delete a stack from the CloudFormation console]().	AWS administrator, Cloud administrator, DevOps engineer

Troubleshooting

Issue	Solution

Amazon QuickSight is attempting to connect to the `us-east-1` AWS Region, but the creation of resources in that Region is not permitted.	A service control policy is restricting your subscription to Amazon QuickSight in this Region. In the service control policy, manually specify the target AWS Region. Replace `<REGION_ID>` with the appropriate Region identifier:
	`https://<REGION_ID>.quicksight.aws.amazon.com/sn/start/dashboards`
	The following is an example:
	`https://eu-central-1.quicksight.aws.amazon.com/sn/start/dashboards`

In Amazon Athena, you encounter the following message: `Before you run your first query, you need to set up a query result location in Amazon S3.`	Make sure that you have prepared an Amazon S3 bucket where you will store the query results from Amazon Athena. Then follow the instructions in [Specify a query result location using the Amazon Athena console]().

Related resources

AWS documentation

- AWS Config documentation
- Amazon QuickSight documentation

AWS blog post

- Automate AWS Config data visualization with AWS Systems Manager
- How to record resource configuration changes periodically with AWS Config

Other resources

- Amazon QuickSight Community Learning Center
- Amazon QuickSight Community Gallery

Configure VPC Flow Logs for centralization across AWS accounts

Benjamin Morris and Aman Kaur Gandhi, Amazon Web Services

Summary

In an Amazon Web Services (AWS) virtual private cloud (VPC), the VPC Flow Logs feature can provide useful data for operational and security troubleshooting. However, there are limitations on using VPC Flow Logs in a multi-account environment. Specifically, cross-account flow logs from Amazon CloudWatch Logs are not supported. Instead, you can centralize the logs by configuring an Amazon Simple Storage Service (Amazon S3) bucket with the appropriate bucket policy.

Note

This pattern discusses the requirements for sending flow logs to a centralized location. However, if you also want logs to be available locally in member accounts, you can create multiple flow logs for each VPC. Users without access to the Log Archive account can see traffic logs for troubleshooting. Alternatively, you can configure a single flow log for each VPC that sends logs to CloudWatch Logs. You can then use an Amazon Data Firehose subscription filter to forward the logs to an S3 bucket. For more information, see the Related resources section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS Organizations organization with an account that is used to centralize logs (for example, Log Archive)

Limitations

If you use the AWS Key Management Service (AWS KMS) managed key `aws/s3` to encrypt your central bucket, it won't receive logs from a different account. Instead, you will see an

`Unsuccessful` error code 400 with a message such as "`LogDestination: <bucketName> is undeliverable`" for your given `ResourceId`.

This is because an account's AWS managed keys can't be shared across accounts.

The solution is to use either Amazon S3 managed encryption (SSE-S3) or an AWS KMS customer managed key that you can share with member accounts.

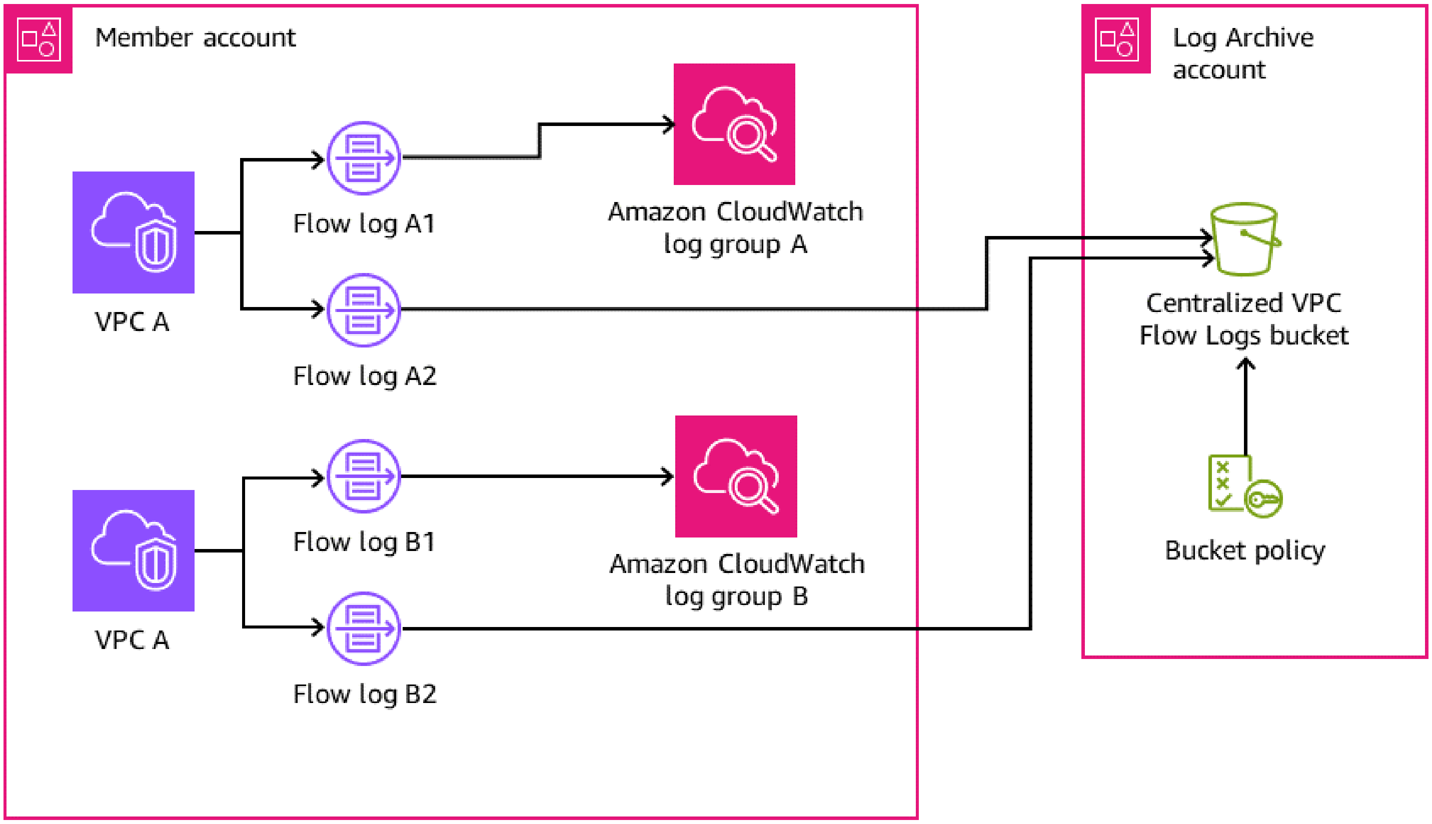
Architecture

Target technology stack

In the following diagram, two flow logs are deployed for each VPC. One sends logs to a local CloudWatch Logs group. The other sends logs to an S3 bucket in a centralized logging account. The bucket policy permits the log delivery service to write logs to the bucket.

> **Note**
>
> As of November 2023, AWS now supports the aws:SourceOrgID condition key. This condition allows you to deny writing to the centralized bucket for accounts outside of your AWS Organizations organization.

Target architecture



Automation and scale

Each VPC is configured to send logs to the S3 bucket in the central logging account. Use one of the following automation solutions to help ensure that flow logs are configured appropriately:

- [AWS CloudFormation StackSets]
- [AWS Control Tower Account Factory for Terraform (AFT)]
- [An AWS Config rule with remediation]

Tools

Tools

- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

- [Amazon Virtual Private Cloud (Amazon VPC)] helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. This pattern uses the VPC Flow Logs feature to capture information about the IP traffic going to and from network interfaces in your VPC.

Best practices

Using infrastructure as code (IaC) can greatly simplify the VPC Flow Logs deployment process. Abstracting your VPC deployment definitions to include a flow log resource construct will deploy your VPCs with flow logs automatically. This is demonstrated in the next section.

Centralized flow logs

Example syntax for adding centralized flow logs to a VPC module in HashiCorp Terraform

This code creates a flow log that sends logs from a VPC to a centralized S3 bucket. Note that this pattern doesn't cover creation of the S3 bucket.

For recommended bucket policy statements, see the Additional information section.

```
variable "vpc_id" { type = string }
locals { custom_log_format_v5 = "$${version} $${account-id} $${interface-id} $
${srcaddr} $${dstaddr} $${srcport} $${dstport} $${protocol} $${packets} $${bytes}
 $${start} $${end} $${action} $${log-status} $${vpc-id} $${subnet-id} $${instance-
id} $${tcp-flags} $${type} $${pkt-srcaddr} $${pkt-dstaddr} $${region} $${az-id} $
${sublocation-type} $${sublocation-id} $${pkt-src-aws-service} $${pkt-dst-aws-service}
 $${flow-direction} $${traffic-path}" }
resource "aws_flow_log" "centralized_flow_log" {
  log_destination      = "arn:aws:s3:::centralized-vpc-flow-logs-
<log_archive_account_id>" # Optionally, a prefix can be added after the ARN.
  log_destination_type = "s3"
  traffic_type         = "ALL"
  vpc_id               = var.vpc_id
  log_format           = local.custom_log_format_v5 # If you want fields from VPC Flow
 Logs v3+, you will need to create a custom log format.
}
```

For more information about the custom log format, see the AWS documentation.

Local flow logs

Example syntax for adding local flow logs to a VPC module in Terraform with required permissions

This code creates a flow log that sends logs from a VPC to a local CloudWatch Logs group.

```
data "aws_region" "current" {}
variable "vpc_id" { type = string }
resource "aws_iam_role" "local_flow_log_role" {
  name = "flow-logs-policy-${var.vpc_id}"
  assume_role_policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Principal": {"Service": "vpc-flow-logs.amazonaws.com"},
      "Action": "sts:AssumeRole"
  }]
}
EOF
}
resource "aws_iam_role_policy" "logs_permissions" {
  name = "flow-logs-policy-${var.vpc_id}"
  role = aws_iam_role.local_flow_log_role.id
  policy = <<EOF
{
  "Version": "2012-10-17",
  "Statement": [{
      "Action": ["logs:CreateLog*", "logs:PutLogEvents", "logs:DescribeLog*",
 "logs:DeleteLogDelivery"],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:${data.aws_region.current.name}:*:log-group:vpc-flow-
logs*"
  }]
}
EOF
}
resource "aws_cloudwatch_log_group" "local_flow_logs" {
  name              = "vpc-flow-logs/${var.vpc_id}"
  retention_in_days = 30
}
resource "aws_flow_log" "local_flow_log" {
  iam_role_arn    = aws_iam_role.local_flow_log_role.arn
  log_destination = aws_cloudwatch_log_group.local_flow_logs.arn
  traffic_type    = "ALL"
  vpc_id          = var.vpc_id
```

```
}
```

Epics

Deploy VPC Flow Logs infrastructure

Task	Description	Skills required
Determine the encryption strategy and create the policy for the central S3 bucket.	The central bucket does not support the `aws/s3` AWS KMS key, so you must use either SSE-S3 or an AWS KMS customer managed key. If you use an AWS KMS key, the key policy must allow member accounts to use the key.	Compliance
Create the central flow log bucket.	Create the central bucket to which flow logs will be sent, and apply the encryption strategy that you chose in the previous step. This should be in a Log Archive or similarly purposed account. Obtain the bucket policy from the [Additional information](#) section, and apply it to your central bucket after updating placeholders with your environment specific values.	General AWS
Configure VPC Flow Logs to send logs to the central flow log bucket.	Add flow logs to each VPC that you want to gather data from. The most scalable way to do this is to use IaC tools such as AFT or AWS Cloud	Network administrator

Task	Description	Skills required
	Development Kit (AWS CDK). For example, you can create a Terraform module that deploys a VPC alongside a flow log. If necessary, you add the flow logs manually.	
Configure VPC Flow Logs to send to local CloudWatch Logs.	(Optional) If you want flow logs to be visible in the accounts where the logs are being generated, create another flow log to send data to CloudWatch Logs in the local account. Alternatively, you can send the data to an account-specific S3 bucket in the local account.	General AWS

Related resources

- [How to Facilitate Data Analysis and Fulfill Security Requirements by Using Centralized Flow Log Data](#) (blog post)
- [How to enable VPC Flow Logs automatically using AWS Config rules](#) (blog post)

Additional information

Bucket policy

This example of a bucket policy can be applied to your central S3 bucket for flow logs, after you add values for placeholder names.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AWSLogDeliveryWrite",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<BUCKET_NAME>/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceOrgID": "<ORG_ID>"
                }
            }
        },
        {
            "Sid": "AWSLogDeliveryCheck",
            "Effect": "Allow",
            "Principal": {
                "Service": "delivery.logs.amazonaws.com"
            },
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::<BUCKET_NAME>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceOrgID": "<ORG_ID>"
                }
            }
        },
        {
            "Sid": "DenyUnencryptedTraffic",
            "Effect": "Deny",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::<BUCKET_NAME>/*",
                "arn:aws:s3:::<BUCKET_NAME>"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
```

```
        }
    ]
}
```

Manage AWS permission sets dynamically by using Terraform

Vinicius Elias and Marcos Vinicius Pinto Jordao, Amazon Web Services

Summary

AWS IAM Identity Center enhances AWS Identity and Access Management (IAM) by providing a centralized hub for managing single sign-on access to AWS accounts and cloud applications. However, manual management of IAM Identity Center permission sets can become increasingly complex and error-prone as your organization grows. This complexity can lead to potential security gaps and administrative overhead.

This solution enables you to manage permission sets through infrastructure as code (IaC) using a continuous integration and continuous delivery (CI/CD) pipeline built with native AWS services. It enables a seamless integration of the permission set assignment mechanism with AWS Control Tower lifecycle events or an Account Factory for Terraform (AFT) environment. This approach provides dynamic identity configurations for both new and existing AWS accounts.

Amazon EventBridge rules monitor AWS account creation and updates, which helps your identity configurations to remain synchronized with your organizational structure. After creating or updating accounts in AWS Control Tower or AFT, the pipeline is triggered. It evaluates a set of JSON files with permission set definitions and assignment rules. Then the pipeline applies and synchronizes the settings across all accounts.

This approach provides the following benefits:

- **Consistency** – Eliminates manual configuration drift across your AWS organization
- **Auditability** – Maintains a complete history of all identity management changes
- **Scalability** – Automatically applies configurations as your AWS environment grows
- **Security** – Reduces human error in permission assignments
- **Compliance** – Facilitates meeting regulatory requirements through documented changes and assignment rules

Prerequisites and limitations

- A multi-account environment with AWS Control Tower and AWS Organizations set up. Optionally, you can use AFT with AWS Control Tower.

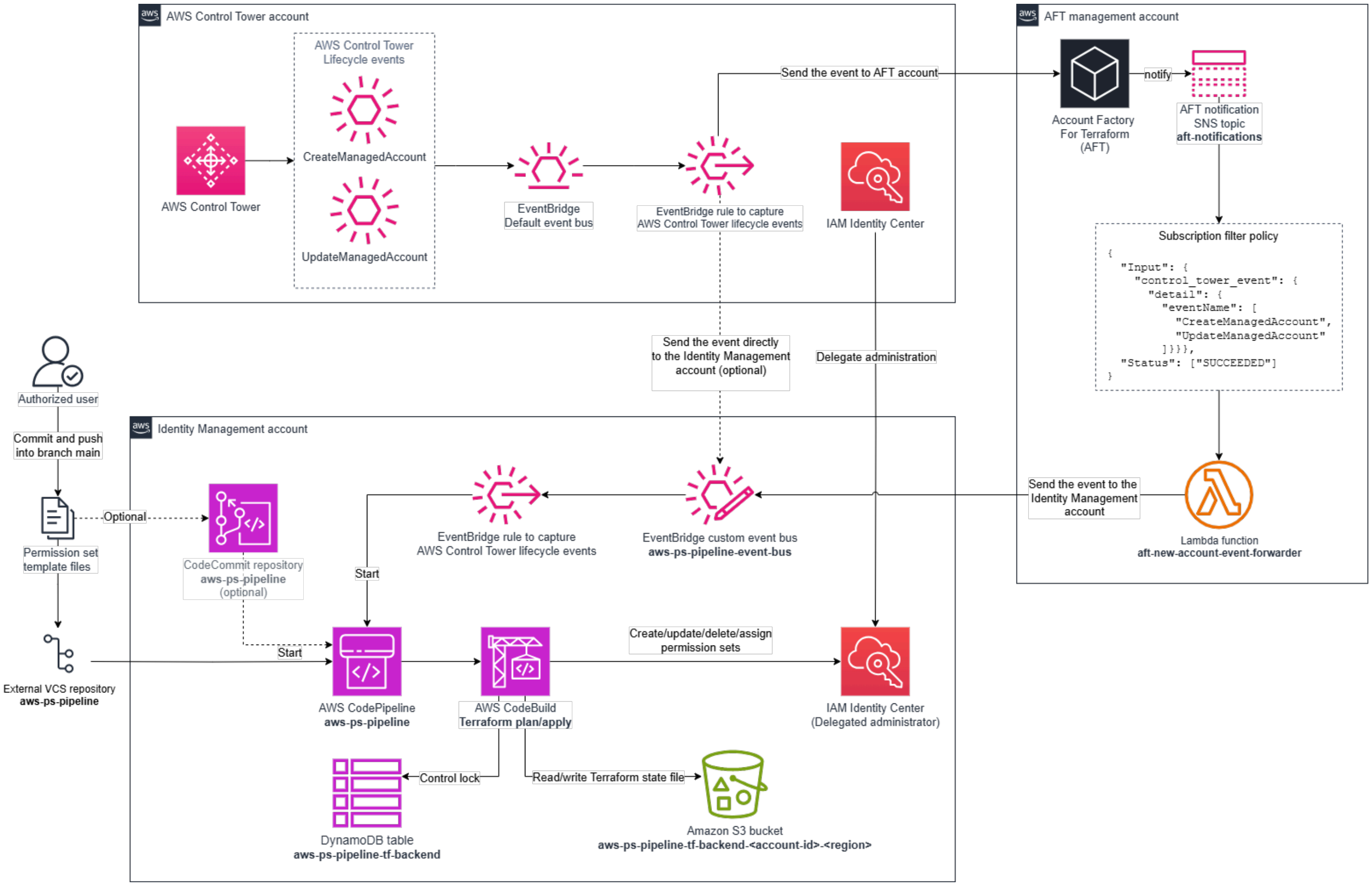
- An IAM Identity Center delegated administrator AWS account to receive the solution. For more information, see Delegated administration in the IAM Identity Center documentation.
- A version control system (VCS) repository to handle the main code. For a sample, see the solution's GitHub repository.
- Necessary AWS resources for the Terraform backend management, such as an Amazon Simple Storage Service (Amazon S3) bucket and Amazon DynamoDB table.

Limitations

- The pipeline uses AWS native resources and open source Terraform. The pipeline is not prepared to make calls to third-party ecosystems.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

The following diagram shows the components and workflow for this pattern.



AWS Control Tower events flow

The solution begins with the integration of events coming from either AWS Control Tower or AFT. The choice between one or the other service is made at the implementation time through variable definition. Regardless of the method used, the pipeline is triggered whenever an account is created or updated. The pipeline reconciles the policies stored in the permission sets management repository.

Following are the AWS Control Tower lifecycle events:

- `CreateManagedAccount` – When a new account is created
- `UpdateManagedAccount` – When an existing account is updated

Event routing

EventBridge serves as the central event processing service, capturing events generated in the AWS Control Tower account. When events occur, EventBridge intelligently routes them to a centralized event bus in the solution account. AWS Control Tower lifecycle events follow distinct routing patterns. If AFT is defined as the event source, the AFT management account handles the events instead of the AWS Control Tower account. This event-driven architecture enables automated responses to organizational changes without manual intervention.

AFT integration process

When AWS Control Tower lifecycle events reach the AFT management account, they automatically trigger multiple downstream processes that are intrinsic to AFT. After the AFT account customization workflow completes, it publishes a message to the dedicated `aft-notifications` Amazon Simple Notification Service (Amazon SNS) topic. That topic triggers the `aft-new-account-forward-event` AWS Lambda function that's implemented by this solution. The Lambda function sends the event to the solution account event bus, where it's used to start the pipeline.

Infrastructure as code pipeline

The solution pipeline operates as a fully automated deployment mechanism. The AWS CodePipeline service continuously monitors the repository for changes. Upon detecting new commits, it automatically initiates the deployment workflow and initiates a sequential process that includes validation and execution phases. The system runs Terraform `plan` operations to identify proposed changes, followed by Terraform `apply` commands to implement those changes in the AWS environment. Notably, the pipeline runs without any manual approval gates. This approach enables rapid deployment of infrastructure changes while maintaining auditability through pipeline logs and Terraform state files.

The pipeline leverages AWS CodeBuild to run Terraform operations in a controlled environment with appropriate permissions. Through this IaC approach, the pipeline can perform comprehensive permission management operations including:

- Create new permission sets.
- Update existing permission sets.
- Remove unnecessary permission sets.
- Manage the assignment of these permissions across accounts and groups within the AWS organizations.

To maintain infrastructure consistency and prevent conflicting changes, the solution implements the Terraform backend state management system using an Amazon S3 bucket and dedicated

Amazon DynamoDB table. This approach provides persistent storage location for Terraform state files and state locking mechanisms to prevent concurrent modifications to the same resources.

The main Terraform code uses the official AWS `permission-sets` Terraform module. This module can dynamically manage permission sets in IAM Identity Center, based on permission set templates.

Source control management

The permission set templates (JSON files) reside in an external version control system, such as GitHub, that provides a centralized repository for identity management configurations. This approach establishes a single source of truth for permission set definitions, while enabling collaborative development through standard code review practices. Authorized users can commit changes to these templates following organizational change management processes. These commits serve as the primary trigger for the automated deployment pipeline, initiating the infrastructure update process.

For an example of how to configure the permission sets using the JSON file in the repository, see Additional information.

Tools

AWS services

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeConnections enables AWS resources and services, such as CodePipeline, to connect to external code repositories, such as GitHub.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command line shell.
- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.

- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to all of your AWS accounts and cloud applications.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses. It enables push notifications for account management events, ensuring that relevant parties are informed of important changes or actions within the system.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the AWS Samples organization on GitHub in the sample-terraform-aws-permission-sets-pipeline repository.

Best practices

- Always pin the versions of the Terraform modules and providers used to run code in production.
- Use a static code analysis tool, such as Checkov, to scan your code and then solve the security issues.

- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see [Grant least privilege] and [Security best practices] in the IAM documentation.

Epics

Create the prerequisites (optional)

Task	Description	Skills required

Create Terraform backend resources.	If you haven't created your Terraform backend AWS resources yet, use the following steps to create an Amazon S3 bucket (`s3-tf-backend-{ACCOUNT_ID}`) and a DynamoDB table (`ddb-tf-backend`).
	
	1. Sign in to the AWS account where you will deploy the solution, and open AWS CloudShell in the AWS Control Tower home AWS Region.
	2. Run the following commands, replacing the placeholder `{ACCOUNT_ID}` with your AWS account ID:
	
	`aws s3api create-bucket --bucket s3-tf-backend-{ACCOUNT_ID}`
	`aws s3api put-bucket-versioning --bucket s3-tf-backend-{ACC`
		AWS administrator

Task	Description	Skills required
	```OUNT_ID} --versioning-configuration Status=Enabled aws dynamodb create-table --table-name ddb-tf-backend --attribute-definitions AttributeName=LockID,AttributeType=S --key-schema AttributeName=LockID,KeyType=HASH --provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1```	

Task	Description	Skills required
Create a cross-account role.	You must provide a cross-account IAM role in the `event-source-account` Terraform AWS provider configuration. If you haven't created this role yet, use the following steps to create it: 1. Sign in to the AWS account that will be your event source (AWS Control Tower management account or AFT account), and open AWS CloudShell. 2. Run the following command, replacing the placeholder `{ACCOUNT_ID}` with the AWS account ID that you're using for this solution, not your current AWS account ID: `aws iam create-role \ --role-name CrossAccountRole \ --assume-role-policy-document '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": {`	AWS administrator

Task	Description	Skills required

	``
	 "AWS": "arn:aws:iam::{ACCOUNT_ID}:root"
	                 },
	
	 "Action": "sts:AssumeRole"
	             }
	         ]
	     }'
	``
	1. Check the command output and copy the role Amazon Resource Name (ARN) (for example, `arn:aws:iam::111122223333:role/CrossAccountRole`).
	2. To attach an IAM policy to the role, run the following command:
	``
	aws iam attach-role-policy \\
	    --role-name CrossAccountRole \\
	    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess
	``
	This example uses the AWS managed IAM policy [AdministratorAccess](). If you prefer, you can use a more specific policy.	

Prepare the permission set repository

Task	Description	Skills required
Create a dedicated repository.	This task assumes that you're using GitHub. Create a dedicated repository to store the main Terraform code and the permission set template JSON files.	DevOps engineer
Prepare the permission set code.	For information about how you can structure the following files, see the sample code on the solution repository: ├── main.tf ├── outputs.tf ├── providers.jinja └── templates Copy the content, keep the `providers.jinja` values, and make the necessary adjustments to the other files. For example, add permission set template files into `templates` or pin the `aws-ia/permission-sets/aws` module version in the `main.tf` file.	DevOps engineer
Commit your changes.	Commit and push the changes to the repository that you created earlier. Save	DevOps engineer

Task	Description	Skills required
	the repository name and its GitHub organization, for example, `myorg/aws-ps-pipeline` .	

Prepare the deployment code

Task	Description	Skills required
Download the content.	Download (clone) the content from the solution [repository]().	DevOps engineer
Fulfill the variables.	Create a `terraform .tfvars` file and add the following necessary variables: • `repository_name` – Name of permission set repository that you created earlier • `branch_name` – Name of the permission set repository branch • `vcs_provider` – Your VCS provider • `account_lifecycle_events_source` – The source of events to trigger the pipeline (AWS Control Tower or AFT) `repository_name = "myorg/aws-ps-pipeline"`	DevOps engineer

Task	Description	Skills required

	<pre>branch_name
	         = "main"
	vcs_provider
	          = "github"
	account_lifecycle_
	events_source = "CT"</pre> For information about additional variable options, see the variables.tf file in this pattern's GitHub repository.	

Task	Description	Skills required

Adjust the Terraform backend configuration.	In the `backend.tf` file, replace the placeholders with your own values. Use the AWS Control Tower home AWS Region, and provide the names of the previously created Amazon S3 bucket and DynamoDB table.
	<pre>terraform {
	  required_version =
	 ">=1.6"
	  backend "s3" {
	    region =
	 "{region}"
	    bucket =
	 "{bucket_name}"
	    key =
	 "terraform.tfstate"
	    dynamodb_table =
	 "{table_name}"
	    encrypt =
	 "true"
	  }
	}</pre>If you prefer, you can use your own Terraform backend configuration.
		DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Adjust the Terraform provider configuration.</td><td>

In the `providers.tf` file, replace the placeholders with your own information. Use the AWS Control Tower home Region, and provide the ARN of the previously created cross-account IAM role for the `event-source-account` provider.

```
provider "aws" {
  region = "{region}"
}

provider "aws" {
  alias  = "event-source-account"
  region = "{region}"
  assume_role {
    role_arn = "{role_arn}"
  }
}
```

</td><td>DevOps engineer</td></tr>
</table>

Deploy the solution manually

Task	Description	Skills required
Select the AWS account.	We recommend that you deploy the solution in the IAM Identity Center delegated administrator account. However, you can also deploy it in the AWS Organizations management account.	AWS administrator

Task	Description	Skills required
	To sign in to the selected account in the same Region as the IAM Identity Center instance, use the AWS CLI. Make sure that the IAM role you're using has permission to assume the role that's specified for the `event-source-account` provider in the previous steps. Also, this role must have access to the AWS resources that are used in the Terraform backend configuration.	
Run Terraform manually.	To initialize, plan and apply the configurations, run the following Terraform commands in the order shown: 1. `terraform init` 2. `terraform plan` 3. `terraform apply`	DevOps engineer
Check the deployment results.	In the IAM Identity Center delegated administrator account, check that the `aws-ps-pipeline` pipeline has been created. Also check that there is a AWS CodeConnections connection with **Pending** status.	AWS DevOps

Task	Description	Skills required
Finish the CodeConnections configuration.	To finish the CodeConnections configuration, use the following steps: 1. Go to CodeConnections, select the connection that was created, and choose **Update pending connection**. 2. Provide your GitHub credentials, install a new GitHub App or select an existing one that has access to the permission set repository, and choose **Connect**. The pipeline should now have access to the permission set repository. For detailed instructions, see [Update a pending connection]() in the Developer Tools console documentation.	AWS DevOps

Choose a pipeline execution flow to test the solution

Task	Description	Skills required
Run the pipeline by AWS Control Tower or AFT updates.	After an account is created or changed by using AWS Control Tower or AFT (depending on the type of	AWS administrator

Task	Description	Skills required
	lifecycle events that you chose), the pipeline starts.	
Run the pipeline by changing the code.	After you change the code and commit it to the `main` branch, the pipeline starts.	AWS DevOps
Run the pipeline manually.	To start the pipeline manually, use the Release change feature in AWS CodePipeline.	AWS DevOps

Troubleshooting

Issue	Solution
Access denied	Verify that you have the permissions required to deploy the solution.
CodeConnections issues	• Check that the connection status is **Available** rather than **Pending**. • Verify that the CodeConnections configura tion is properly completed with GitHub credentials. For more information, see Update a pending connection in the Developer Tools console documentation. • Make sure that the GitHub app has proper access permissions to the permission set repository.
Pipeline execution problems	• Check Amazon CloudWatch logs for pipeline execution errors. • Make sure that the IAM Identity Center delegation is set up correctly. Otherwise,

Issue	Solution
	it can cause pipeline failures when reading AWS account metadata.
Permission sets deployment issues	• Validate that the JSON syntax in the permission set template files is correct. • Verify that the IAM managed policies referenced in templates exist. • Check if the specified groups and users in assignments exist in IAM Identity Center.

Related resources

AWS service documentation

- AWS IAM Identity Center User Guide
- Manage AWS accounts with permission sets (IAM Identity Center documentation)

Other resources

- AWS Permission Sets module (Terraform)

Additional information

JSON file with sample permission set

The following example shows how to configure a permission set by using the JSON file in the repository:

```
{
  "Name": "ps-billing", // Permission set identifier
  "Comment": "Sample permission set for billing access", // Comment to document the
 purpose of the permission set
  "Description": "Billing access in AWS", // Detailed description
  "SessionDuration": "PT4H", // Session duration = 4 hours (ISO 8601 format)
  "ManagedPolicies": [ // List of AWS IAM managed policies
    "arn:aws:iam::aws:policy/job-function/Billing",
```

```
    "arn:aws:iam::aws:policy/job-function/SupportUser",
    "arn:aws:iam::aws:policy/AWSSupportAccess",
    "arn:aws:iam::aws:policy/job-function/ViewOnlyAccess"
  ],
  "CustomerPolicies": [], // References to IAM policies previously created
  "CustomPolicy": {}, // Inline IAM policy defined directly in the permission set
  "PermissionBoundary": {  // AWS or customer managed IAM policy to be used as boundary
    "ManagedPolicy": "",
    "CustomerPolicy": ""
  },
  "Assignments": [ // Define the assignment rules
    {
      "all_accounts": true, // Apply to ALL active AWS accounts in organization
      "principal": "G_BILLING_USERS", // Group/user name in Identity Center
      "type": "GROUP", // Can be "GROUP" or "USER"
      "account_id": [], // List of AWS account ID (empty since all_accounts=true)
      "account_ou": [], // List of AWS Organizational Unit IDs with target AWS accounts
      "account_tag": [] // List of tags (key:value) to match AWS Organization accounts
 tags
    }
  ]
}
```

For more information, see the JSON schema in the AWS Permission Sets module documentation on the Terraform website.

Tips

- You can use Terraform import blocks to import an existing permission set to the solution.
- You can use AFT to implement the AWS permission set pipeline in a delegated account. For more information, see AFT Blueprints.

Tag Transit Gateway attachments automatically using AWS Organizations

Richard Milner-Watts, Haris Bin Ayub, and John Capps, Amazon Web Services

Summary

On Amazon Web Services (AWS), you can use AWS Resource Access Manager to share AWS Transit Gateway across AWS account boundaries. When you create Transit Gateway attachments across account boundaries, however, the attachments are created without a Name tag. That can make identifying attachments time consuming.

This solution provides an automated mechanism to gather information about each Transit Gateway attachment for accounts within an organization that is managed by AWS Organizations. The process includes looking up the Classless Inter-Domain Routing (CIDR) range from the Transit Gateway route table. The solution then applies a Name tag in the form of `<CIDR-range>-<AccountName>` to the attachment within the account that holds the transit gateway.

This solution can be used alongside a solution such as the Serverless Transit Network Orchestrator from the AWS Solutions Library. Serverless Transit Network Orchestrator enables the automated creation of Transit Gateway Gateway attachments at scale.

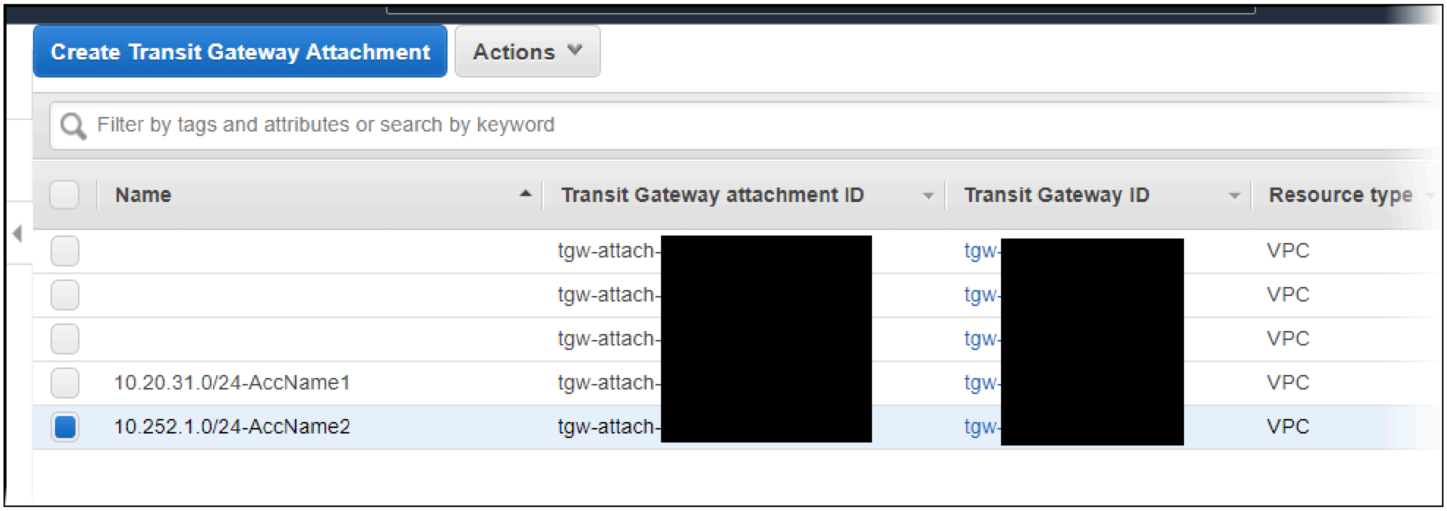
Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS Organizations organization that contains all related accounts
- Access to the organization management account, under the organization's root, to create the required AWS Identity and Access Management (IAM) role
- A Shared Networking member account containing one or more transit gateways that are shared with the organization and have attachments

Architecture

The following screenshot of the AWS Management Console shows examples of Transit Gateway attachments with no associated Name tag and two Transit Gateway attachments with Name tags generated by this solution. The structure of the generated Name tag is `<CIDR-range>-<AccountName>`.



This solution uses AWS CloudFormation to deploy an AWS Step Functions workflow that manages the creation of Transit Gateway Name tags across all configured AWS Regions. The workflow invokes AWS Lambda functions, which perform the underlying tasks.

After the solution has obtained the account names from AWS Organizations, the Step Functions state machine gets all Transit Gateway attachment IDs. These are processed in parallel by Region. This processing includes looking up the CIDR range for each attachment. The CIDR range is obtained by searching the Transit Gateway route tables within the Region for a matching Transit Gateway attachment ID. If all the required information is available, the solution applies a Name tag to the attachment. The solution will not overwrite any existing Name tags.

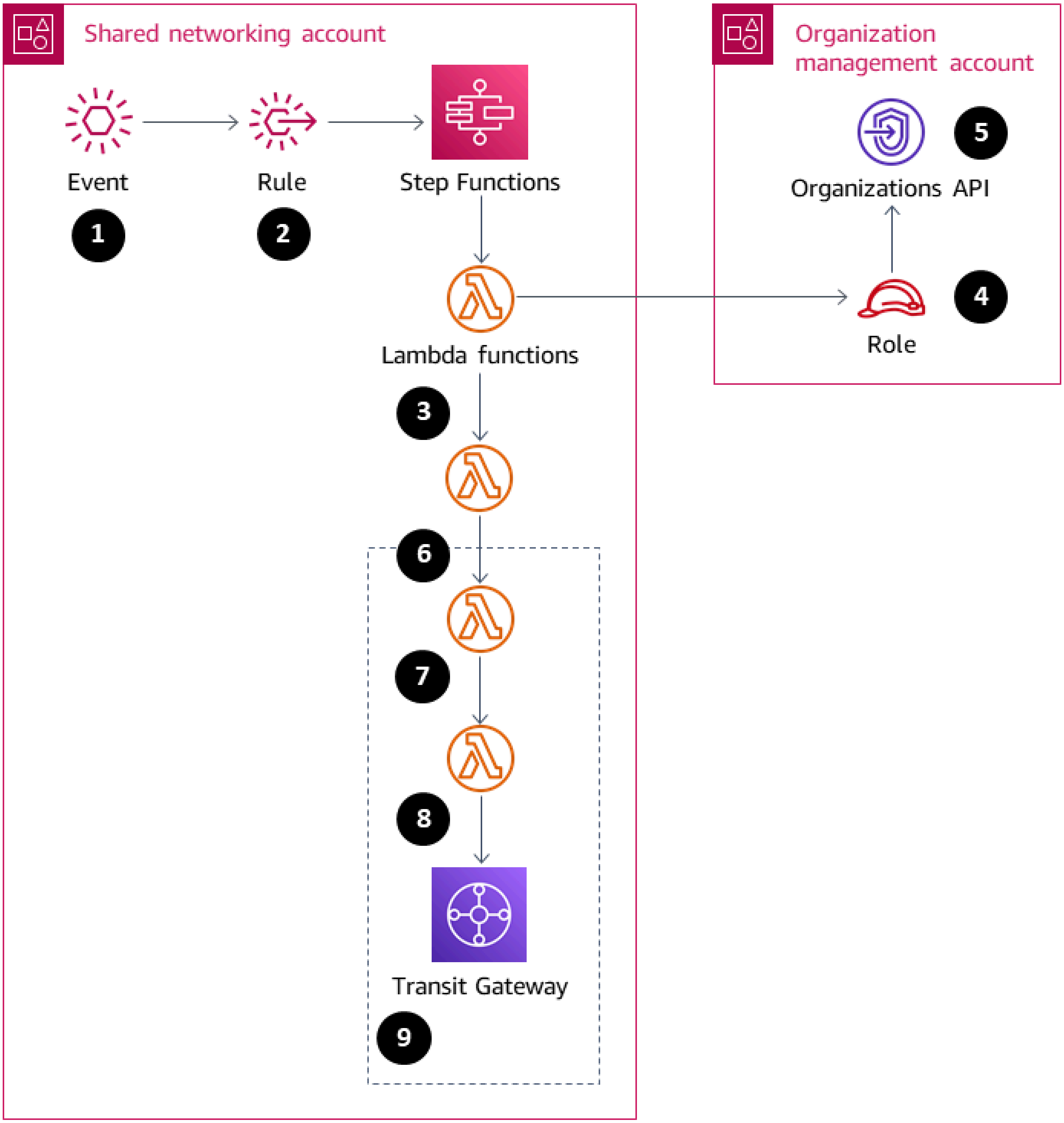
The solution runs on a schedule controlled by an Amazon EventBridge event. The event initiates the solution each day at 6:00 AM UTC.

Target technology stack

- Amazon EventBridge
- AWS Lambda
- AWS Organizations
- AWS Transit Gateway
- Amazon Virtual Private Cloud (Amazon VPC)
- AWS X-Ray

Target architecture

The solution architecture and workflow are shown in the following diagram.



1. The scheduled event initiates the rule.

2. The EventBridge rule starts the Step Functions state machine.
3. The state machine invokes the `tgw-tagger-organizations-account-query` Lambda function.
4. The `tgw-tagger-organizations-account-query` Lambda function assumes the role in the organization management account.
5. The `tgw-tagger-organizations-account-query` Lambda function calls the Organizations API to return AWS account metadata.
6. The state machine invokes the `tgw-tagger-attachment-query` Lambda function.
7. For each Region, in parallel, the state machine invokes `tgw-tagger-rtb-query` Lambda function to read the CIDR range for each attachment.
8. For each Region, in parallel, the state machine invokes `tgw-tagger-attachment-tagger` Lambda function.
9. Name tags are created for Transit Gateway attachments in the Shared Networking account.

Automation and scale

The solution processes each Region in parallel to reduce the total duration of the run.

Tools

AWS services

- [AWS CloudFormation](#) provides a way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code.
- [Amazon CloudWatch](#) helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- [Amazon EventBridge](#) is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge receives an event, an indicator of a change in environment, and applies a rule to route the event to a target. Rules match events to targets based on either the structure of the event, called an event pattern, or on a schedule.
- [AWS Lambda](#) is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests each day to thousands each second. You pay only for the compute time that you consume. There is no charge when your code is not running.

- AWS Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources. Using Organizations, you can programmatically create new AWS accounts and allocate resources, group accounts to organize your workflows, apply policies to accounts or groups for governance, and simplify billing by using a single payment method for all of your accounts.
- AWS Step Functions is a low-code visual workflow service used to orchestrate AWS services, automate business processes, and build serverless applications. Workflows manage failures, retries, parallelization, service integrations, and observability so developers can focus on higher-value business logic.
- AWS Transit Gateway connects VPCs and on-premises networks through a central hub. This simplifies your network and puts an end to complex peering relationships. It acts as a cloud router, so that each new connection is made only one time.
- Amazon Virtual Private Cloud (Amazon VPC) is a service for launching AWS resources in a logically isolated virtual network that you define.
- AWS X-Ray collects data about requests that your application serves, and provides tools that you can use to view, filter, and gain insights into that data to identify issues and opportunities for optimization.

Code

The source code for this solution is available in the Transit Gateway Attachment Tagger GitHub repository. The repository includes the following files:

- `tgw-attachment-tagger-main-stack.yaml` creates all the resources to support this solution within the Shared Networking account.
- `tgw-attachment-tagger-organizations-stack.yaml` creates a role in the management account of the organization.

Epics

Deploy the main solution stack

Task	Description	Skills required
Gather required prerequisite information.	To configure cross-acc ount access from the	DevOps engineer

Task	Description	Skills required
	Lambda function to the AWS Organizations API, you need the account ID for the organization's management account. **Note** The order in which the two AWS CloudFormation stacks are created matters. You must deploy resources into the Shared Networking account first. The role in the Shared Networking account must already exist before deploying resources into the organization's management account. For more information, see the [AWS documentation]().	

Task	Description	Skills required
Launch the AWS CloudFormation template for the main solution stack.	The template for the main solution stack will deploy the IAM roles, Step Functions workflow, Lambda functions, and the Amazon CloudWatch event. Open the AWS Management Console for the Shared Networking account, and then open the :&CFN console. Create the stack by using the `tgw-attachment-tagger-main-stack.yaml` template and the following values: • **Stack name – tgw-attachment-tagger-main-stack** • **awsOrganizationsRootAccountId** – Account ID for the organization's management account • **TGWRegions** parameter – AWS Regions for the solution, entered as a comma-delimited string • **TGWList** parameter – Transit gateway IDs to be excluded from the solution, entered in a comma-delimited string	DevOps engineer

Task	Description	Skills required
	For more information about launching a AWS CloudFormation stack, see the AWS documentation.	
Verify that the solution has launched successfully.	Wait for the CloudFormation stack to reach a status of **CREATE_COMPLETE**. This should take less than one minute. Open the Step Functions console, and verify that a new state machine has been created with the name **tgw-attachment-tagger-state-machine**.	DevOps engineer

Deploy the AWS Organizations stack

Task	Description	Skills required
Gather required prerequisite information.	To configure cross-account access from the Lambda function to the AWS Organizations API, you need the account ID for the Shared Networking account.	DevOps engineer
Launch the CloudFormation template for the Organizations stack	The template for the AWS Organizations stack will deploy the IAM role in the organization's management account.	DevOps engineer

Task	Description	Skills required
	Access the AWS console for the organization's management account. and then open the CloudFormation console. Create the stack by using the `tgw-attachment-tagger-organizations-stack.yaml` template and the following values: • **Stack name – tgw-attachment-tagger-organizations-stack** • **NetworkingAccountId** parameter – Account ID for the Shared Networking account For the other stack creation options, use the defaults.	
Verify that the solution has launched successfully.	Wait for the AWS CloudFormation stack to reach a status of **CREATE_COMPLETE**. This should take less than one minute. Open the AWS Identity and Access Management (IAM) console, and verify that a new role has been created with the name **tgw-attachment-tagger-organization-query-role**.	DevOps engineer

Verify the solution

Task	Description	Skills required
Run the state machine.	Open the Step Functions console for the Shared Networking account, and choose **State machines** in the navigation pane. Select the state machine **tgw-attachment-tagger-state-machine**, and choose **Start Execution**. Because the input to this state machine is not used by the solution, you can use the default value. `{ "Comment": "Insert your JSON here" }` Choose **Start Execution**.	DevOps engineer
Watch the state machine until completion.	On the new page that opens, you can watch the state machine run. The duration will depend on the number of Transit Gateway attachments to process. On this page, you can examine each step of the state machine. You can view the various tasks within the	DevOps engineer

Task	Description	Skills required
	state machine and follow links to the CloudWatch logs for the Lambda functions. For the tasks that run in parallel within the map, you can use the **Index** dropdown list to view the specific implement ations for each Region.	
Verify the Transit Gateway attachment tags.	Open the VPC console for the Shared Networking account, and choose **Transit Gateway Attachments**. On the console, a Name tag is provided for attachments that met the criteria (the attachment is propagated to a Transit Gateway route table, and the resource owner is a member of the organization).	DevOps engineer
Verify the CloudWatch event initiation.	Wait for the CloudWatc h event to initiate. This is scheduled for 06:00 UTC. Then open the Step Functions console for the Shared Networking account, and choose **State machines** in the navigation pane. Select the state machine **tgw-attachment-tagger-state -machine**. Verify that the solution ran at 06:00 UTC.	DevOps engineer

Related resources

- AWS Organizations
- AWS Resource Access Manager
- Serverless Transit Network Orchestrator
- Creating IAM roles
- Creating a stack on the AWS CloudFormation console

More patterns

- Govern permission sets for multiple accounts by using Account Factory for Terraform
- Provision AWS Service Catalog products based on AWS CloudFormation templates by using GitHub Actions
- Provision least-privilege IAM roles by deploying a role vending machine solution

AI & machine learning

Topics

- Aggregate data in Amazon DynamoDB for ML forecasting in Athena
- Associate an AWS CodeCommit repository in one AWS account with Amazon SageMaker AI Studio Classic in another account
- Automatically extract content from PDF files using Amazon Textract
- Build a cold start forecasting model by using DeepAR for time series in Amazon SageMaker AI Studio Lab
- Build an MLOps workflow by using Amazon SageMaker AI and Azure DevOps
- Configure model invocation logging in Amazon Bedrock by using AWS CloudFormation
- Create a custom Docker container image for SageMaker and use it for model training in AWS Step Functions
- Use Amazon Bedrock agents to automate creation of access entry controls in Amazon EKS through text-based prompts
- Deploy a RAG use case on AWS by using Terraform and Amazon Bedrock
- Deploy preprocessing logic into an ML model in a single endpoint using an inference pipeline in Amazon SageMaker
- Develop advanced generative AI chat-based assistants by using RAG and ReAct prompting
- Develop a fully automated chat-based assistant by using Amazon Bedrock agents and knowledge bases
- Document institutional knowledge from voice inputs by using Amazon Bedrock and Amazon Transcribe
- Generate personalized and re-ranked recommendations using Amazon Personalize
- Streamline machine learning workflows from local development to scalable experiments by using SageMaker AI and Hydra
- Train and deploy a custom GPU-supported ML model on Amazon SageMaker
- Translate natural language into query DSL for OpenSearch and Elasticsearch queries
- Use Amazon Q Developer as a coding assistant to increase your productivity
- Use SageMaker Processing for distributed feature engineering of terabyte-scale ML datasets
- Visualize AI/ML model results using Flask and AWS Elastic Beanstalk

- More patterns

Aggregate data in Amazon DynamoDB for ML forecasting in Athena

Sachin Doshi and Peter Molnar, Amazon Web Services

Summary

This pattern shows you how to build complex aggregations of Internet of Things (IoT) data in an Amazon DynamoDB table by using Amazon Athena. You also learn how to enrich the data with machine learning (ML) inference by using Amazon SageMaker AI and how to query geospatial data by using Athena. You can use this pattern as the basis for creating an ML forecasting solution that meets your organization's requirements.

For demonstration purposes, this pattern uses an example scenario of a business that's operating a scooter rideshare and wants to predict the optimal number of scooters that must be deployed for customers in different urban neighborhoods. The business uses a pre-trained ML model that predicts customer demand for the next hour based on the past four hours. The scenario uses a public dataset from the Office of Civic Innovation & Technology for the Louisville Metro government. The resources for this scenario are available in a GitHub repository.

Prerequisites and limitations

- An active AWS account
- Permissions to create an AWS CloudFormation stack with AWS Identity and Access Management (IAM) roles for the following:
  - Amazon Simple Storage Service (Amazon S3) bucket
  - Athena
  - DynamoDB
  - SageMaker AI
  - AWS Lambda

Architecture

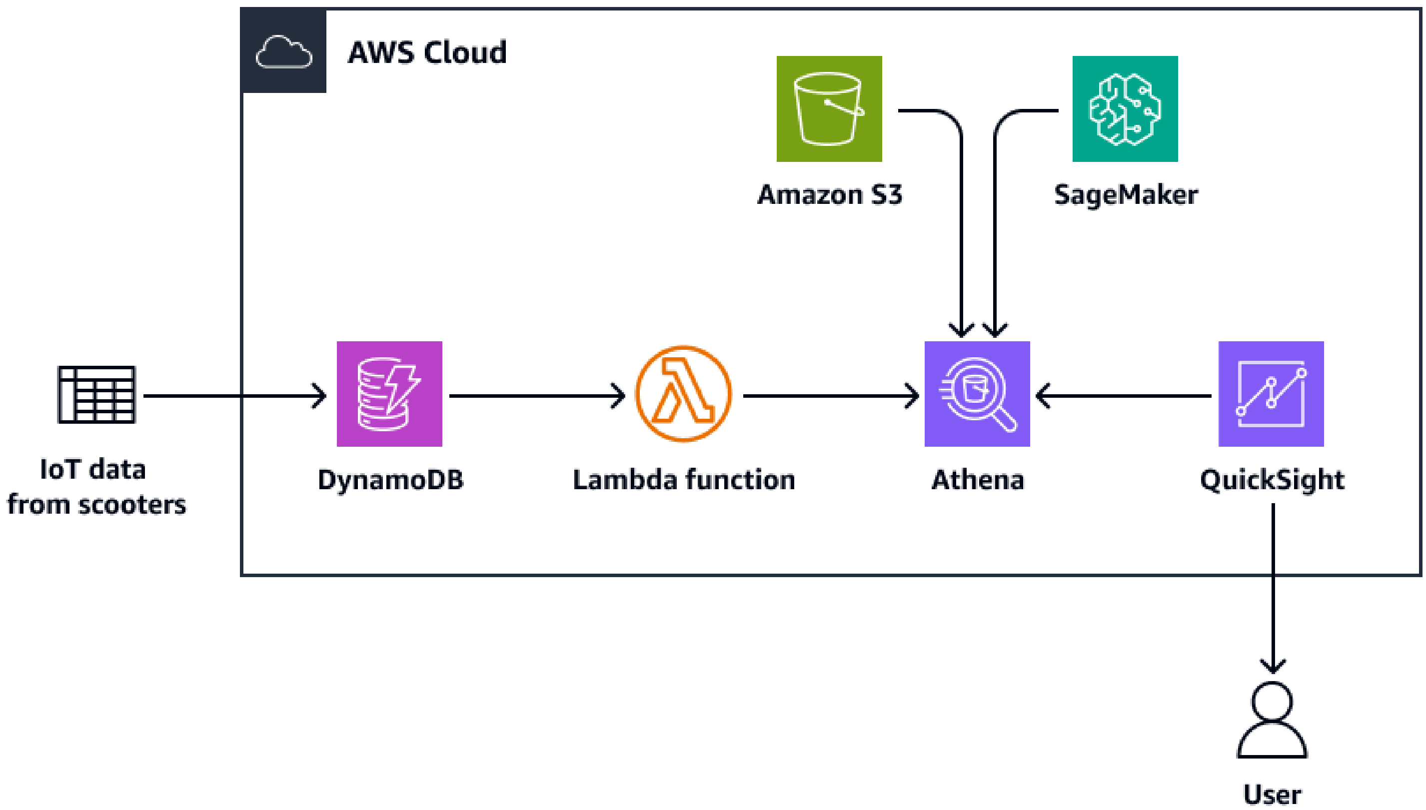
Technology stack

- Amazon QuickSight

- Amazon S3
- Athena
- DynamoDB
- Lambda
- SageMaker AI

Target architecture

The following diagram shows an architecture for building complex aggregations of data in DynamoDB by using the querying capabilities of Athena, a Lambda function, Amazon S3 storage, a SageMaker AI endpoint, and a QuickSight dashboard.



The diagram shows the following workflow:

1. A DynamoDB table ingests IoT data that's transmitted from a fleet of scooters.
2. A Lambda function loads the DynamoDB table with the ingested data.
3. An Athena query creates a new DynamoDB table for the geospatial data that represents the urban neighborhoods.

4. The query location is saved in an S3 bucket.
5. An Athena function queries the ML inference from the SageMaker AI endpoint that hosts the pre-trained ML model.
6. Athena queries data directly from the DynamoDB tables and aggregates the data for analysis.
7. A user views the output of the analyzed data in a QuickSight dashboard.

Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- Amazon SageMaker AI is a managed ML service that helps you build and train ML models and then deploy them into a production-ready hosted environment.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data in a single dashboard.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Code repository

The code for this pattern is available in the GitHub Use ML predictions over Amazon DynamoDB data with Amazon Athena ML repository. You can use the CloudFormation template from the repository to create the following resources used in the example scenario:

- A DynamoDB table
- A Lambda function to load the table with relevant data
- A SageMaker AI endpoint for inference requests, with the pre-trained XGBoost model that's stored in Amazon S3
- An Athena workgroup named `V2EngineWorkGroup`

- Named Athena queries to look up the geospatial shapefiles and predict scooter demand
- A prebuilt Amazon Athena DynamoDB connector that enables Athena to communicate with DynamoDB and uses AWS Serverless Application Model (AWS SAM) to build the application in reference to the DynamoDB connector

Epics

Get the example dataset

Task	Description	Skills required
Download the dataset and resources.	1. Download a public dataset of dockless vehicle rentals. For demo purposes, this data is prepopulated in DynamoDB as part of the use case, but in a production environment you send this data to DynamoDB through various mechanisms such as IoT devices or Amazon Kinesis consumers. These mechanisms use Lambda to insert data into DynamoDB. 2. Download the GIS shapefiles that represent the boundaries of historical and cultural neighborhoods within the city of Louisville, KY. The public dataset is provided by the Louisville and Jefferson County, KY Information Consortium. The original	App developer, Data scientist

Task	Description	Skills required
	shapefiles are already converted into a text file that you can query with Athena, but you can find the Python code for transforming shapefiles in the Jupyter notebook at [Geo-Spatial processing of GIS shapefiles with Amazon Athena](#) in GitHub. 3. Download the pretrained [Python code](#) that trains the ML model for hourly forecasts by using SageMaker AI and Athena. 4. Get the SQL query in Athena that brings everything together for live predictions from the data stored in DynamoDB. 5. (Optionally) Use QuickSight to [visualize geospatial data over a map of Louisville, Kentucky](#).	

Use a CloudFormation template to deploy the required resources

Task	Description	Skills required
Create a CloudFormation stack.	1. Download the CloudFormation template from the GitHub [repository](#).	AWS DevOps

Task	Description	Skills required
	2. **Note** Sign in to the AWS Management Console, and then choose `us-east-1` . : The ML model is stored in the Amazon Elastic Container Registry (Amazon ECR) for the `us-east-1` AWS Region, but the pattern is Region agnostic. You can replicate the pattern in any Region where the AWS services used in this pattern are supported. 3. Open the [CloudFormation console](), and then choose **Stacks** on the navigation pane. 4. Choose **Create stack**, and then choose **With existing resources (import resources)**. 5. On the **Identify resources** page, choose **Next**. 6. In the **Specify template** section, for **Template**	

Task	Description	Skills required
	source, select **Upload a template file**. 7. Choose **File**, and then choose the CloudFormation template that you downloaded earlier. 8. Choose **Next**, accept the default parameter values, and choose **Next** to step through the rest of the setup wizard. 9. Select the **I acknowledge that AWS CloudFormation might create IAM resources with custom names** check box. 10. Choose **Create stack**. **Note** It can take 15–20 minutes for the CloudFormation stack to create these resources.	

Task	Description	Skills required
Verify the CloudFormation deployment.	To verify that the sample data from the CloudFormation template is loaded into DynamoDB, do the following: 1. Open the DynamoDB console, and then choose **Tables** from the navigation pane. 2. In the **Tables** section, check for the `DynamoDBTableDocklessVehicles` table. 3. After resource creation is complete, open the Athena console, and then choose **Workgroups** from the navigation pane. 4. Choose the `V2EngineWorkGroup` workgroup, and then choose **Switch workgroup**. 5. If you get a prompt to save the query result location, choose an Amazon S3 location where you have write permissions. 6. Choose **Save**. 7. In the navigation pane, choose **Query editor**, and then select the `athena-ml-db-<your-AWS-account-number>` database.	App developer

Load geolocation files into Athena

Task	Description	Skills required
Create an Athena table with geospatial data.	To load the geolocation files into Athena, do the following: 1. Open the [Athena console](), and then choose **Query editor** from the navigation pane. 2. Choose the **Saved queries** tab. 3. Search for and select **Q1: Neighborhoods**. 4. To return to the query editor, choose the **Editor** tab. 5. Choose **Run**. This creates a table named `louisville_ky_neighborhoods` in your database. Make sure the table is created in the `athena-ml-db-<your-AWS-account-number>` database. The query creates a new table for the geospatial data that represents the urban neighborhoods. The data table is created from GIS shapefiles. The `CREATE EXTERNAL TABLE` statement defines the schema of the table and the location and	Data engineer

Task	Description	Skills required
	format of the underlying data file. For the Python code to process shapefiles and produce this table, see Geo-Spatial processing of GIS shapefiles with Amazon Athena in AWS Samples. For detailed SQL code, see create_neighborhood_table.sql on GitHub.	

Predict demand for scooters by neighborhood from the aggregated DynamoDB data

Task	Description	Skills required
Declare a function in Athena to query SageMaker AI.	1. Open the Athena console, choose **Query editor** from the navigation pane, and then choose the **Editor** tab. 2. Copy and paste the following SQL statement into the query editor. `USING EXTERNAL FUNCTION predict_demand ( location_id BIGINT, hr BIGINT , dow BIGINT, n_pickup_1 BIGINT, n_pickup_2 BIGINT, n_pickup_3 BIGINT, n_pickup_4 BIGINT,`	Data scientist, Data engineer

Task	Description	Skills required
	``` n_dropoff_1 BIGINT, n_dropoff_2 BIGINT, n_dropoff_3 BIGINT, n_dropoff_4 BIGINT) RETURNS DOUBLE SAGEMAKER '<Your SageMaker endpoint>' ``` The first part of the SQL statement declares the external function to query ML inferences from the SageMaker AI endpoint that hosts the pre-trained model. 3. Define the order and type of the input parameters and the type of the return values. 4. Choose **Run**.	

Task	Description	Skills required
Predict demand for scooters by neighborhood from the aggregated DynamoDB data.	Now you can use Athena to query transactional data directly from DynamoDB, and then aggregate the data for analysis and forecasting. This isn't easily achieved by directly querying a DynamoDB NoSQL database. 1. Open the [Athena console,](https://console.aws.amazon.com/athena/) and then choose the **Query editor** from the navigation pane. 2. Choose the **Saved queries** tab. 3. Search for and select **Q2: DynamoDBAthenaMLScooterPredict**. 4. To return to the query editor, choose the **Editor** tab. 5. Choose **Run**. The SQL statement does the following: • Uses an [Athena Federated Query](https://docs.aws.amazon.com/athena/latest/ug/connect-to-a-data-source.html) to query the DynamoDB table with the raw trip data • Places geographic coordinates into neighborhoods by using the geospatial functions of Athena	App developer, Data scientist

Task	Description	Skills required
	• Enriches data with ML inference by using SageMaker AI For information about using SQL to aggregate DynamoDB data and SageMaker AI inference data in Athena, see [athena_long.sql](#) in GitHub.	
Verify the output.	The output table includes the neighborhood, longitude, and latitude of the centroid of the neighborhood. It also includes the number of vehicles that are predicted for the next hour. The query produces the predictions for a selected point in time. You can make predictions for any other time by changing the expression `TIMESTAMP '2019-09-07 15:00'` everywhere in the statement. If you have a real-time data feed in your DynamoDB table, change the timestamp to `NOW()`.	App developer, Data scientist

Clean up the environment

Task	Description	Skills required
Delete resources.	1. Open the Athena console and empty the bucket that you created as part of the CloudFormation stack. 2. Open the CloudFormation console, and then delete the stack named `bdb-1462-athena-dynamodb-ml-stack` . 3. Open the Amazon CloudWatch console, and then delete the log group named `/aws/sagemaker/Endpoints/Sg-athena-ml-dynamodb-model-endpoint` .	App developer, AWS DevOps

Related resources

- Amazon Athena Query Federation SDK (GitHub)
- Querying geospatial data (AWS documentation)
- Use ML predictions over Amazon DynamoDB data with Amazon Athena ML (AWS Big Data Blog)
- Amazon ElastiCache (Redis OSS) (AWS documentation)
- Amazon Neptune (AWS documentation)

Associate an AWS CodeCommit repository in one AWS account with Amazon SageMaker AI Studio Classic in another account

Laurens van der Maas and Aubrey Oosthuizen, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern provides instructions and code on how to associate an AWS CodeCommit repository in one AWS account (Account A) with Amazon SageMaker AI Studio Classic in another AWS account (Account B). To set up the association, you must create an AWS Identity and Access Management (IAM) policy and role in Account A and an IAM inline policy in Account B. Then, you use a shell script to clone the CodeCommit repository from Account A to Amazon SageMaker AI Classic in Account B.

Prerequisites and limitations

Prerequisites

- Two AWS accounts, one containing the CodeCommit repository and the other containing a SageMaker AI Domain with a user
- Provisioned SageMaker AI Domain and user, with internet access or access to CodeCommit and AWS Security Token Service (AWS STS) through virtual private network (VPC) endpoints
- A basic understanding of IAM
- A basic understanding of SageMaker AI Studio Classic
- A basic understanding of Git and CodeCommit

Limitations

This pattern applies to SageMaker AI Studio Classic only, not to RStudio on Amazon SageMaker AI.

Architecture

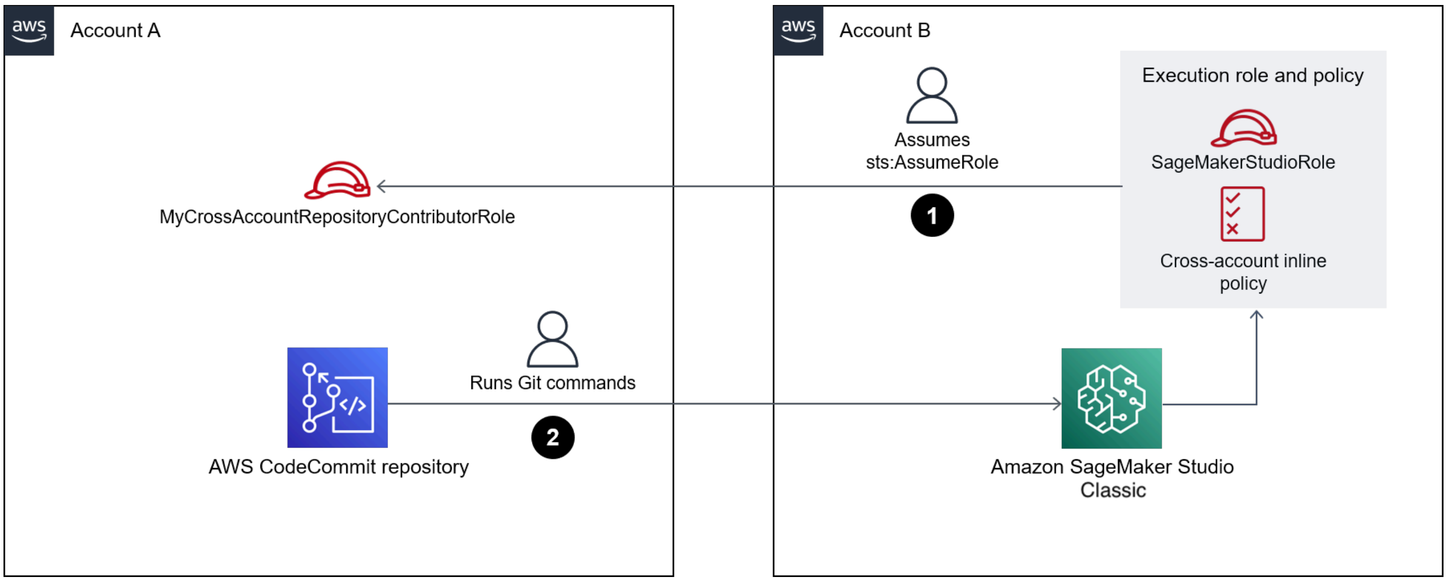
Technology stack

- Amazon SageMaker AI
- Amazon SageMaker AI Studio Classic

- AWS CodeCommit
- AWS Identity and Access Management (IAM)
- Git

Target architecture

The following diagram shows an architecture that associates a CodeCommit repository from Account A to SageMaker AI Studio Classic in Account B.



The diagram shows the following workflow:

1. A user assumes the `MyCrossAccountRepositoryContributorRole` role in Account A through the `sts:AssumeRole` role, while using the SageMaker AI execution role in SageMaker AI Studio Classic in Account B. The assumed role includes the CodeCommit permissions to clone and interact with the specified repository.
2. The user performs Git commands from the system terminal in SageMaker AI Studio Classic.

Automation and scale

This pattern consists of manual steps that can be automated by using the AWS Cloud Development Kit (AWS CDK), AWS CloudFormation, or Terraform.

Tools

AWS tools

- Amazon SageMaker AI is a managed machine learning (ML) service that helps you build and train ML models and then deploy them into a production-ready hosted environment.
- Amazon SageMaker AI Studio Classic is a web-based, integrated development environment (IDE) for machine learning that lets you build, train, debug, deploy, and monitor your machine learning models.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.

  Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Other tools

- Git is a distributed version-control system for tracking changes in source code during software development.

Epics

Create an IAM policy and IAM role in Account A

Task	Description	Skills required
Create an IAM policy for repository access in Account A.	1. Sign in to the AWS Management Console and open the IAM console. 2. In the navigation pane, choose **Policies**, and then choose **Create policy**. 3. Choose the **JSON** tab. 4. Copy the policy statement from *Example IAM policy* in the Additional information section of this pattern, and then paste the statement	AWS DevOps

Task	Description	Skills required
	into the JSON editor. Make sure to replace all placeholder values in the policy. 5. Choose **Next:Tags**, and then choose **Next:Review**. 6. For **Name**, enter a name for the policy. **Note**: In this pattern, the IAM policy is referred to as `CrossAccountAccessForMySharedDemoRepo` , but you can choose whatever policy name that you prefer. 7. Choose **Create policy**. **Tip** It's a best practice to restrict the scope of your IAM policies to the minimum required permissions for your use case.	

Task	Description	Skills required
Create an IAM role for repository access in Account A.	1. In the navigation pane of the [IAM console](https://console.aws.amazon.com/iam/), choose **Roles**, and then choose **Create role**. 2. For **Trusted entity type**, select **AWS account**. 3. In the **AWS account** section, select **Another AWS account**. 4. For **Account ID**, enter the account ID for Account B. 5. On the **Add permissions** page, search for and choose the `CrossAccountAccessForMySharedDemoRepo` policy that you created earlier. 6. Choose **Next**. 7. For **Role name**, enter a name. **Note**: In this pattern, the IAM role name is referred to as `MyCrossAccountRepositoryContributorRole`, but you can choose whatever role name that you prefer. 8. Choose **Create role**, and then copy the Amazon Resource Name (ARN) of the new role.	AWS DevOps

Create an IAM inline policy in Account B

Task	Description	Skills required
Attach an inline policy to the execution role that's attached to your SageMaker Domain user in Account B.	1. In the navigation pane of the IAM console, choose **Roles**. 2. Search for and choose the execution role that's attached to your SageMaker AI Domain user in Account B. 3. Choose **Add permissions**, and then choose **Create inline policy**. 4. Choose the **JSON** tab. 5. Copy the following policy statement, and then paste it into the JSON editor. `{ "Version": "2012-10-17", "Statement": [ { "Sid": "VisualEditor0", "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": "arn:aws:iam::<Account_A_ID>:role/<Account_A_Role_Name>" } ] }`	AWS DevOps

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::<Account_A_ID>:role/<Account_A_Role_Name>"
        }
    ]
}
```

Task	Description	Skills required
	6. Replace `<Account_A_ID>` with the account ID for Account A. Replace `<Account_A_Role_Name>` with the name of the IAM role that you created earlier. 7. Choose **Review policy**. 8. For **Name**, enter a name for your inline policy. 9. Choose **Create policy**.	

Clone the repository in SageMaker AI Studio Classic for Account B

Task	Description	Skills required
Create the shell script in SageMaker AI Studio Classic in Account B.	1. In the navigation pane of the [SageMaker console](https://console.aws.amazon.com/sagemaker/), choose **Studio**. 2. Select your user profile and then choose **Open Studio**. 3. In the **Home** section, choose **Open Launcher**. 4. In the **Utilities and files** section, choose **Text file**. 5. Copy the script from *Example SageMaker shell script* in the [Additional information](#additional-information) section of this pattern, and then paste the statement into the new file. Make sure to replace	AWS DevOps

Task	Description	Skills required
	all placeholder values in the script. 6. Right-click the **untitled.txt** tab of your new file, and then choose **Rename Text**. For **New Name**, enter **cross_account_git_clone.sh**, and then choose **Rename**.	
Invoke the shell script from the system terminal.	1. In the **Home** section of the [SageMaker console](https://console.aws.amazon.com/sagemaker/), choose **Open Launcher**. 2. In the **Utilities and files** section, choose **System terminal**. 3. In the terminal, run the following command: `chmod u+x ./cross_account_git_clone.sh && ./cross_account_git_clone.sh` You have cloned your CodeCommit repository in a SageMaker AI Studio cross-account. You can now perform all Git commands from the system terminal.	AWS DevOps

Additional information

Example IAM policy

If you use this example policy, do the following:

- Replace <CodeCommit_Repository_Region> with the AWS Region for the repository.
- Replace <Account_A_ID> with the account ID for Account A.
- Replace <CodeCommit_Repository_Name> with the name of your CodeCommit repository in Account A.

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "codecommit:BatchGet*",
            "codecommit:Create*",
            "codecommit:DeleteBranch",
            "codecommit:Get*",
            "codecommit:List*",
            "codecommit:Describe*",
            "codecommit:Put*",
            "codecommit:Post*",
            "codecommit:Merge*",
            "codecommit:Test*",
            "codecommit:Update*",
            "codecommit:GitPull",
            "codecommit:GitPush"
        ],
        "Resource": [

 "arn:aws:codecommit:<CodeCommit_Repository_Region>:<Account_A_ID>:<CodeCommit_Repository_Name>
        ]
    }
]
}
```

Example SageMaker AI shell script

If you use this example script, do the following:

- Replace <Account_A_ID> with the account ID for Account A.
- Replace <Account_A_Role_Name> with the name of the IAM role that you created earlier.

- Replace <CodeCommit_Repository_Region> with the AWS Region for the repository.
- Replace <CodeCommit_Repository_Name> with the name of your CodeCommit repository in Account A.

```
#!/usr/bin/env bash
#Launch from system terminal
pip install --quiet git-remote-codecommit

mkdir -p ~/.aws
touch ~/.aws/config

echo "[profile CrossAccountAccessProfile]
region = <CodeCommit_Repository_Region>
credential_source=EcsContainer
role_arn = arn:aws:iam::<Account_A_ID>:role/<Account_A_Role_Name>
output = json" > ~/.aws/config

echo '[credential "https://git-
codecommit.<CodeCommit_Repository_Region>.amazonaws.com"]
        helper = !aws codecommit credential-helper $@ --profile
 CrossAccountAccessProfile
        UseHttpPath = true' > ~/.gitconfig

git clone codecommit::<CodeCommit_Repository_Region>://
CrossAccountAccessProfile@<CodeCommit_Repository_Name>
```

Automatically extract content from PDF files using Amazon Textract

Tianxia Jia, Amazon Web Services

Summary

Many organizations need to extract information from PDF files that are uploaded to their business applications. For example, an organization could need to accurately extract information from tax or medical PDF files for tax analysis or medical claim processing.

On the Amazon Web Services (AWS) Cloud, Amazon Textract automatically extracts information (for example, printed text, forms, and tables) from PDF files and produces a JSON-formatted file that contains information from the original PDF file. You can use Amazon Textract in the AWS Management Console or by implementing API calls. We recommend that you use programmatic API calls to scale and automatically process large numbers of PDF files.

When Amazon Textract processes a file, it creates the following list of `Block` objects: pages, lines and words of text, forms (key-value pairs), tables and cells, and selection elements. Other object information is also included, for example, bounding boxes, confidence intervals, IDs, and relationships. Amazon Textract extracts the content information as strings. Correctly identified and transformed data values are required because they can be more easily used by your downstream applications.

This pattern describes a step-by-step workflow for using Amazon Textract to automatically extract content from PDF files and process it into a clean output. The pattern uses a template matching technique to correctly identify the required field, key name, and tables, and then applies post-processing corrections to each data type. You can use this pattern to process different types of PDF files and you can then scale and automate this workflow to process PDF files that have an identical format.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing Amazon Simple Storage Service (Amazon S3) bucket to store the PDF files after they are converted to JPEG format for processing by Amazon Textract. For more information about S3 buckets, see Buckets overview in the Amazon S3 documentation.

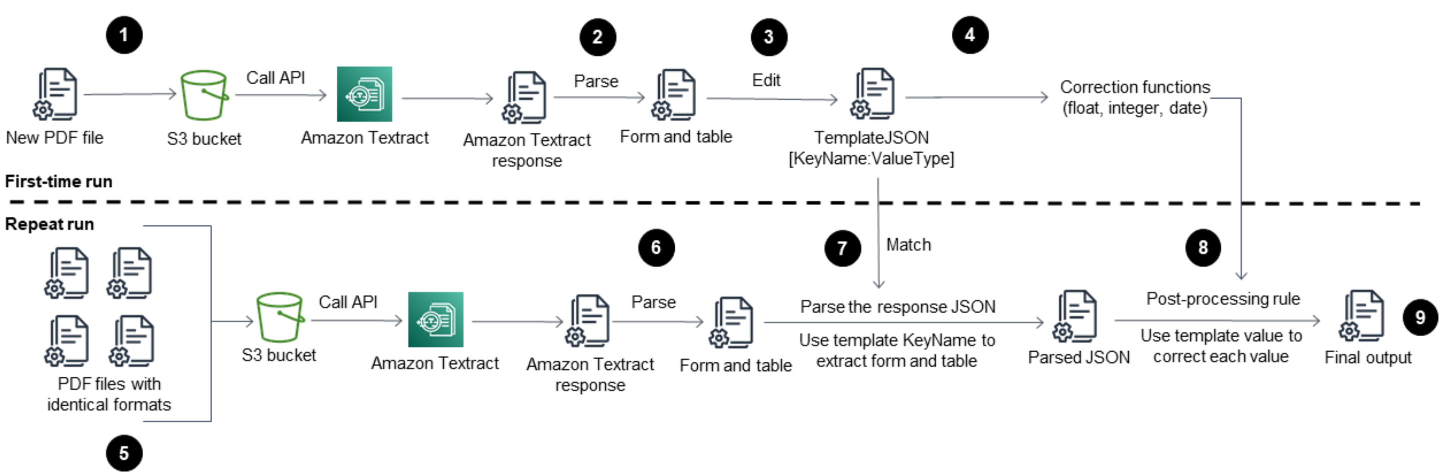
- The `Textract_PostProcessing.ipynb` Jupyter notebook (attached), installed and configured. For more information about Jupyter notebooks, see [Create a Jupyter notebook] in the Amazon SageMaker documentation.
- Existing PDF files that have an identical format.
- An understanding of Python.

Limitations

- Your PDF files must be of good quality and clearly readable. Native PDF files are recommended, but you can use scanned documents that are converted to a PDF format if all the individual words are clear. For more information about this, see PDF document preprocessing with Amazon Textract: Visuals detection and removal on the AWS Machine Learning Blog.
- For multipage files, you can use an asynchronous operation or split the PDF files into a single page and use a synchronous operation. For more information about these two options, see Detecting and analyzing text in multipage documents and Detecting and analyzing text in single-page documents in the Amazon Textract documentation.

Architecture

This pattern's workflow first runs Amazon Textract on a sample PDF file (*First-time run*) and then runs it on PDF files that have an identical format to the first PDF (*Repeat run*). The following diagram shows the combined *First-time run* and *Repeat run* workflow that automatically and repeatedly extracts content from PDF files with identical formats.



The diagram shows the following workflow for this pattern:

1. Convert a PDF file into JPEG format and store it in an S3 bucket.
2. Call the Amazon Textract API and parse the Amazon Textract response JSON file.
3. Edit the JSON file by adding the correct `KeyName:DataType` pair for each required field. Create a `TemplateJSON` file for the *Repeat run* stage.
4. Define the post-processing correction functions for each data type (for example, float, integer, and date).
5. Prepare the PDF files that have an identical format to your first PDF file.
6. Call the Amazon Textract API and parse the Amazon Textract response JSON.
7. Match the parsed JSON file with the `TemplateJSON` file.
8. Implement post-processing corrections.

The final JSON output file has the correct `KeyName` and `Value` for each required field.

Target technology stack

- Amazon SageMaker
- Amazon S3
- Amazon Textract

Automation and scale

You can automate the *Repeat run* workflow by using an AWS Lambda function that initiates Amazon Textract when a new PDF file is added to Amazon S3. Amazon Textract then runs the processing scripts and the final output can be saved to a storage location. For more information about this, see Using an Amazon S3 trigger to invoke a Lambda function in the Lambda documentation.

Tools

- Amazon SageMaker is a fully managed ML service that helps you to quickly and easily build and train ML models, and then directly deploy them into a production-ready hosted environment.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Textract makes it easy to add document text detection and analysis to your applications.

Epics

First-time run

Task	Description	Skills required
Convert the PDF file.	Prepare the PDF file for your first-time run by splitting it into a single page and converting it into JPEG format for the Amazon Textract [synchronous operation](#) (`Syn API`). **Note** You can also use the Amazon Textract [asynchronous operation](#) (`Asyn API`) for multipage PDF files.	Data scientist, Developer

Parse the Amazon Textract response JSON.	Open the `Textract_PostProcessing.ipynb` Jupyter notebook (attached) and call the Amazon Textract API by using the following code:
	
	`response = textract.analyze_document(`
	`Document={`
	`'S3Object': {`
	`'Bucket': BUCKET,`
	`'Name': '{}'.format(filename)`
		Data scientist, Developer

Task	Description	Skills required

	``
	                    }
	                },
	        FeatureTy
	pes=["TABLES",
	 "FORMS"])
	``
	Parse the response JSON into a form and table by using the following code:
	``
	parseformKV=form_k
	v_from_JSON(response)
	parseformTable
	s=get_tables_fromJ
	SON(response)
	``	

Edit the TemplateJSON file.	Edit the parsed JSON for each `KeyName` and correspon ding `DataType` (for example, string, float, integer, or date), and table headers (for example, `ColumnNames` and `RowNames`). This template is used for each individual PDF file type, which means that the template can be reused for PDF files that have an identical format.	Data scientist, Developer

Task	Description	Skills required
Define the post-processing correction functions.	The values in Amazon Textract's response for the `TemplateJSON` file are strings. There is no differentiation for date, float, integer, or currency. These values must be converted to the correct data type for your downstream use case. Correct each data type according to the `TemplateJSON` file by using the following code: `finalJSON=postprocessingCorrection(parsedJSON,templateJSON)`	Data scientist, Developer

Repeat run

Task	Description	Skills required
Prepare the PDF files.	Prepare the PDF files by splitting them into a single page and converting them into JPEG format for the Amazon Textract [synchronous operation](https://docs.aws.amazon.com/textract/latest/dg/sync.html) (`Syn API`). **Note** You can also use the Amazon Textract	Data scientist, Developer

Task	Description	Skills required
	asynchronous operation (Asyn API) for multipage PDF files.	
Call the Amazon Textract API.	Call the Amazon Textract API by using the following code: ```response = textract.analyze_document(Document={ 'S3Object': { 'Bucket': BUCKET, 'Name': '{}'.format(filename) } }, FeatureTypes=["TABLES", "FORMS"])```	Data scientist, Developer
Parse the Amazon Textract response JSON.	Parse the response JSON into a form and table by using the following code: ```parseformKV=form_kv_from_JSON(response) parseformTables=get_tables_fromJSON(response)```	Data scientist, Developer

Task	Description	Skills required
Load the TemplateJSON file and match it with the parsed JSON.	Use the `TemplateJSON` file to extract the correct key-value pairs and table by using the following commands: `form_kv_corrected=form_kv_correction(parseformKV,templateJSON) form_table_corrected=form_Table_correction(parseformTables, templateJSON) form_kv_table_corrected_final={**form_kv_corrected , **form_table_corrected}`	Data scientist, Developer
Post-processing corrections.	Use `DataType` in the `TemplateJSON` file and post-processing functions to correct data by using the following code: `finalJSON=postprocessingCorrection(form_kv_table_corrected_final,templateJSON)`	Data scientist, Developer

Related resources

- Automatically extract text and structured data from documents with Amazon Textract
- Extract text and structured data with Amazon Textract
- Amazon Textract resources

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Build a cold start forecasting model by using DeepAR for time series in Amazon SageMaker AI Studio Lab

Ivan Cui and Eyal Shacham, Amazon Web Services

Summary

Whether you're allocating resources more efficiently for web traffic, forecasting patient demand for staffing needs, or anticipating sales of a company's products, forecasting is an essential tool. Cold start forecasting builds forecasts for a time series that has little historical data, such as a new product that just entered the retail market. This pattern uses the Amazon SageMaker AI DeepAR forecasting algorithm to train a cold start forecasting model and demonstrates how to perform forecasting on cold start items.

DeepAR is a supervised learning algorithm for forecasting scalar (one-dimensional) time series using recurrent neural networks (RNN). DeepAR takes the approach of training a single model jointly over all of the time series of related products' time series.

Traditional time series forecasting methods such as autoregressive integrated moving average (ARIMA) or exponential smoothing (ETS) rely heavily on historical time series of each individual product. Therefore, those methods aren't effective for cold start forecasting. When your dataset contains hundreds of related time series, DeepAR outperforms the standard ARIMA and ETS methods. You can also use the trained model to generate forecasts for new time series that are similar to the time series that it has been trained on.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An Amazon SageMaker AI domain.
- An Amazon SageMaker AI Studio Lab or Jupiter lab application.
- An Amazon Simple Storage Service (Amazon S3) bucket with read and write permissions.
- Knowledge of programming in Python.
- Knowledge of using a Jupyter notebook.

Limitations

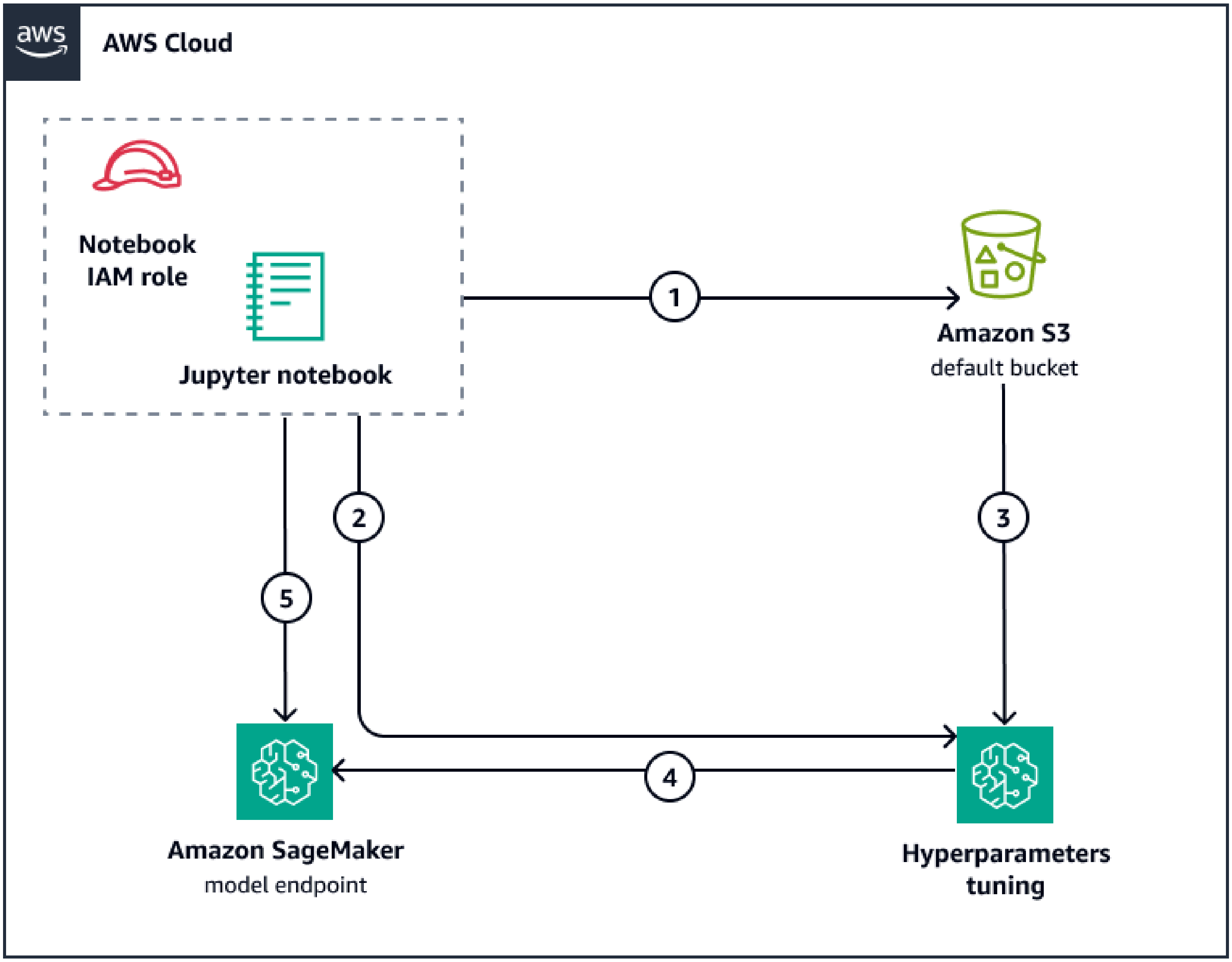
- Invoking the forecast model without any historical data points will return an error. Invoking the model with minimal historical data points will return inaccurate predictions with high confidence. This pattern suggests an approach to resolving these known limitations of cold start forecasting.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- Python version 3.10 or later.
- The pattern's notebook was tested in Amazon SageMaker AI Studio on an ml.t3.medium instance with the Python 3 (Data Science) kernel.

Architecture

The following diagram shows the workflow and architecture components for this pattern.



The workflow performs the following tasks:

1. Input files of training and testing data are synthesized and then uploaded to an Amazon S3 bucket. This data includes multiple time series with categorical and dynamic features, along with target values (to be predicted). The Jupyter notebook visualizes the data to better understand the requirements of the training data and the expected predicted values.
2. A hyperparameters tuner job is created to train the model and find the best model based on predefined metrics.
3. The input files are downloaded from the Amazon S3 bucket to each instance of the hyperparameters tuning jobs.
4. After the tuner job selects the best model based on the tuner's predefined threshold, the model is deployed as a SageMaker AI endpoint.

5. The deployed model is then ready to be invoked where its predictions are validated against the test data.

The notebook demonstrates how well the model predicts the target values when an adequate number of historical data points are available. However, when we invoke the model with fewer historical data points (which represent a cold product), the model's predictions do not match the original testing data even within the model's confidence levels. In the pattern, a new model is built for cold products where its initial context length (predicted points) is defined as the amount of available historical points, and a new model is trained iteratively as new data points are acquired. The notebook shows that the model will have accurate predictions as long as the amount of historical data points is close to its context length.

Tools

AWS services

- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon SageMaker AI is a managed machine learning (ML) service that helps you build and train ML models and then deploy them into a production-ready hosted environment.
- Amazon SageMaker AI Studio is a web-based, integrated development environment (IDE) for ML that lets you build, train, debug, deploy, and monitor your ML models.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub DeepAR-ColdProduct-Pattern repository.

Best practices

- Train your model in a virtual environment, and always use version control for the highest reproducibility effort.

- Include as many high-quality categorical features as you can to get the highest predictive model.
- Make sure that the metadata contains similar categorical items in order for the model to infer cold start products predictions adequately.
- Run a hyperparameter tuning job to get the highest predictive model.
- In this pattern, the model you build has a context length of 24 hours, which means that it will predict the next 24 hours. If you try to predict the next 24 hours when you have less than 24 hours of data historically, the model's prediction accuracy degrades linearly based on the amount of historical data points. To mitigate this issue, create a new model for each set of historical data points until this number reaches the desired prediction (context) length. For example, start with a context length model of 2 hours, then increase the model progressively to 4 hours, 8 hours, 16 hours, and 24 hours.

Epics

Start your SageMaker AI Studio Classic application

Task	Description	Skills required
Start your notebook environment.	1. Sign in to the AWS Management Console, and open the SageMaker AI Studio home page. Then choose **Open Studio**. 2. In the left navigation pane, choose the **Studio Classic** icon in **Applications**. Then, choose the **Open** button on the **Application** list. For more information, see Launch Amazon SageMaker AI Studio in the SageMaker AI documentation.	Data scientist

Create and activate the notebook

Task	Description	Skills required
Set up your virtual environment for model training.	To set up your virtual environment for model training, do the following: 1. Download the `deepar_synthetic.ipynb` notebook from this pattern's GitHub repository to your local machine. 2. In Amazon SageMaker AI Studio Classic, choose the **Upload Files** icon from the **Studio Classic** menu bar and select the downloaded notebook. 3. Choose the notebook in the **File Browser** in the left navigation pane. Follow the prompts to set up a notebook environment. Select the Data Science 3.0 Image and Python 3 Kernel. For more information, see Upload Files to SageMaker AI Studio Classic in the SageMaker AI documentation.	Data scientist
Create and validate a forecasting model.	• Follow the instructions in the notebook to create the training and testing data,	Data scientist

Task	Description	Skills required
	train the model, and then invoke the model. • Observe how accurate the model's predictions are when it's provided with adequate historical data points.	

Related resources

- DeepAR Hyperparameters
- Forecasting demand for new product introductions by using AWS machine learning services
- Launch Amazon SageMaker AI Studio Classic
- Use the SageMaker AI DeepAR forecasting algorithm

Build an MLOps workflow by using Amazon SageMaker AI and Azure DevOps

Deepika Kumar, Sara van de Moosdijk, and Philips Kokoh Prasetyo, Amazon Web Services

Summary

Machine learning operations (MLOps) is a set of practices that automate and simplify machine learning (ML) workflows and deployments. MLOps focuses on automating the ML lifecycle. It helps ensure that models are not just developed but also deployed, monitored, and retrained systematically and repeatedly. It brings DevOps principles to ML. MLOps results in faster deployment of ML models, better accuracy over time, and stronger assurance that they provide real business value.

Organizations often have existing DevOps tools and data storage solutions before starting their MLOps journey. This pattern showcases how to harness the strengths of both Microsoft Azure and AWS. It helps you integrate Azure DevOps with Amazon SageMaker AI to create an MLOps workflow.

The solution simplifies working between Azure and AWS. You can use Azure for development and AWS for machine learning. It promotes an effective process for making machine learning models from start to finish, including data handling, training, and deployment on AWS. For efficiency, you manage these processes through Azure DevOps pipelines. The solution is applicable to foundation model operations (FMOps) and large language model operations (LLMOps) in generative AI, which includes fine-tuning, vector databases, and prompt management.

Prerequisites and limitations

Prerequisites

- **Azure subscription –** Access to Azure services, such as Azure DevOps, for setting up the continuous integration and continuous deployment (CI/CD) pipelines.
- **Active AWS account –** Permissions to use the AWS services used in this pattern.
- **Data –** Access to historical data for training the machine learning model.
- **Familiarity with ML concepts –** Understanding of Python, Jupyter Notebooks, and machine learning model development.
- **Security configuration –** Proper configuration of roles, policies, and permissions across both Azure and AWS to ensure secure data transfer and access.

- **(Optional) Vector database –** If you're using a Retrieval Augmented Generation (RAG) approach and a third-party service for the vector database, you need access to the external vector database.

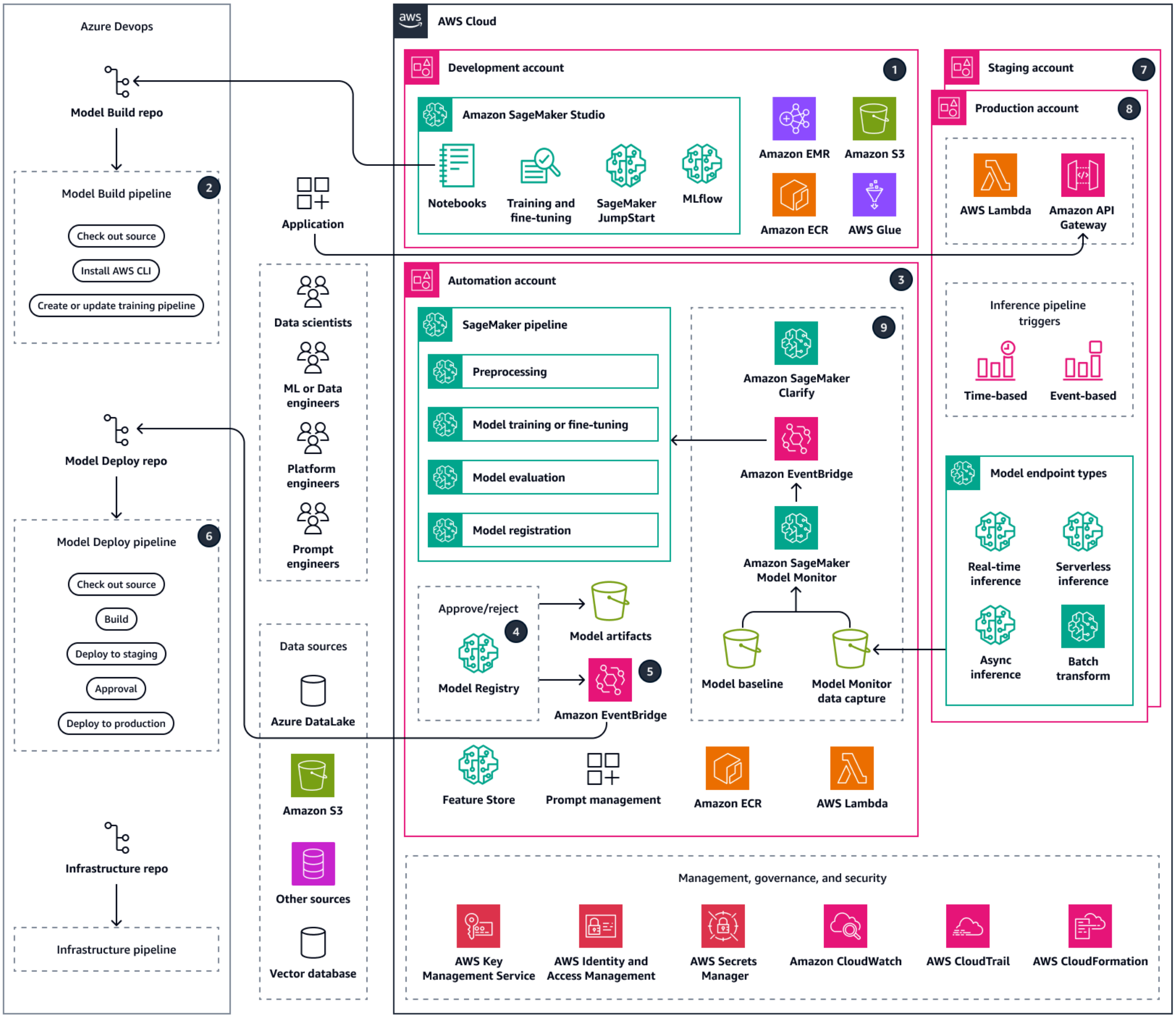
Limitations

- This guidance does not discuss secure cross-cloud data transfers. For more information about cross-cloud data transfers, see AWS Solutions for Hybrid and Multicloud.
- Multicloud solutions may increase latency for real-time data processing and model inference.
- This guidance provides one example of a multi-account MLOps architecture. Adjustments are necessary based on your machine learning and AWS strategy.
- This guidance does not describe the use of AI/ML services other than Amazon SageMaker AI.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

Target architecture

The target architecture integrates Azure DevOps with Amazon SageMaker AI, creating a cross-cloud ML workflow. It uses Azure for CI/CD processes and SageMaker AI for ML model training and deployment. It outlines the process of obtaining data (from sources such as Amazon S3, Snowflake, and Azure Data Lake) through model building and deployment. Key components include CI/CD pipelines for model building and deployment, data preparation, infrastructure management, and Amazon SageMaker AI for training and fine-tuning, evaluation, and deployment of ML models. This architecture is designed to provide efficient, automated, and scalable ML workflows across cloud platforms.



The architecture consists of the following components:

1. Data scientists perform ML experiments in the development account to explore different approaches for ML use cases by using various data sources. Data scientists perform unit tests and trials, and to track their experiments, they can use [Amazon SageMaker AI with MLflow](). In generative AI model development, data scientists fine-tune foundation models from Amazon SageMaker AI JumpStart model hub. Following model evaluation, data scientists push and merge the code to the Model Build repository, which is hosted on Azure DevOps. This repository contains code for a multi-step model building pipeline.

2. On Azure DevOps, the Model Build pipeline, which provides continuous integration (CI), can be activated automatically or manually upon code merge to the main branch. In the Automation account, this activates the SageMaker AI pipeline for data preprocessing, model training and fine-tuning, model evaluation, and conditional model registration based on accuracy.
3. The Automation account is a central account across ML platforms that hosts ML environments (Amazon ECR), models (Amazon S3), model metadata (SageMaker AI Model Registry), features (SageMaker AI Feature Store), automated pipelines (SageMaker AI Pipelines), and ML log insights (CloudWatch). For a generative AI workload, you might require additional evaluations for prompts in the downstream applications. A prompt management application helps to streamline and automate the process. This account allows reusability of ML assets and enforces best practices accelerate delivery of ML use cases.
4. The latest model version is added to SageMaker AI Model Registry for review. It tracks model versions and respective artifacts (lineage and metadata). It also manages the status of the model (approve, reject, or pending), and it manages the version for downstream deployment.
5. After a trained model in Model Registry is approved through the studio interface or an API call, an event can be dispatched to Amazon EventBridge. EventBridge starts the Model Deploy pipeline on Azure DevOps.
6. The Model Deploy pipeline, which provides continuous deployment (CD), checks out the source from the Model Deploy repository. The source contains code, the configuration for the model deployment, and test scripts for quality benchmarks. The Model Deploy pipeline can be tailored to your inference type.
7. After quality control checks, the Model Deploy pipeline deploys the model to the Staging account. The Staging account is a copy of the Production account, and it is used for integration testing and evaluation. For a batch transformation, the Model Deploy pipeline can automatically update the batch inference process to use the latest approved model version. For a real-time, serverless, or asynchronous inference, it sets up or updates the respective model endpoint.
8. After successful testing in the Staging account, a model can be deployed to the Production account by manual approval through the Model Deploy pipeline. This pipeline provisions a production endpoint in the **Deploy to production** step, including model monitoring and a data feedback mechanism.
9. After the model is in production, use tools such as SageMaker AI Model Monitor and SageMaker AI Clarify to identify bias, detect drift, and continuously monitor the model's performance.

Automation and scale

Use infrastructure as code (IaC) to automatically deploy to multiple accounts and environments. By automating the process of setting up an MLOps workflow, it is possible to separate the environments used by ML teams working on different projects. AWS CloudFormation helps you model, provision, and manage AWS resources by treating infrastructure as code.

Tools

AWS services

- Amazon SageMaker AI is a managed ML service that helps you build and train ML models and then deploy them into a production-ready hosted environment.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data. In this pattern, Amazon S3 is used for data storage and integrated with SageMaker AI for model training and model objects.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use. In this pattern, Lambda is used for data pre-processing and post-processing tasks.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable. In this pattern, it stores Docker containers that SageMaker AI uses as training and deployment environments.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. In this pattern, EventBridge orchestrates event-driven or time-based workflows that initiate automatic model retraining or deployment.
- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale. In this pattern, it is used to create an external-facing, single point of entry for SageMaker AI endpoints.
- For RAG applications, you can use AWS services, such as Amazon OpenSearch Service and Amazon RDS for PostgreSQL, to store the vector embeddings that provide the LLM with your internal data.

Other tools

- Azure DevOps helps you manage CI/CD pipelines and facilitate code builds, tests, and deployment.
- Azure Data Lake Storage or Snowflake are possible third-party sources of training data for ML models.
- Pinecone, Milvus, or ChromaDB are possible third-party vector databases to store vector embeddings.

Best practices

Before implementing any component of this multicloud MLOps workflow, complete the following activities:

- Define and understand the machine learning workflow and the tools required to support it. Different use cases require different workflows and components. For example, a feature store might be required for feature reuse and low latency inference in a personalization use case, but it may not be required for other use cases. Understanding the target workflow, use case requirements, and preferred collaboration methods of the data science team is needed to successfully customize the architecture.
- Create a clear separation of responsibility for each component of the architecture. Spreading data storage across Azure Data Lake Storage, Snowflake, and Amazon S3 can increase complexity and cost. If possible, choose a consistent storage mechanism. Similarly, avoid using a combination of Azure and AWS DevOps services, or a combination of Azure and AWS ML services.
- Choose one or more existing models and datasets to perform end-to-end testing of the MLOps workflow. The test artifacts should reflect real use cases that the data science teams develop when the platform enters production.

Epics

Design your MLOps architecture

Task	Description	Skills required
Identify data sources.	Based on current and future use cases, available data sources, and types of data (such as confidential data),	Data engineer, Data scientist, Cloud architect

Task	Description	Skills required
	document the data sources that need to be integrated with the MLOps platform. Data can be stored in Amazon S3, Azure Data Lake Storage, Snowflake, or other sources. For generative AI workloads , data might also include a knowledge base that grounds the generated response. This data is stored as vector embeddings in vector databases. Create a plan for integrating these sources with your platform and securing access to the correct resources.	
Choose applicable services.	Customize the architect ure by adding or removing services based on the desired workflow of the data science team, applicable data sources, and existing cloud architect ure. For example, data engineers and data scientists may perform data preproces sing and feature engineering in SageMaker AI, AWS Glue, or Amazon EMR. It is unlikely that all three services would be required.	AWS administrator, Data engineer, Data scientist, ML engineer

Task	Description	Skills required
Analyze security requireme nts.	Gather and document security requirements. This includes determining: • Which teams or engineers can access specific data sources • Which teams or engineers can access pre-trained foundation models • Whether teams are allowed to access the code and models of other teams • Which permissions (if any) team members should have for non-development accounts • Which security measures need to be implement ed for cross-cloud data transfer For more information about securing generative AI workloads, see [Securing generative AI: An introduction to the Generative AI Security Scoping Matrix]() (AWS blog post).	AWS administrator, Cloud architect

Set up AWS Organizations

Task	Description	Skills required
Set up AWS Organizations.	Set up AWS Organizations on the root AWS account. This helps you manage the subsequent accounts that you create as part of a multi-account MLOps strategy. For more information, see the AWS Organizations documentation.	AWS administrator

Set up the development environment and versioning

Task	Description	Skills required
Create an AWS development account.	Create an AWS account where data engineers and data scientists have permissions to experiment and create ML models. For instructions, see Creating a member account in your organization in the AWS Organizations documentation.	AWS administrator
Create a Model Build repository.	Create a Git repository in Azure where data scientists can push their model build and deployment code after the experimentation phase is complete. For instructions, see Set up a Git repository in	DevOps engineer, ML engineer

Task	Description	Skills required
	the Azure DevOps documenta tion.	
Create a Model Deploy repository.	Create a Git repository in Azure that stores standard deployment code and templates. It should include code for every deploymen t option that the organizat ion uses, as identified in the design phase. For example, it should include real-time endpoints, asynchron ous endpoints, serverless inference, or batch transform s. For instructions, see Set up a Git repository in the Azure DevOps documentation.	DevOps engineer, ML engineer
Create an Amazon ECR repository.	Set up an Amazon ECR repository that stores the approved ML environments as Docker images. Allow data scientists and ML engineers to define new environments. For instructions, see Creating a private repository in the Amazon ECR documentation.	ML engineer

Task	Description	Skills required
Set up SageMaker AI Studio.	Set up SageMaker AI Studio on the development account according to the previously defined security requireme nts, preferred data science tools (such as MLflow), and preferred integrated development environment (IDE). Use lifecycle configura tions to automate the installation of key functiona lity and create a uniform development environment for data scientists. For more information, see Amazon SageMaker AI Studio and MLflow tracking server in the SageMaker AI documentation.	Data scientist, ML engineer, Prompt engineer

Integrate CI/CD pipelines

Task	Description	Skills required
Create an Automation account.	Create an AWS account where automated pipelines and jobs run. You can give data science teams read access to this account. For instructions, see Creating a member account in your organization in the AWS Organizations documenta tion.	AWS administrator

Task	Description	Skills required
Set up a model registry.	Set up SageMaker AI Model Registry in the Automation account. This registry stores the metadata for ML models and helps certain data scientists or team leads to approve or reject models. For more information, see Register and deploy models with Model Registry in the SageMaker AI documentation.	ML engineer
Create a Model Build pipeline.	Create a CI/CD pipeline in Azure that starts manually or automatically when code is pushed to the Model Build repository. The pipeline should check out the source code and create or update a SageMaker AI pipeline in the Automation account. The pipeline should add a new model to the model registry. For more information about creating a pipeline, see the Azure Pipelines documentation.	DevOps engineer, ML engineer

Build the deployment stack

Task	Description	Skills required
Create AWS staging and deployment accounts.	Create AWS accounts for staging and deployment of	AWS administrator

Task	Description	Skills required
	ML models. These accounts should be identical to allow for accurate testing of the models in staging before moving to production. You can give data science teams read access to the staging account. For instructions, see [Creating a member account in your organization] in the AWS Organizations documentation.	
Set up S3 buckets for model monitoring.	Complete this step if you want to enable model monitoring for the deployed models that are created by the Model Deploy pipeline. Create Amazon S3 buckets for storing the input and output data. For more information about creating S3 buckets, see [Creating a bucket] in the Amazon S3 documentation. Set up cross-account permissions so that the automated model monitoring jobs run in the Automation account. For more information, see [Monitor data and model quality] in the SageMaker AI documentation.	ML engineer

Task	Description	Skills required
Create a Model Deploy pipeline.	Create a CI/CD pipeline in Azure that starts when a model is approved in the model registry. The pipeline should check out the source code and model artifact, build the infrastructure templates for deploying the model in the staging and production accounts, deploy the model in the staging account, run automated tests, wait for manual approval, and deploy the approved model into the production account. For more information about creating a pipeline, see the [Azure Pipelines documentation].	DevOps engineer, ML engineer

(Optional) Automate ML environment infrastructure

Task	Description	Skills required
Build AWS CDK or CloudForm ation templates.	Define AWS Cloud Developme nt Kit (AWS CDK) or AWS CloudFormation templates for all environments that need to be deployed automatically. This might include the development environment, automation environment, and staging and deployment environments. For more information, see	AWS DevOps

Task	Description	Skills required
	the [AWS CDK](#) and [CloudFormation](#) documentation.	
Create an Infrastructure pipeline.	Create a CI/CD pipeline in Azure for infrastructure deployment. An administrator can initiate this pipeline to create new AWS accounts and set up the environments that the ML team requires.	DevOps engineer

Troubleshooting

Issue	Solution
**Insufficient monitoring and drift detection – **Inadequate monitoring can lead to missed detection of model performance issues or data drift.	Strengthen monitoring frameworks with tools such as Amazon CloudWatch, SageMaker AI Model Monitor, and SageMaker AI Clarify. Configure alerts for immediate action on identified issues.
CI pipeline trigger errors – The CI pipeline in Azure DevOps might not be triggered upon code merge due to misconfiguration.	Check the Azure DevOps project settings to ensure that the webhooks are properly set up and pointing to the correct SageMaker AI endpoints.
Governance – The central Automation account might not enforce best practices across ML platforms, leading to inconsistent workflows.	Audit the Automation account settings, ensuring that all ML environments and models conform to predefined best practices and policies.
Model registry approval delays – This happens when there's a delay in checking and approving the model, either because people	Implement a notification system to alert stakeholders of models that are pending approval, and streamline the review process.

Issue	Solution
take time to review it or because of technical issues.	
Model deployment event failures – Events dispatched to start model deployment pipelines might fail, causing deployment delays.	Confirm that Amazon EventBridge has the correct permissions and event patterns to invoke Azure DevOps pipelines successfully.
Production deployment bottlenecks – Manual approval processes can create bottlenecks, delaying the production deployment of models.	Optimize the approval workflow within the model deploy pipeline, promoting timely reviews and clear communication channels.

Related resources

AWS documentation

- Amazon SageMaker AI documentation
- Machine Learning Lens (AWS Well Architected Framework)
- Planning for successful MLOps (AWS Prescriptive Guidance)

Other AWS resources

- MLOps foundation roadmap for enterprises with Amazon SageMaker AI (AWS blog post)
- AWS Summit ANZ 2022 - End-to-end MLOps for architects (YouTube video)
- FMOps/LLMOps: Operationalize generative AI and differences with MLOps (AWS blog post)
- Operationalize LLM Evaluation at Scale using Amazon SageMaker AI Clarify and MLOps services (AWS blog post)
- The role of vector databases in generative AI applications (AWS blog post)

Azure documentation

- Azure DevOps documentation
- Azure Pipelines documentation

Configure model invocation logging in Amazon Bedrock by using AWS CloudFormation

Vikramaditya Bhatnagar, Amazon Web Services

Summary

You can configure Amazon Bedrock to collect invocation logs, model input data, and model output data for all model invocations in your AWS account. This is a best practice for building robust generative AI applications with Amazon Bedrock. You can store model invocation logs in an Amazon CloudWatch Logs log group, in an Amazon Simple Storage Service (Amazon S3) bucket, or in both. Having log data in CloudWatch Logs helps you create custom metric filters, alarms, and dashboards. Amazon S3 is ideal for replicating data across AWS Regions or for long-term storage, as governed by your organization's policies.

This pattern provides a sample AWS CloudFormation template that uses an infrastructure as code (IaC) approach to configure model invocation logging for Amazon Bedrock. The template configures log storage in both CloudWatch Logs and Amazon S3.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The following permissions:
  - Permissions to create CloudFormation stacks
  - Permissions to access Amazon Bedrock
  - Permissions to create and access Amazon S3 buckets
  - Permissions to create and access CloudWatch Logs log groups
  - Permissions to create and access AWS Lambda functions
  - Permissions to create and access AWS Key Management Service (AWS KMS) keys

Limitations

This pattern logs model invocations to both CloudWatch Logs and Amazon S3. It does not support choosing only one of these two services.

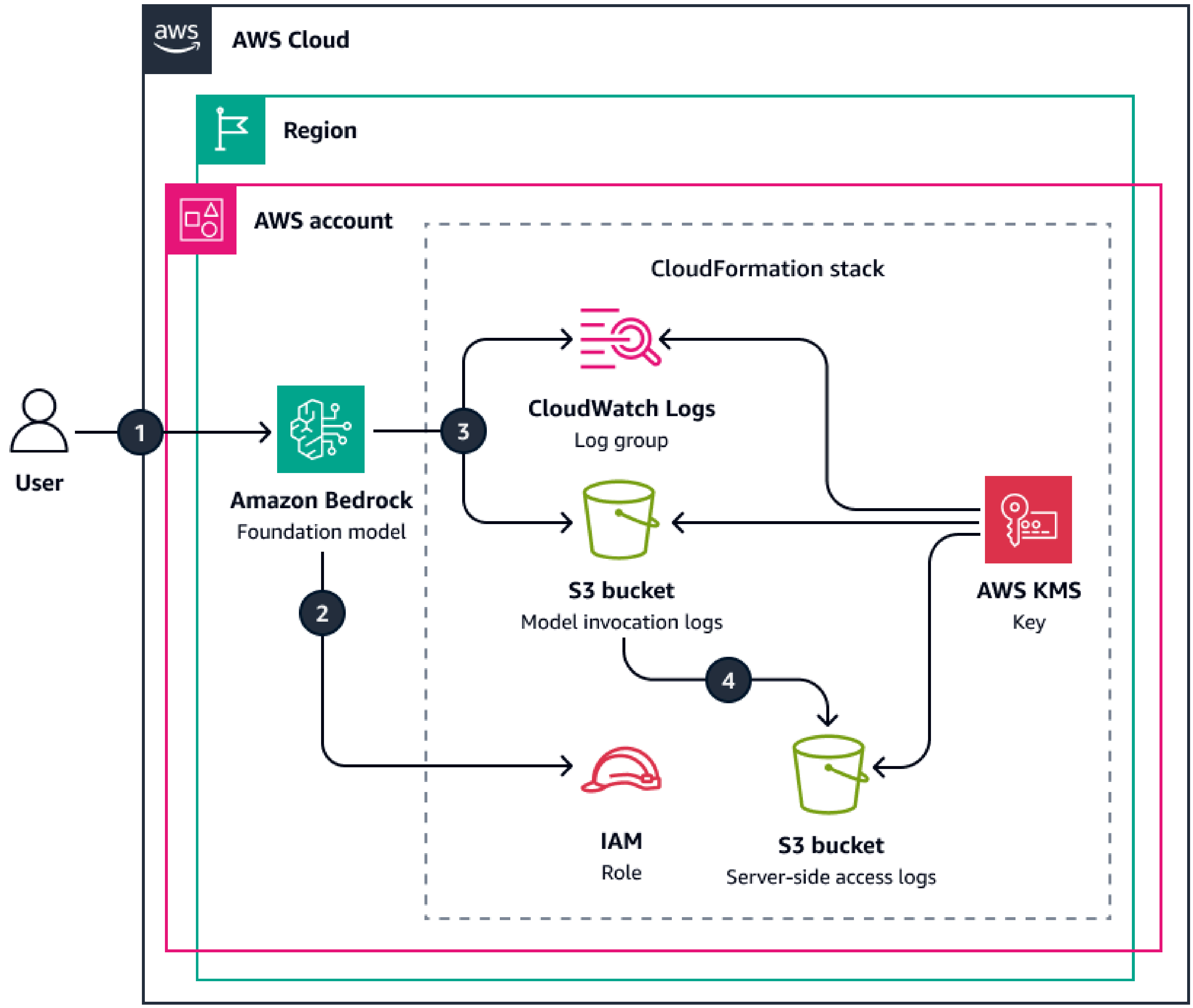
Architecture

Target architecture

The CloudFormation template provisions the following resources in your target AWS account:

- A CloudWatch Logs log group for storing model invocation logs
- An Amazon S3 bucket for storing model invocation logs and a corresponding bucket policy
- An Amazon S3 bucket for storing server-side access logs and a corresponding bucket policy
- An AWS Lambda function that configures logging settings in Amazon Bedrock
- An AWS KMS key and a corresponding key alias
- An AWS Identity and Access Management (IAM) service role for Amazon Bedrock

The following diagram shows how invocation logs are stored after you deploy the CloudFormation stack associated with this pattern. Amazon Bedrock publishes log data when the foundation model delivers text, an image, a video, or embedding data. As shown in the diagram, the Amazon S3 buckets and the CloudWatch Logs log group are encrypted with an AWS KMS key.



The diagram shows the following workflow:

1. A user submits a query to a foundation model in Amazon Bedrock.
2. Amazon Bedrock assumes the IAM service role.
3. Amazon Bedrock generates log data and stores it in an CloudWatch Logs log group and in an Amazon S3 bucket.
4. If a user reads, uploads, or deletes any files in the Amazon S3 bucket that contains the model invocation logs, those activities are logged in another Amazon S3 bucket for server-side access logs.

Automation and scale

To scale this solution, you can deploy the CloudFormation template as a stack set to multiple AWS Regions and AWS accounts. For more information, see Managing stacks across accounts and Regions with StackSets in the CloudFormation documentation.

Tools

AWS services

- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI companies and Amazon available for your use through a unified API.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- Amazon CloudWatch Logs helps you centralize the logs from all of your systems, applications, and AWS services so you can monitor them and archive them securely.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Other tools

- Git is an open source, distributed version control system.

Code repository

The code for this pattern is available in the GitHub enable-bedrock-logging-using-cloudformation repository.

Epics

Create the CloudFormation stack

Task	Description	Skills required
Download the CloudForm ation template.	Download the [CloudForm ation template](#) from the GitHub repository.	Cloud architect
Deploy the template.	Create a stack in your target account and Region. In the **Parameters** section, specify values for the parameter s that are defined in the template. For instructions, see [Creating a stack](#) in the CloudFormation documenta tion.	Cloud architect

Test the solution

Task	Description	Skills required
Enable model access.	In Amazon Bedrock, add access to the foundation model. For instructions, see [Add or remove access to Amazon Bedrock foundatio n models](#) in the Amazon Bedrock documentation.	Cloud architect
Run a sample prompt.	In Amazon Bedrock playgroun ds, run a sample prompt. For instructions, see [Generate responses in the console using playgrounds](#) in the	Cloud architect

Task	Description	Skills required
	Amazon Bedrock documenta tion.	
Review the logging configura tion.	1. Sign into the [Amazon Bedrock console](). 2. In the navigation bar, choose the AWS Region where you deployed the CloudFormation stack. 3. In the left navigation pane, under **Bedrock configura tions**, choose **Settings**. 4. Confirm the following: • Model invocation logging is enabled. • All data types are selected. • For logging destinations, **Both S3 and CloudWatc h Logs** is selected.	Cloud architect
Review the Amazon S3 bucket.	1. In the **S3 configurations** section, choose **Browse S3**. This opens the target bucket in the Amazon S3 console. 2. Confirm that logging data is present for the sample prompt that you previously ran.	Cloud architect

Task	Description	Skills required
Review the log group.	1. Navigate back to the **Settings** page in the Amazon Bedrock console. 2. In the **CloudWatch Logs configurations** section, review the settings for the CloudWatch Logs log group. Make note of the log group name. 3. Open the CloudWatch console. 4. In the navigation pane, under **Logs**, choose **Log groups**. 5. Choose the name of the log group where Amazon Bedrock publishes log data. 6. Confirm that logging data is present for the sample prompt that you previously ran.	Cloud architect

Related resources

AWS documentation

- Accessing an Amazon S3 bucket (Amazon S3 documentation)
- Creating and managing stacks (CloudFormation documentation)
- Monitor model invocation (Amazon Bedrock documentation)
- Working with log groups and log streams (CloudWatch Logs documentation)

AWS blog posts

- Monitoring Generative AI applications using Amazon Bedrock and Amazon CloudWatch integration
- Best practices for building robust generative AI applications with Amazon Bedrock Agents – Part 1
- Best practices for building robust generative AI applications with Amazon Bedrock Agents – Part 2

Create a custom Docker container image for SageMaker and use it for model training in AWS Step Functions

Julia Bluszcz, Aubrey Oosthuizen, Mohan Gowda Purushothama, Neha Sharma, and Mateusz Zaremba, Amazon Web Services

Summary

This pattern shows how to create a Docker container image for Amazon SageMaker and use it for a training model in AWS Step Functions. By packaging custom algorithms in a container, you can run almost any code in the SageMaker environment, regardless of programming language, framework, or dependencies.

In the example SageMaker notebook provided, the custom Docker container image is stored in Amazon Elastic Container Registry (Amazon ECR). Step Functions then uses the container that's stored in Amazon ECR to run a Python processing script for SageMaker. Then, the container exports the model to Amazon Simple Storage Service (Amazon S3).

Prerequisites and limitations

Prerequisites

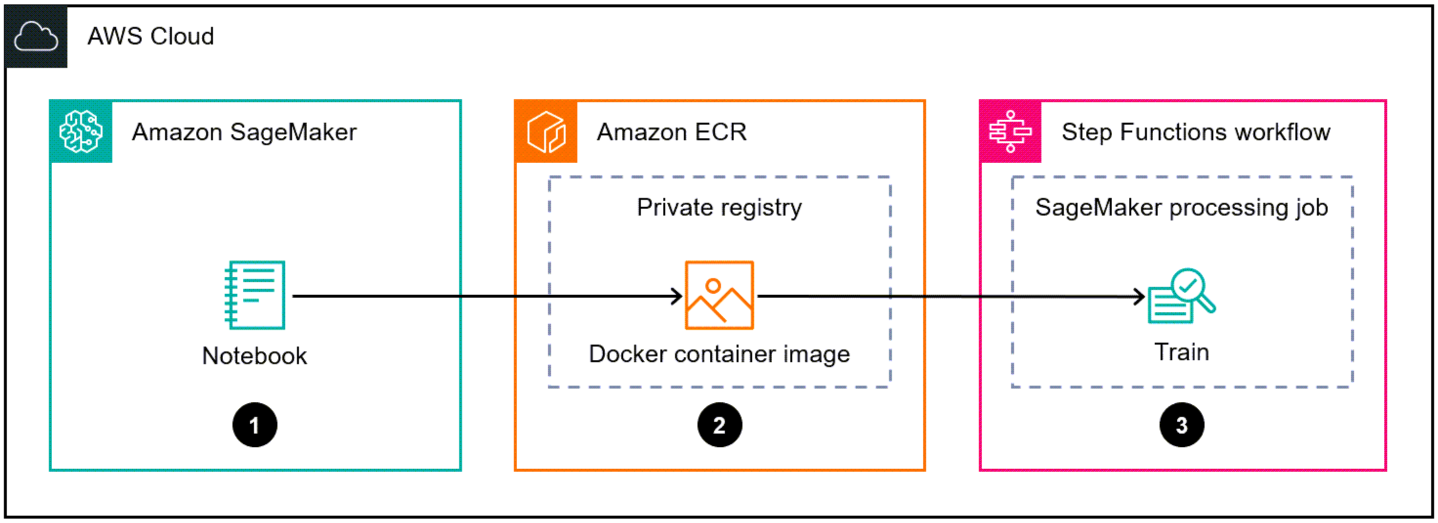
- An active AWS account
- An AWS Identity and Access Management (IAM) role for SageMaker with Amazon S3 permissions
- An IAM role for Step Functions
- Familiarity with Python
- Familiarity with the Amazon SageMaker Python SDK
- Familiarity with the AWS Command Line Interface (AWS CLI)
- Familiarity with AWS SDK for Python (Boto3)
- Familiarity with Amazon ECR
- Familiarity with Docker

Product versions

- AWS Step Functions Data Science SDK version 2.3.0
- Amazon SageMaker Python SDK version 2.78.0

Architecture

The following diagram shows an example workflow for creating a Docker container image for SageMaker, then using it for a training model in Step Functions:



The diagram shows the following workflow:

1. A data scientist or DevOps engineer uses a Amazon SageMaker notebook to create a custom Docker container image.
2. A data scientist or DevOps engineer stores the Docker container image in an Amazon ECR private repository that's in a private registry.
3. A data scientist or DevOps engineer uses the Docker container to run a Python SageMaker processing job in a Step Functions workflow.

Automation and scale

The example SageMaker notebook in this pattern uses an `ml.m5.xlarge` notebook instance type. You can change the instance type to fit your use case. For more information about SageMaker notebook instance types, see [Amazon SageMaker Pricing].

Tools

- [Amazon Elastic Container Registry (Amazon ECR)] is a managed container image registry service that's secure, scalable, and reliable.
- [Amazon SageMaker] is a managed machine learning (ML) service that helps you build and train ML models and then deploy them into a production-ready hosted environment.

- Amazon SageMaker Python SDK is an open source library for training and deploying machine-learning models on SageMaker.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- AWS Step Functions Data Science Python SDK is an open source library that helps you create Step Functions workflows that process and publish machine learning models.

Epics

Create a custom Docker container image and store it in Amazon ECR

Task	Description	Skills required
Setup Amazon ECR and create a new private registry.	If you haven't already, set up Amazon ECR by following the instructions in Setting up with Amazon ECR in the *Amazon ECR User Guide*. Each AWS account is provided with a default private Amazon ECR registry.	DevOps engineer
Create an Amazon ECR private repository.	Follow the instructions in Creating a private repository in the *Amazon ECR User Guide*. **Note** The repository that you create is where you'll store your custom Docker container images.	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a Dockerfile that includes the specifica tions needed to run your SageMaker processing job.</td><td>Create a Dockerfile that includes the specifica tions needed to run your SageMaker processing job by configuring a Dockerfile. For instructions, see Adapting your own training container in the Amazon SageMaker Developer Guide.

For more information about Dockerfiles, see the Dockerfil e Reference in the Docker documentation.

Example Jupyter notebook code cells to create a Dockerfile

Cell 1

<pre># Make docker folder
!mkdir -p docker</pre>
Cell 2

<pre>%%writefile docker/Do
ckerfile

FROM python:3.7-slim-bu
ster

RUN pip3 install
 pandas==0.25.3 scikit-
learn==0.21.3
ENV PYTHONUNBUFFERED=T
RUE</pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	`ENTRYPOINT ["python3"]`	

Task	Description	Skills required
Build your Docker container image and push it to Amazon ECR.	1. Build the container image using the Dockerfile that you created by running the `docker build` command in the AWS CLI. 2. Push the container image to Amazon ECR by running the `docker push` command. For more information, see [Building and registering the container](#) in *Building your own algorithm container* on GitHub. **Example Jupyter notebook code cells to build and register a Docker image** **Important** Before running the following cells, make sure that you've created a Dockerfil e and stored it in the directory called `docker`. Also, make sure that you've created an Amazon ECR repository, and that you replace the `ecr_repository` value in the first cell	DevOps engineer

Task	Description	Skills required

	with your repository's name.
	
	Cell 1
	
	`import boto3`
	`tag = ':latest'`
	`account_id = boto3.client('sts').get_caller_identity().get('Account')`
	`region = boto3.Session().region_name`
	`ecr_repository = 'byoc'`
	
	`image_uri = '{}.dkr.ecr.{}.amazonaws.com/{}'.format(account_id, region, ecr_repository + tag)`
	
	Cell 2
	
	`# Build docker image`
	`!docker build -t $image_uri docker`
	
	Cell 3
	
	`# Authenticate to ECR`
	`!aws ecr get-login-password --region {region} | docker login --username AWS --password-stdin {account_id}.dkr.ecr.{region}.amazonaws.com`	

Task	Description	Skills required

	Cell 4
	`# Push docker image`
	`!docker push $image_uri`
	
	Note
	You must [authenticate your Docker client to your private registry](#) so that you can use the `docker push` and `docker pull` commands. These commands push and pull images to and from the repositories in your registry.	

Create a Step Functions workflow that uses your custom Docker container image

Task	Description	Skills required
Create a Python script that includes your custom processing and model training logic.	Write custom processing logic to run in your data processing script. Then, save it as a Python script named `training.py` . For more information, see [Bring your own model with](#)	Data scientist

Task	Description	Skills required

	SageMaker Script Mode on GitHub.
	
	Example Python script that includes custom processing and model training logic
	
	`%%writefile training.py`
	`from numpy import empty`
	`import pandas as pd`
	`import os`
	`from sklearn import datasets, svm`
	`from joblib import dump, load`
	
	`if __name__ == '__main__':`
	`    digits = datasets.load_digits()`
	`    #create classifier object`
	`    clf = svm.SVC(gamma=0.001, C=100.)`
	
	`    #fit the model`
	`    clf.fit(digits.data[:-1], digits.target[:-1])`
	
	`    #model output in binary format`
	`    output_path = os.path.join('/opt/ml/processing/model', "model.joblib")`
	`    dump(clf, output_path)`	

Task	Description	Skills required

Create a Step Functions workflow that includes your SageMaker Processing job as one of the steps.	Install and import the [AWS Step Functions Data Science SDK] and upload the **training.py** file to Amazon S3. Then, use the [Amazon SageMaker Python SDK] to define a processing step in Step Functions.
	
	⚠ **Important**
	Make sure that you've [created an IAM execution role for Step Functions] in your AWS account.
	
	Example environment set up and custom training script to upload to Amazon S3
	
	`!pip install stepfunct ions`
	
	`import boto3`
	`import stepfunctions`
	`import sagemaker`
	`import datetime`
	
	`from stepfunctions import steps`
	`from stepfunct ions.inputs import ExecutionInput`
	`from stepfunct ions.steps import (`
		Data scientist

Task	Description	Skills required

	``
	    Chain
	)
	from stepfunct
	ions.workflow import
	 Workflow
	from sagemaker
	.processing import
	 ScriptProcessor,
	 ProcessingInput,
	 ProcessingOutput
	
	sagemaker_session =
	 sagemaker.Session()
	bucket = sagemaker
	_session.default_b
	ucket()
	role = sagemaker
	.get_execution_role()
	prefix = 'byoc-tra
	ining-model'
	
	# See prerequisites
	 section to create this
	 role
	workflow_execution_rol
	e = f"arn:aws:iam::
	{account_id}:role/Ama
	zonSageMaker-StepF
	unctionsWorkflowEx
	ecutionRole"
	
	execution_input =
	 ExecutionInput(
	    schema={
	        "Preproce
	ssingJobName": str})
	
	input_code = sagemaker
	_session.upload_data(
	    "training.py",
	    bucket=bucket,
	``	

Task	Description	Skills required
	``` key_prefix="prepro cessing.py",) ``` **Example SageMaker processing step definition that uses a custom Amazon ECR image and Python script** **Note** Make sure that you use the `execution_input` parameter to specify the job name. The parameter's value must be unique each time the job runs. Also, the **training.py** file's code is passed as an `input` parameter to the `ProcessingStep`, which means that it will be copied inside the container. The destination for the `ProcessingInput` code is the same as the second argument inside the `container_entrypoint`.	

Task	Description	Skills required

	``
	script_processor =
	 ScriptProcessor(co
	mmand=['python3'],
	
	 image_uri=image_uri,
	
	 role=role,
	
	 instance_count=1,
	
	 instance_type='ml.
	m5.xlarge')
	
	processing_step =
	 steps.ProcessingStep(
	    "training-step",
	    processor=script_p
	rocessor,
	    job_name=execution
	_input["Preprocess
	ingJobName"],
	    inputs=[
	        Processin
	gInput(
	            source=in
	put_code,
	            destinati
	on="/opt/ml/proces
	sing/input/code",
	            input_nam
	e="code",
	        ),
	    ],
	    outputs=[
	        Processin
	gOutput(
	            source='/
	opt/ml/processing/
	model',
	``	

Task	Description	Skills required

	``
	            destinati
	on="s3://{}/{}".fo
	rmat(bucket, prefix),
	            output_na
	me='byoc-example')
	    ],
	    container_entrypoi
	nt=["python3", "/opt/
	ml/processing/input/c
	ode/training.py"],
	)
	``
	
	Example Step Functions workflow that runs a SageMaker processing job
	
	Note
	
	This example workflow includes the SageMaker processing job step only, not a complete Step Functions workflow. For a full example workflow, see [Example notebooks in SageMaker]() in the AWS Step Functions Data Science SDK documentation.
	
	``
	workflow_graph =
	 Chain([processing_
	step])
	``	

Task	Description	Skills required

	``
	workflow = Workflow(
	    name="ProcessingWo
	rkflow",
	    definition=workflo
	w_graph,
	    role=workflow_exec
	ution_role
	)
	
	workflow.create()
	# Execute workflow
	execution = workflow.
	execute(
	    inputs={
	        "Preproce
	ssingJobName":
	 str(datetime.datet
	ime.now().strftime
	("%Y%m%d%H%M-%SS")),
	  # Each pre processin
	g job (SageMaker
	 processing job)
	 requires a unique name,
	    }
	)
	execution_output =
	 execution.get_outp
	ut(wait=True)
	``	

Related resources

- Process data (*Amazon SageMaker Developer Guide*)
- Adapting your own training container (*Amazon SageMaker Developer Guide*)

Use Amazon Bedrock agents to automate creation of access entry controls in Amazon EKS through text-based prompts

Keshav Ganesh and Sudhanshu Saurav, Amazon Web Services

Summary

Organizations face challenges in managing access controls and resource provisioning when multiple teams need to work with a shared Amazon Elastic Kubernetes Service (Amazon EKS) cluster. A managed Kubernetes service such as Amazon EKS has simplified cluster operations. However, the administrative overhead of managing team access and resource permissions remains complex and time-consuming.

This pattern shows how Amazon Bedrock agents can help you automate Amazon EKS cluster access management. This automation allows development teams to focus on their core application development rather than dealing with access control setup and management. You can customize an Amazon Bedrock agent to perform actions for a wide variety of tasks through simple natural language prompts.

By using AWS Lambda functions as action groups, an Amazon Bedrock agent can handle tasks such as creating user access entries and managing access policies. In addition, an Amazon Bedrock agent can configure pod identity associations that allow access to AWS Identity and Access Management (IAM) resources for the pods running in the cluster. Using this solution, organizations can streamline their Amazon EKS cluster administration with simple text-based prompts, reduce manual overhead, and improve overall development efficiency.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Established IAM roles and permissions for the deployment process. This includes permissions to access Amazon Bedrock foundation models (FM), create Lambda functions, and any other required resources across the target AWS accounts.
- Access enabled in the active AWS account to these Amazon Bedrock FMs: Amazon Titan Text Embeddings V2 and Anthropic Claude 3 Haiku.
- AWS Command Line Interface (AWS CLI) version 2.9.11 or later, installed and configured.

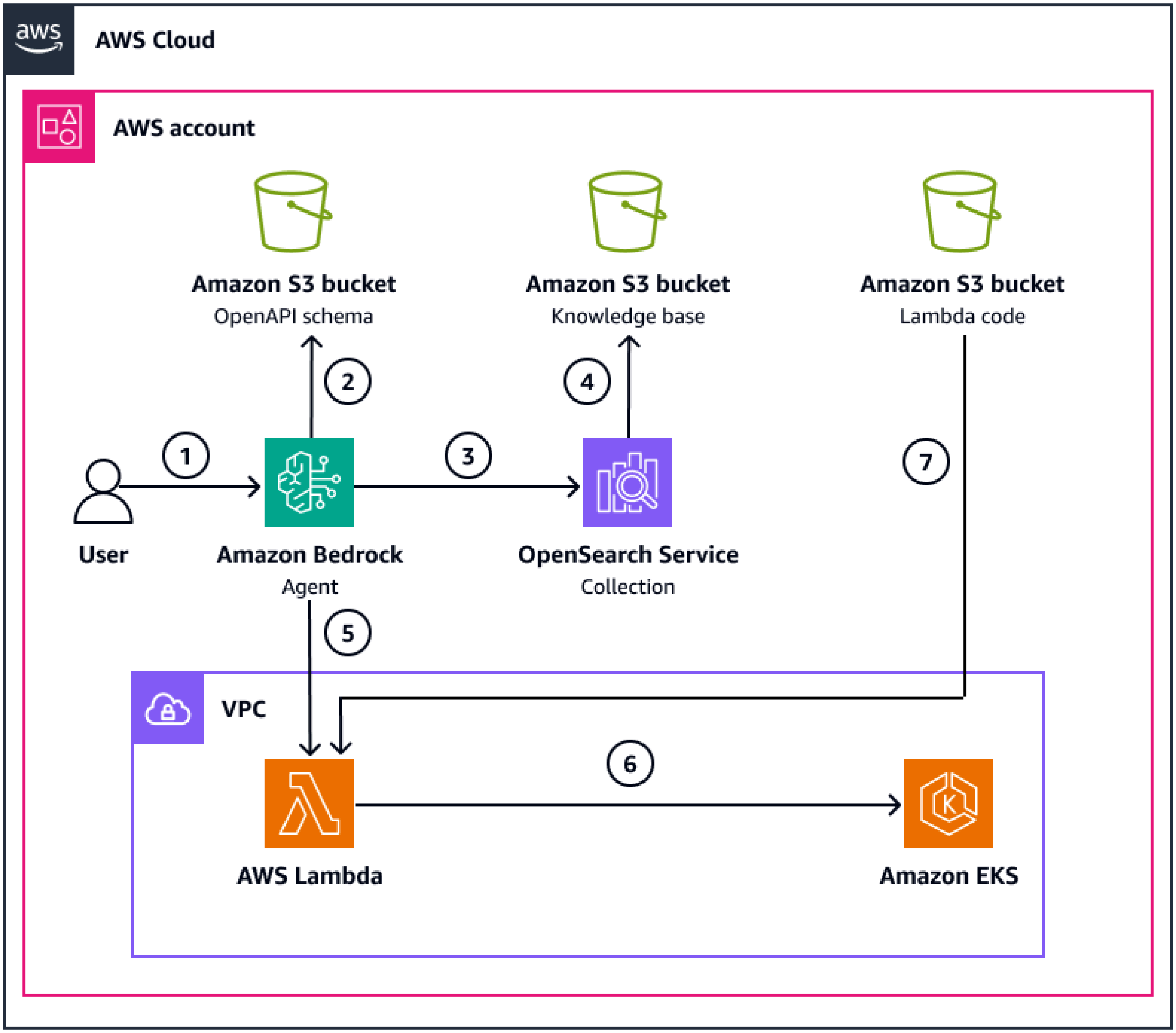
- eksctl 0.194.0 or later, installed.

Limitations

- Training and documentation might be required to help ensure smooth adoption and effective use of these techniques. Using Amazon Bedrock, Amazon EKS, Lambda, Amazon OpenSearch Service, and OpenAPI involve a significant learning curve for developers and DevOps teams.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

The following diagram shows the workflow and architecture components for this pattern.



This solution performs the following steps:

1. The user interacts with the Amazon Bedrock agent by submitting a prompt or query that serves as input for the agent to process and take action.
2. Based on the prompt, the Amazon Bedrock agent checks the OpenAPI schema to identify the correct API to target. If the Amazon Bedrock agent finds the correct API call, the request goes to the action group that is associated with the Lambda function that implements these actions.

3. If a relevant API isn't found, the Amazon Bedrock agent queries the OpenSearch collection. The OpenSearch collection uses indexed knowledge base content that is sourced from the Amazon S3 bucket that contains the *Amazon EKS User Guide*.
4. The OpenSearch collection returns relevant contextual information to the Amazon Bedrock agent.
5. For actionable requests (those that match an API operation), the Amazon Bedrock agent executes within a virtual private cloud (VPC) and triggers the Lambda function.
6. The Lambda function performs an action that's based on the user's input inside the Amazon EKS cluster.
7. The Amazon S3 bucket for the Lambda code stores the artifact that has the code and logic written for the Lambda function.

Tools

AWS services

- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon OpenSearch Service is a managed service that helps you deploy, operate, and scale OpenSearch clusters in the AWS Cloud. Its collections feature helps you to organize your data and build comprehensive knowledge bases that AI assistants such as Amazon Bedrock agents can use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- eksctl is a command-line utility for creating and managing Kubernetes clusters on Amazon EKS.

Code repository

The code for this pattern is available in the GitHub eks-access-controls-bedrock-agent repository.

Best practices

- Maintain the highest possible security when implementing this pattern. Make sure that the Amazon EKS cluster is private, has limited access permissions, and all the resources are inside a virtual private cloud (VPC). For additional information, see Best practices for security in the Amazon EKS documentation.
- Use AWS KMS customer managed keys wherever possible, and grant limited access permissions to them.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Set up the environment

Task	Description	Skills required
Clone the repository.	To clone this pattern's repository, run the following command in your local workstation: `git clone https://github.com/aws-samples/eks-access-controls-bedrock-agent.git`	AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Get the AWS account ID.</td><td>To get the AWS account ID, use the following steps:

1. Open a shell in the root folder of the cloned repo, <code>eks-access-controls-bedrock-agent</code> .
2. To get your AWS account ID, navigate to the cloned directory and run the following command:

<pre>AWS_ACCOUNT=$(aws sts
 get-caller-identity
 --query "Account" --
output text)</pre>
This command stores your AWS account ID in the <code>AWS_ACCOUNT</code> variable.</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required

Create the S3 bucket for Lambda code.	To implement this solution, you must create three Amazon S3 buckets that serve different purposes, as shown in the [architecture](#) diagram. The S3 buckets are for Lambda code, a knowledge base, and OpenAPI schema.
	
	To create the Lambda code bucket, use the following steps:
	
	1. To create an S3 bucket for Lambda code, run the following command:
	`aws s3 mb s3://bedrock-agent-lambda-artifacts-${AWS_ACCOUNT} --region us-east-1`
	2. To install the Lambda code dependency, run the following command:
	`cd eks-lambda`
	`npm install`
	`tsc`
	`cd .. && cd opensearch-lambda`
	`npm install`
	`tsc`
	`cd ..`
	3. To package the code and upload it to the S3
		AWS DevOps

Task	Description	Skills required
	bucket for Lambda, run the following command: `aws cloudformation package \ --template-file eks-access-controls.yaml \ --s3-bucket bedrock-agent-lambda-artifacts-${AWS_ACCOUNT} \ --output-template-file eks-access-controls-template.yaml \ --region us-east-1` The package command creates a new CloudFormation template (`eks-access-controls-template.yaml`) that contains: • References to the Lambda function code stored in your S3 bucket. • Definitions for all required AWS infrastructure including VPC, subnets, Amazon Bedrock agent, and OpenSearch collection. You can use this template to deploy the complete solution by using CloudFormation.	

Task	Description	Skills required

Create the S3 bucket for the knowledge base.	To create the Amazon S3 bucket for the knowledge base, use the following steps:
	1. To create the Amazon S3 bucket for the knowledge base, run the following command:
	`aws s3 mb s3://eks-knowledge-base-${AWS_ACCOUNT} --region us-east-1`
	2. To download the *Amazon EKS User Guide* and store it in a directory, run the following commands:
	`mkdir dataSource`
	`cd dataSource`
	`curl https://docs.aws.amazon.com/pdfs/eks/latest/userguide/eks-ug.pdf -o eks-user-guide.pdf`
	3. To upload the user guide to the S3 bucket that you created in step 1, run the following command:
	`aws s3 cp eks-user-guide.pdf s3://eks-knowledge-base-${AWS_ACCOUNT} \`
	`--region us-east-1 \`
		AWS DevOps

Task	Description	Skills required
	4. To return to the root directory, run the following command: `cd ..`	

Create the S3 bucket for the OpenAPI schema.	To create the Amazon S3 bucket for the OpenAPI schema, use the following steps:
	1. To create the S3 bucket, run the following command:
	`aws s3 mb s3://eks-openapi-schema-${AWS_ACCOUNT} --region us-east-1`
	2. To upload the OpenAPI schema to the S3 bucket, run the following command:
	`aws s3 cp openapi-schema.yaml s3://eks-openapi-schema-${AWS_ACCOUNT} --region us-east-1`
		AWS DevOps

Deploy the CloudFormation stack

Task	Description	Skills required
Deploy the CloudFormation stack.	To deploy the CloudFormation stack, use the CloudFormation template file `eks-access-controls-template.yaml` that you created earlier. For more detailed instructions, see [Create a stack from the CloudFormation console](#) in the CloudFormation documentation. **Note** Provisioning the OpenSearch index with the CloudFormation template takes about 10 minutes. After the stack is created, make a note of the `VPC_ID` and `PRIVATE_SUBNET ID` s.	AWS DevOps
Create the Amazon EKS cluster.	To create the Amazon EKS cluster inside the VPC, use the following steps: 1. Create a copy of the `eks-config.yaml` configuration file, and name the copy as `eks-deploy.yaml` .	AWS DevOps

Task	Description	Skills required

	2. Open `eks-deploy.yaml` in a text editor. Then, replace the following placeholder values with values from the deployed stack: `VPC_ID`, `PRIVATE_SUBNET1`, and `PRIVATE_SUBNET2`
	3. To create the cluster by using the eksctl utility, run the following command:
	`eksctl create cluster -f eks-deploy.yaml`
	Note
	This cluster creation process can take up to 15-20 minutes to complete.
	4. To verify that the cluster was created successfully, run the following commands:
	`aws eks describe-cluster --name --query "cluster.status"`
	`aws eks update-kubeconfig --name --region`
	`kubectl get nodes`	

Task	Description	Skills required
	The expected results are as follows: • The cluster status is `ACTIVE`. • The command `kubectl get nodes` shows that all nodes are in `Ready` state.	

Connect the Lambda function and the Amazon EKS cluster

Task	Description	Skills required
Create a connection between the Amazon EKS cluster and the Lambda function.	To set up network and IAM permissions to allow the Lambda function to communicate with the Amazon EKS cluster, use the following steps: 1. To identify the IAM role that's attached to the Lambda function, open the AWS Management Console and locate the Lambda function named `bedrock-agent-eks-access-control` . Make a note of the Amazon Resource Name (ARN) of the IAM role. 2. To create an access entry in the Amazon EKS cluster for the Lambda function's	AWS DevOps

Task	Description	Skills required

	IAM role, run the following command:
	`aws eks create-access-entry --cluster-name eks-testing-cluster --principal-arn <principal-Role-ARN>`
	3. To assign `AmazonEKSClusterAdminPolicy` permissions to this role, run the following command:
	`aws eks associate-access-policy --cluster-name eks-testing-cluster --principal-arn <principal-Role-ARN> --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy --access-scope type=cluster`
	For more information, see [Associate access policies with access entries] and [AmazonEKSClusterAdminPolicy] in the Amazon EKS documentation.
	4. Locate the Amazon EKS cluster's security group.	

Task	Description	Skills required
	Add an inbound rule to allow incoming network traffic from the Lambda function to the Amazon EKS cluster. Use the following values for the inbound rule: • **Type** – HTTPS • **Port range** – 443 • **Source** – Lambda security group For more information, see Configure security group rules in the Amazon VPC documentation.	

Test the solution

Task	Description	Skills required
Test the Amazon Bedrock agent.	Before testing the Amazon Bedrock agent, make sure that you do the following: • Test with non-production roles first. • Document any changes made to cluster access. • Have a plan to revert changes if needed.	AWS DevOps

Task	Description	Skills required
	To access the Amazon Bedrock agent, use the following steps: 1. Sign in to the AWS Management Console using an [IAM role with Amazon Bedrock permissions](), and open the Amazon Bedrock console at [https://console.aws.amazon.com/bedrock/](). 2. Select **Agents** from the left navigation pane. Then, choose your configured agent in the **Agents** section. 3. To test the agent, try the following sample prompts in which you replace `Principal-ARN-OF-ROLE` with an actual IAM role ARN: • To create an access entry for any IAM role that you wanted to provide access to the EKS cluster, use the following prompt: `Create an access entry in cluster eks-testing-new for a role whose principal arn is <Principal-ARN-`	

Task	Description	Skills required
	OF-ROLE> with access policy as AmazonEKSAdminPolicy *Expected result*: • The agent should confirm the access entry creation. • To verify, check using the AWS Management Console or use the Amazon EKS API and run the following command: aws eks list-access-entries --cluster-name ekscluster • To describe the access entry that you created, use the following prompt: Describe an access entry in cluster eks-testing-new whose principal arn is <Principal-ARN-OF-ROLE> *Expected result*: • The agent should return details about the access entry. • The details should match what you configured earlier for the access entry.	

Task	Description	Skills required
	• To delete the access entry that you created, use the following prompt: `Delete the access entry in cluster eks-testing-new whose principal arn is <Principal-ARN-OF-ROLE>` *Expected result*: • The agent should confirm the deletion of the access entry. • To verify, check using the AWS Management Console or use the Amazon EKS API and run the following command: `aws eks list-access-entries --cluster-name ekscluster` You can also ask the agent to perform actions for EKS Pod Identity associations. For more details, see [Learn how EKS Pod Identity grants pods access to AWS services]() in the Amazon EKS documentation.	

Clean up

Task	Description	Skills required
Clean up resources.	To clean up the resources that this pattern created, use the following procedure. Wait for each deletion step to complete before proceeding to the next step. **Warning** This procedure will permanently delete all resources created by these stacks. Make sure that you've backed up any important data before proceeding. 1. To delete the Amazon EKS cluster, run the following command: `eksctl delete cluster -f eks-deploy.yaml` **Note** This operation can take 15-20 minutes to complete.	AWS DevOps

Task	Description	Skills required

	2. To delete the Amazon S3 buckets, run the following commands:
	• To empty the Lambda bucket:
	`aws s3 rm s3://bedrock-agent-lambda-artifacts-${AWS_ACCOUNT} --recursive`
	• To empty the knowledge base bucket:
	`aws s3 rm s3://eks-knowledge-base-${AWS_ACCOUNT} –recursive`
	• To empty the OpenAPI schema bucket:
	`aws s3 rm s3://bedrock-agent-openapi-schema-${AWS_ACCOUNT} –recursive`
	• To delete the empty buckets:
	`aws s3 rb s3://bedrock-agent-lambda-artifacts-${AWS_ACCOUNT}`
	`aws s3 rb s3://eks-knowledge-base-${AWS_ACCOUNT}`
	`aws s3 rb s3://bedrock-agent-openapi-`	

Task	Description	Skills required

	```schema-${AWS_ACCOUNT}```
	3. To delete the CloudFormation stack, run the following command:
	```aws cloudformation delete-stack \ --stack-name```
	4. To verify deletion of the Amazon EKS cluster, run the following command:
	```eksctl get clusters```
	5. To verify deletion of the Amazon S3 buckets, run the following commands:
	• To verify deletion of the Lambda bucket:
	```aws s3 ls | grep "bedrock-agent-lambda-artifacts"```
	• To verify deletion of the knowledge base bucket:
	```aws s3 ls | grep "eks-knowledge-base"```
	• To verify deletion of the OpenAPI schema bucket:	

Task	Description	Skills required

	`aws s3 ls | grep "bedrock-agent-openapi-schema"`
	6. To verify stack deletion, run the following command:
	`aws cloudformation list-stacks \--query 'StackSummaries[?StackName==``]'`
	If the stack fails to delete, see Troubleshooting.	

Troubleshooting

Issue	Solution
A non-zero error code is returned during environment setup.	Verify that you're using the correct folder when running any command to deploy this solution. For more information, see the FIRST_DEPLOY.md file in this pattern's repository.
The Lambda function isn't able to do the task.	Make sure that connectivity is set up correctly from the Lambda function to the Amazon EKS cluster.
The agent prompts don't recognize the APIs.	Redeploy the solution. For more information, see the RE_DEPLOY.md file in this pattern's repository.

Issue	Solution
The stack fails to delete.	An initial attempt to delete the stack might fail. This failure can occur because of dependency issues with the custom resource that was created for the OpenSearch collectio n which does the indexing for the knowledge base. To delete the stack, retry the delete operation by retaining the custom resource.

Related resources

AWS Blog

- A deep dive into simplified Amazon EKS access management controls

Amazon Bedrock documentation

- Automate tasks in your application using AI agents
- How Amazon Bedrock Agents works
- Test and troubleshoot agent behavior
- Use action groups to define actions for your agent to perform

Amazon EKS documentation

- Learn how access control works in Amazon EKS

Deploy a RAG use case on AWS by using Terraform and Amazon Bedrock

Martin Maritsch, Nicolas Jacob Baer, Olivier Brique, Julian Ferdinand Grueber, Alice Morano, and Nicola D Orazio, Amazon Web Services

Summary

AWS provides various options to build your Retrieval Augmented Generation (RAG)-enabled generative AI use cases. This pattern provides you with a solution for a RAG-based application based on LangChain and Amazon Aurora PostgreSQL-Compatible as a vector store. You can directly deploy this solution with Terraform into your AWS account and implement the following simple RAG use case:

1. The user manually uploads a file to an Amazon Simple Storage Service (Amazon S3) bucket, such as a Microsoft Excel file or a PDF document. (For more information about supported file types, see the Unstructured documentation.)
2. The content of the file is extracted and embedded into a knowledge database that's based on serverless Aurora PostgreSQL-Compatible, which supports near real-time ingestion of documents into the vector store. This approach enables the RAG model to access and retrieve relevant information for use cases where low latencies matter.
3. When the user engages with the text generation model, it enhances the interaction through retrieval augmentation of relevant content from the previously uploaded files.

The pattern uses Amazon Titan Text Embeddings v2 as the embedding model and Anthropic Claude 3 Sonnet as the text generation model, both available on Amazon Bedrock.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) installed and configured with your AWS account. For installation instructions, see Install or update to the latest version of the AWS CLI in the AWS CLI documentation. To review your AWS credentials and your access to your account, see Configuration and credential file settings in the AWS CLI documentation.

- Model access that's enabled for the required large language models (LLMs) in the Amazon Bedrock console of your AWS account. This pattern requires the following LLMs:
  - `amazon.titan-embed-text-v2:0`
  - `anthropic.claude-3-sonnet-20240229-v1:0`

Limitations

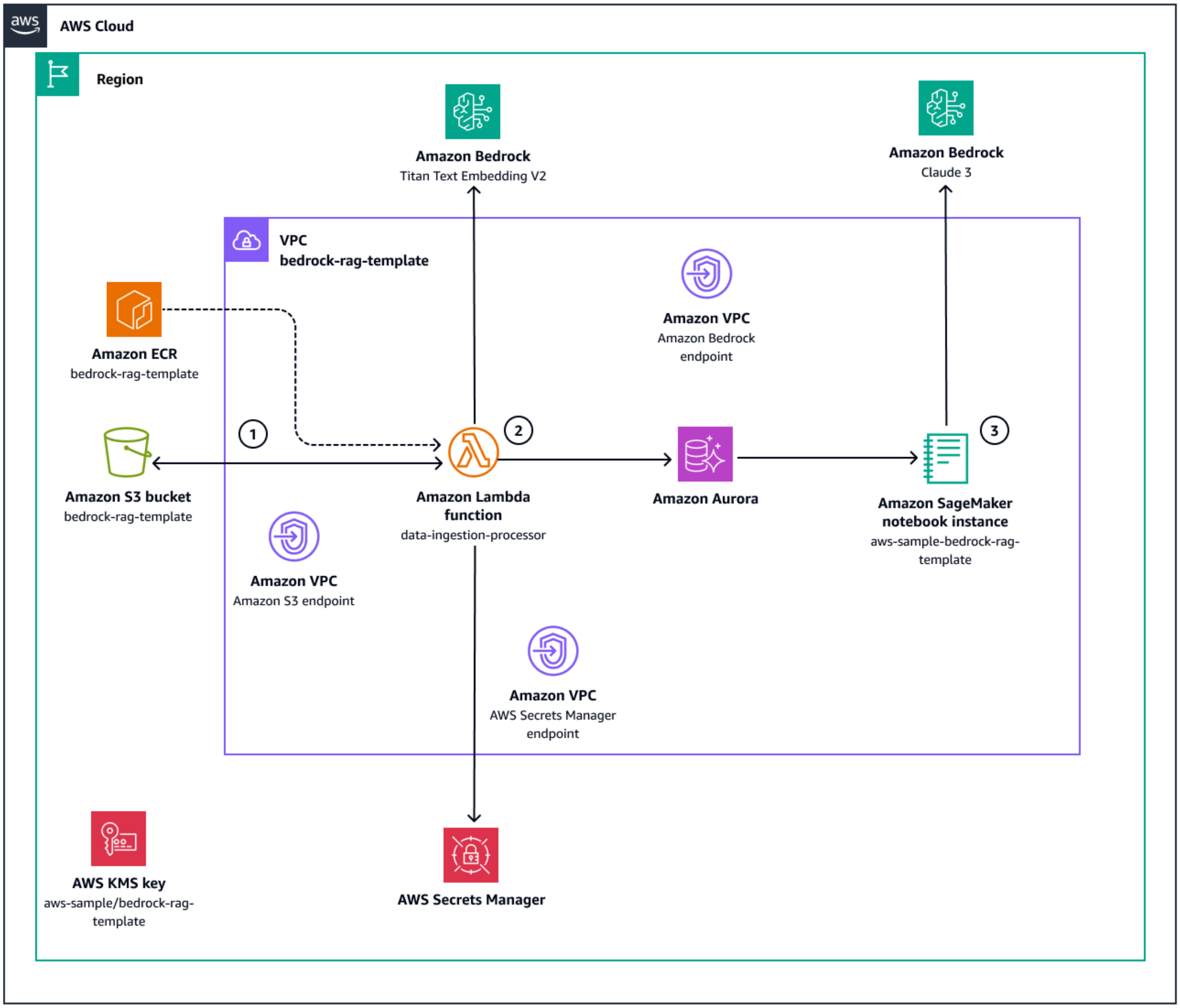
- This sample architecture doesn't include an interface for programmatic question answering with the vector database. If your use case requires an API, consider adding Amazon API Gateway with an AWS Lambda function that runs retrieval and question-answering tasks.
- This sample architecture doesn't include monitoring features for the deployed infrastructure. If your use case requires monitoring, consider adding AWS monitoring services.
- If you upload a lot of documents in a short time frame to the Amazon S3 bucket, the Lambda function might encounter rate limits. As a solution, you can decouple the Lambda function with an Amazon Simple Queue Service (Amazon SQS) queue where you can control the rate of Lambda invocations.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- AWS CLI version 2 or later
- Docker version 26.0.0 or later
- Poetry version 1.7.1 or later
- Python version 3.10 or later
- Terraform version 1.8.4 or later

Architecture

The following diagram shows the workflow and architecture components for this pattern.



This diagram illustrates the following:

1. When an object is created in the Amazon S3 bucket `bedrock-rag-template-<account_id>`, an [Amazon S3 notification](#) invokes the Lambda function `data-ingestion-processor`.
2. The Lambda function `data-ingestion-processor` is based on a Docker image stored in the Amazon Elastic Container Registry (Amazon ECR) repository `bedrock-rag-template`.

   The function uses the LangChain S3FileLoader to read the file as a LangChain Document. Then, the LangChain RecursiveCharacterTextSplitter chunks each document, given a `CHUNK_SIZE` and a `CHUNK_OVERLAP` that depends on the maximum token size of the Amazon Titan Text

Embedding V2 embedding model. Next, the Lambda function invokes the embedding model on Amazon Bedrock to embed the chunks into numerical vector representations. Lastly, these vectors are stored in the Aurora PostgreSQL database. To access the database, the Lambda function first retrieves the username and password from AWS Secrets Manager.

3. On the Amazon SageMaker AI [notebook instance](#) `aws-sample-bedrock-rag-template`, the user can write a question prompt. The code invokes Claude 3 on Amazon Bedrock and adds the knowledge base information to the context of the prompt. As a result, Claude 3 provides responses using the information in the documents.

This pattern's approach to networking and security is as follows:

- The Lambda function `data-ingestion-processor` is in a private subnet within the virtual private cloud (VPC). The Lambda function isn't allowed to send traffic to the public internet because of its security group. As a result, the traffic to Amazon S3 and Amazon Bedrock is routed through the VPC endpoints only. Consequently, the traffic doesn't traverse the public internet, which reduces latency and adds an additional layer of security at the networking level.
- All the resources and data are encrypted whenever applicable by using the AWS Key Management Service (AWS KMS) key with the alias `aws-sample/bedrock-rag-template`.

Automation and scale

This pattern uses Terraform to deploy the infrastructure from the code repository into an AWS account.

Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments. In this pattern, Aurora PostgreSQL-Compatible uses the pgvector plugin as the vector database.
- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command line shell.

- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable. In this pattern, Amazon ECR hosts the Docker image for the `data-ingestion-processor` Lambda function.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use. In this pattern, Lambda ingests data into the vector store.
- Amazon SageMaker AI is a managed machine learning (ML) service that helps you build and train ML models and then deploy them into a production-ready hosted environment.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. The VPC includes subnets and routing tables to control traffic flow.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources.
- Poetry is a tool for dependency management and packaging in Python.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub terraform-rag-template-using-amazon-bedrock repository.

Best practices

- Although this code sample can be deployed into any AWS Region, we recommend that you use US East (N. Virginia) – `us-east-1` or US West (N. California) – `us-west-1`. This recommendation is based on the availability of foundation and embedding models in Amazon Bedrock at the time of this pattern's publication. For an up-to-date list of Amazon Bedrock foundation model support in AWS Regions, see Model support by AWS Region in the Amazon Bedrock documentation. For information about deploying this code sample to other Regions, see Additional information.
- This pattern provides a proof-of-concept (PoC) or pilot demo only. If you want to take the code to production, be sure to use the following best practices:
  - Enable server access logging for Amazon S3.
  - Set up monitoring and alerting for the Lambda function.
  - If your use case requires an API, consider adding Amazon API Gateway with a Lambda function that runs retrieval and question-answering tasks.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Deploy the solution in an AWS account

Task	Description	Skills required
Clone the repository.	To clone the GitHub repository provided with this pattern, use the following command: `git clone https://github.com/aws-samples/terraform-rag-template-using-amazon-bedrock`	AWS DevOps

Task	Description	Skills required
Configure the variables.	To configure the parameters for this pattern, do the following: 1. On your computer, in the GitHub repository, use the following command to open the `terraform` folder: `cd terraform` 2. Open the `commons.tfvars` file, and customize the parameters according to your needs.	AWS DevOps

Task	Description	Skills required

Deploy the solution.	To deploy the solution, do the following:
	
	1. In the `terraform` folder, use the following command to run Terraform and pass in the variables that you customized:
	`terraform init`
	`terraform apply -var-file=commons.tfvars`
	2. Confirm that the resources shown in the architecture diagram were deployed successfully.
	
	The infrastructure deployment provisions an SageMaker AI instance inside the VPC and with the permissions to access the Aurora PostgreSQL database.
		AWS DevOps

Test the solution

Task	Description	Skills required
Run the demo.	After the previous infrastructure deployment has succeeded, use the following steps to run the demo in a Jupyter notebook:	General AWS

Task	Description	Skills required
	1. Sign in to the AWS Management Console of the AWS account where the infrastructure is deployed. 2. Open the SageMaker AI notebook instance `aws-sample-bedrock-rag-template` . 3. Move the `rag_demo.ipynb` Jupyter notebook onto the SageMaker AI notebook instance by using drag and drop. 4. Open the `rag_demo.ipynb` on the SageMaker AI notebook instance, and choose the `conda_python3` kernel. 5. To run the demo, run the cells of the notebook. The Jupyter notebook guides you through the following process: • Installing requirements • Embedding definition • Database connection • Data ingestion • Retrieval augmented text generation • Relevant document queries	

Clean up infrastucture

Task	Description	Skills required
Clean up the infrastructure.	To remove all the resources that you created when they are no longer required, use the following command: `terraform destroy -var-file=commons.tfvars`	AWS DevOps

Related resources

AWS resources

- Building Lambda functions with Python
- Inference parameters for foundation models
- Access to Amazon Bedrock foundation models
- The role of vector databases in generative AI applications (AWS Database Blog)
- Working with Amazon Aurora PostgreSQL

Other resources

- pgvector documentation

Additional information

Implementing a vector database

This pattern uses Aurora PostgreSQL-Compatible to implement a vector database for RAG. As alternatives to Aurora PostgreSQL, AWS provides other capabilities and services for RAG, such as Amazon Bedrock Knowledge Bases and Amazon OpenSearch Service. You can choose the solution that best fits your specific requirements:

- Amazon OpenSearch Service provides distributed search and analytics engines that you can use to store and query large volumes of data.
- Amazon Bedrock Knowledge Bases is designed for building and deploying knowledge bases as an additional abstraction to simplify the RAG ingestion and retrieval process. Amazon Bedrock Knowledge Bases can work with both Aurora PostgreSQL and Amazon OpenSearch Service.

Deploying to other AWS Regions

As described in Architecture, we recommend that you use either the Region US East (N. Virginia) – `us-east-1` or US West (N. California) – `us-west-1` to deploy this code sample. However, there are two possible ways to deploy this code sample to Regions other than `us-east-1` and `us-west-1`. You can configure the deployment Region in the `commons.tfvars` file. For cross-Region foundation model access, consider the following options:

- **Traversing the public internet** – If the traffic can traverse the public internet, add internet gateways to the VPC. Then, adjust the security group assigned to the Lambda function `data-ingestion-processor` and the SageMaker AI notebook instance to allow outbound traffic to the public internet.
- **Not traversing the public internet** – To deploy this sample to any Region other than `us-east-1` or `us-west-1`, do the following:

1. In either the `us-east-1` or `us-west-1` Region, create an additional VPC including a VPC endpoint for `bedrock-runtime`.
2. Create a peering connection by using VPC peering or a transit gateway to the application VPC.
3. When configuring the `bedrock-runtime` boto3 client in any Lambda function outside of `us-east-1` or `us-west-1`, pass the private DNS name of the VPC endpoint for `bedrock-runtime` in `us-east-1` or us-west-1 as the `endpoint_url` to the boto3 client.

Deploy preprocessing logic into an ML model in a single endpoint using an inference pipeline in Amazon SageMaker

Mohan Gowda Purushothama, Gabriel Rodriguez Garcia, and Mateusz Zaremba, Amazon Web Services

Summary

This pattern explains how to deploy multiple pipeline model objects in a single endpoint by using an inference pipeline in Amazon SageMaker. The pipeline model object represents different machine learning (ML) workflow stages, such as preprocessing, model inference, and postprocessing. To illustrate the deployment of serially connected pipeline model objects, this pattern shows you how to deploy a preprocessing Scikit-learn container and a regression model based on the linear learner algorithm built into SageMaker. The deployment is hosted behind a single endpoint in SageMaker.

Note

The deployment in this pattern uses the ml.m4.2xlarge instance type. We recommend using an instance type that aligns with your data size requirements and the complexity of your workflow. For more information, see Amazon SageMaker Pricing. This pattern uses prebuilt Docker images for Scikit-learn, but you can use your own Docker containers and integrate them into your workflow.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Python 3.9
- Amazon SageMaker Python SDK and Boto3 library
- AWS Identity and Access Management (AWS IAM) role with basic SageMaker permissions and Amazon Simple Storage Service (Amazon S3) permissions

Product versions

- Amazon SageMaker Python SDK 2.49.2

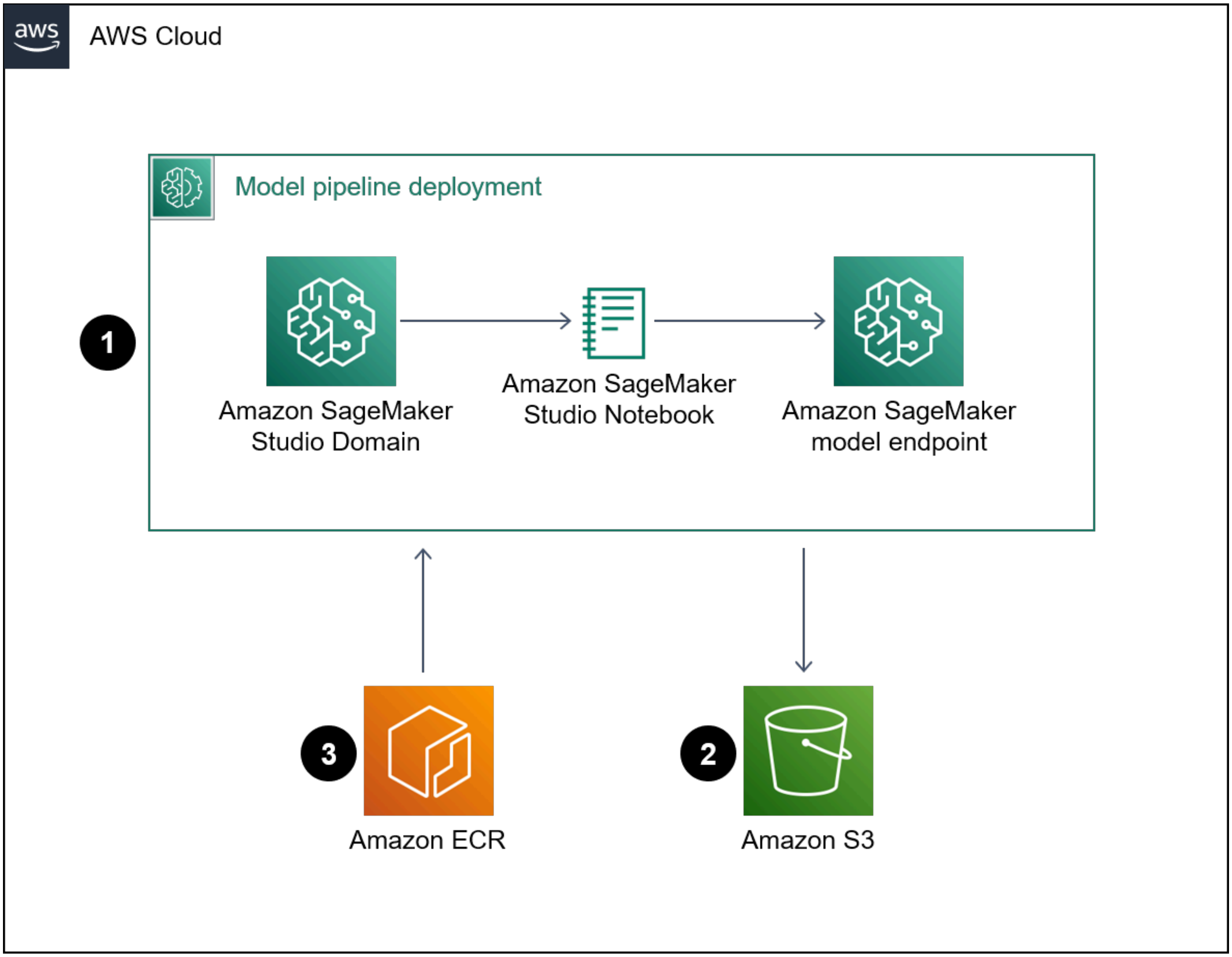
Architecture

Target technology stack

- Amazon Elastic Container Registry (Amazon ECR)
- Amazon SageMaker
- Amazon SageMaker Studio
- Amazon Simple Storage Service (Amazon S3)
- Real-time inference endpoint for Amazon SageMaker

Target architecture

The following diagram shows the architecture for the deployment of an Amazon SageMaker pipeline model object.



The diagram shows the following workflow:

1. A SageMaker notebook deploys a pipeline model.
2. An S3 bucket stores the model artifacts.
3. Amazon ECR gets the source container images from the S3 bucket.

Tools

AWS tools

- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.

- Amazon SageMaker is a managed ML service that helps you build and train ML models and then deploy them into a production-ready hosted environment.
- Amazon SageMaker Studio is a web-based, integrated development environment (IDE) for ML that lets you build, train, debug, deploy, and monitor your ML models.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code

The code for this pattern is available in the GitHub Inference Pipeline with Scikit-learn and Linear Learner repository.

Epics

Prepare the dataset

Task	Description	Skills required

Prepare the dataset for your regression task.	Open a notebook in Amazon SageMaker Studio.
	
	To import all necessary libraries and initialize your working environment, use the following example code in your notebook:
	
	`import sagemaker`
	`from sagemaker import get_execution_role`
	
	`sagemaker_session = sagemaker.Session()`
	
	`# Get a SageMaker-compatible role used by this Notebook Instance.`
		Data scientist

Task	Description	Skills required

	``
	role = get_execu
	tion_role()
	
	# S3 prefix
	bucket = sagemaker
	_session.default_b
	ucket()
	prefix = "Scikit-L
	inearLearner-pipel
	ine-abalone-example"
	``
	
	To download a sample dataset, add the following code to your notebook:
	
	``
	! mkdir abalone_data
	! aws s3 cp s3://sage
	maker-sample-files
	/datasets/tabular/
	uci_abalone/abalon
	e.csv ./abalone_data
	``
	
	Note
	
	The example in this pattern uses the Abalone Data Set from the UCI Machine Learning Repository.	

Task	Description	Skills required

Upload the dataset to an S3 bucket.	In the notebook where you prepared your dataset earlier, add the following code to upload your sample data to an S3 bucket:
	`WORK_DIRECTORY = "abalone_data"`
	
	`train_input = sagemaker_session.upload_data(`
	`    path="{}/{}".format(WORK_DIRECTORY, "abalone.csv"),`
	`    bucket=bucket,`
	`    key_prefix="{}/{}".format(prefix, "train"),`
	`)`
		Data scientist

Create the data preprocessor using SKLearn

Task	Description	Skills required
Prepare the preprocessor.py script.	1. Copy the preprocessing logic from the Python file in the GitHub sklearn_abalone_featurizer.py repository, and then paste the code into a separate Python file called `sklearn_abalone_featurizer.py` . You can modify the code to fit your custom dataset and custom workflow.	Data scientist

Task	Description	Skills required
	2. Save the `sklearn_abalone_featurizer.py` file in the root directory of your project (that is, in the same location where you run your SageMaker notebook).	

Task	Description	Skills required

Create the SKLearn preproces sor object.	To create an SKLearn preprocessor object (called SKLearn Estimator) that you can incorporate into your final inference pipeline, run the following code in your SageMaker notebook:
	
	`from sagemaker.sklearn.estimator import SKLearn`
	
	`FRAMEWORK_VERSION = "0.23-1"`
	`script_path = "sklearn_abalone_featurizer.py"`
	
	`sklearn_preprocessor = SKLearn(`
	`    entry_point=script_path,`
	`    role=role,`
	`    framework_version=FRAMEWORK_VERSION,`
	`    instance_type="ml.c4.xlarge",`
	`    sagemaker_session=sagemaker_session,`
	`)`
	`sklearn_preprocessor.fit({"train": train_input})`
		Data scientist

Task	Description	Skills required

Test the preprocessor's inference.	To confirm that your preprocessor is defined correctly, launch a [batch transform job](https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform.html) by entering the following code in your SageMaker notebook:
	
	`# Define a SKLearn Transformer from the trained SKLearn Estimator`
	`transformer = sklearn_preprocessor.transformer(`
	`    instance_count=1, instance_type="ml.m5.xlarge", assemble_with="Line", accept="text/csv"`
	`)`
	
	`# Preprocess training input`
	`transformer.transform(train_input, content_type="text/csv")`
	`print("Waiting for transform job: " + transformer.latest_transform_job.job_name)`
	`transformer.wait()`
	`preprocessed_train = transformer.output_path`	

Create a machine learning model

Task	Description	Skills required

Create a model object.	To create a model object based on the linear learner algorithm, enter the following code in your SageMaker notebook:
	
	`import boto3`
	`from sagemaker`
	`.image_uris import`
	` retrieve`
	
	`ll_image = retrieve(`
	`"linear-learner",`
	` boto3.Session().re`
	`gion_name)`
	`s3_ll_output_key`
	`_prefix = "ll_train`
	`ing_output"`
	`s3_ll_output_location`
	` = "s3://{}/{}/{}/{}"`
	`.format(`
	`    bucket, prefix,`
	` s3_ll_output_key_p`
	`refix, "ll_model"`
	`)`
	
	`ll_estimator =`
	` sagemaker.estimato`
	`r.Estimator(`
	`    ll_image,`
	`    role,`
	`    instance_count=1,`
	`    instance_type="ml.`
	`m4.2xlarge",`
	`    volume_size=20,`
	`    max_run=3600,`
	`    input_mode="File",`
		Data scientist

Task	Description	Skills required

	``
	    output_path=s3_ll_
	output_location,
	    sagemaker_session=
	sagemaker_session,
	)
	
	ll_estimator.s
	et_hyperparameters
	(feature_dim=10,
	 predictor_type="re
	gressor", mini_batc
	h_size=32)
	
	ll_train_data =
	 sagemaker.inputs.T
	rainingInput(
	    preprocessed_train
	,
	    distribution="Full
	yReplicated",
	    content_type="text
	/csv",
	    s3_data_type="S3Pr
	efix",
	)
	
	data_channels =
	 {"train": ll_train_
	data}
	ll_estimator.fit(inpu
	ts=data_channels,
	 logs=True)
	``
	
	The preceding code retrieves the relevant Amazon ECR Docker image from the public Amazon ECR Registry for the model, creates an estimator object, and then uses that	

Task	Description	Skills required
	object to train the regression model.	

Deploy the final pipeline

Task	Description	Skills required

Deploy the pipeline model.	To create a pipeline model object (that is, a preprocessor object) and deploy the object, enter the following code in your SageMaker notebook:
	
	`from sagemaker.model import Model`
	`from sagemaker.pipeline import PipelineModel`
	`import boto3`
	`from time import gmtime, strftime`
	
	`timestamp_prefix = strftime("%Y-%m-%d-%H-%M-%S", gmtime())`
	
	`scikit_learn_inferencee_model = sklearn_preprocessor.create_model()`
	`linear_learner_model = ll_estimator.create_model()`
	
	`model_name = "inference-pipeline-" + timestamp_prefix`
		Data scientist

Task	Description	Skills required
	``` endpoint_name = "inference-pipeline- ep-" + timestamp_prefix sm_model = PipelineM odel(name=model_name, role=role, models= [scikit_learn_infe rencee_model, linear_learner_model]) sm_model.deploy(init ial_instance_count =1, instance_type="ml. c4.xlarge", endpoint_ name=endpoint_name) ``` **Note** You can adjust the instance type used in the model object to meet your needs.	

Task	Description	Skills required

Test the inference.	To confirm the endpoint is working correctly, run the following sample inference code in your SageMaker notebook:
	`from sagemaker.predictor import Predictor`
	`from sagemaker.serializers import CSVSerializer`
	
	`payload = "M, 0.44, 0.365, 0.125, 0.516, 0.2155, 0.114, 0.155"`
	`actual_rings = 10`
	`predictor = Predictor(`
	`    endpoint_name=endpoint_name, sagemaker_session=sagemaker_session, serializer=CSVSerializer()`
	`)`
	
	`print(predictor.predict(payload))`
		Data scientist

Related resources

- Preprocess input data before making predictions using Amazon SageMaker inference pipelines and Scikit-learn (AWS Machine Learning Blog)
- End to end Machine Learning with Amazon SageMaker (GitHub)

Develop advanced generative AI chat-based assistants by using RAG and ReAct prompting

Praveen Kumar Jeyarajan, Shuai Cao, Noah Hamilton, Kiowa Jackson, Jundong Qiao, and Kara Yang, Amazon Web Services

Summary

A typical corporation has 70 percent of its data trapped in siloed systems. You can use generative AI-powered chat-based assistants to unlock insights and relationships between these data silos through natural language interactions. To get the most out of generative AI, the outputs must be trustworthy, accurate, and inclusive of the available corporate data. Successful chat-based assistants depend on the following:

- Generative AI models (such as Anthropic Claude 2)
- Data source vectorization
- Advanced reasoning techniques, such as the ReAct framework, for prompting the model

This pattern provides data-retrieval approaches from data sources such as Amazon Simple Storage Service (Amazon S3) buckets, AWS Glue, and Amazon Relational Database Service (Amazon RDS). Value is gained from that data by interleaving Retrieval Augmented Generation (RAG) with chain-of-thought methods. The results support complex chat-based assistant conversations that draw on the entirety of your corporation's stored data.

This pattern uses Amazon SageMaker manuals and pricing data tables as an example to explore the capabilities of a generative AI chat-based assistant. You will build a chat-based assistant that helps customers evaluate the SageMaker service by answering questions about pricing and the service's capabilities. The solution uses a Streamlit library for building the frontend application and the LangChain framework for developing the application backend powered by a large language model (LLM).

Inquiries to the chat-based assistant are met with an initial intent classification for routing to one of three possible workflows. The most sophisticated workflow combines general advisory guidance with complex pricing analysis. You can adapt the pattern to suit enterprise, corporate, and industrial use cases.

Prerequisites and limitations

Prerequisites

- AWS Command Line Interface (AWS CLI) installed and configured
- AWS Cloud Development Kit (AWS CDK) Toolkit 2.114.1 or later installed and configured
- Basic familiarity with Python and AWS CDK
- Git installed
- Docker installed
- Python 3.11 or later installed and configured (for more information, see the Tools section)
- An active AWS account bootstrapped by using AWS CDK
- Amazon Titan and Anthropic Claude model access enabled in the Amazon Bedrock service
- AWS security credentials, including `AWS_ACCESS_KEY_ID`, correctly configured in your terminal environment

Limitations

- LangChain doesn't support every LLM for streaming. The Anthropic Claude models are supported, but models from AI21 Labs are not.
- This solution is deployed to a single AWS account.
- This solution can be deployed only in AWS Regions where Amazon Bedrock and Amazon Kendra are available. For information about availability, see the documentation for Amazon Bedrock and Amazon Kendra.

Product versions

- Python version 3.11 or later
- Streamlit version 1.30.0 or later
- Streamlit-chat version 0.1.1 or later
- LangChain version 0.1.12 or later
- AWS CDK version 2.132.1 or later

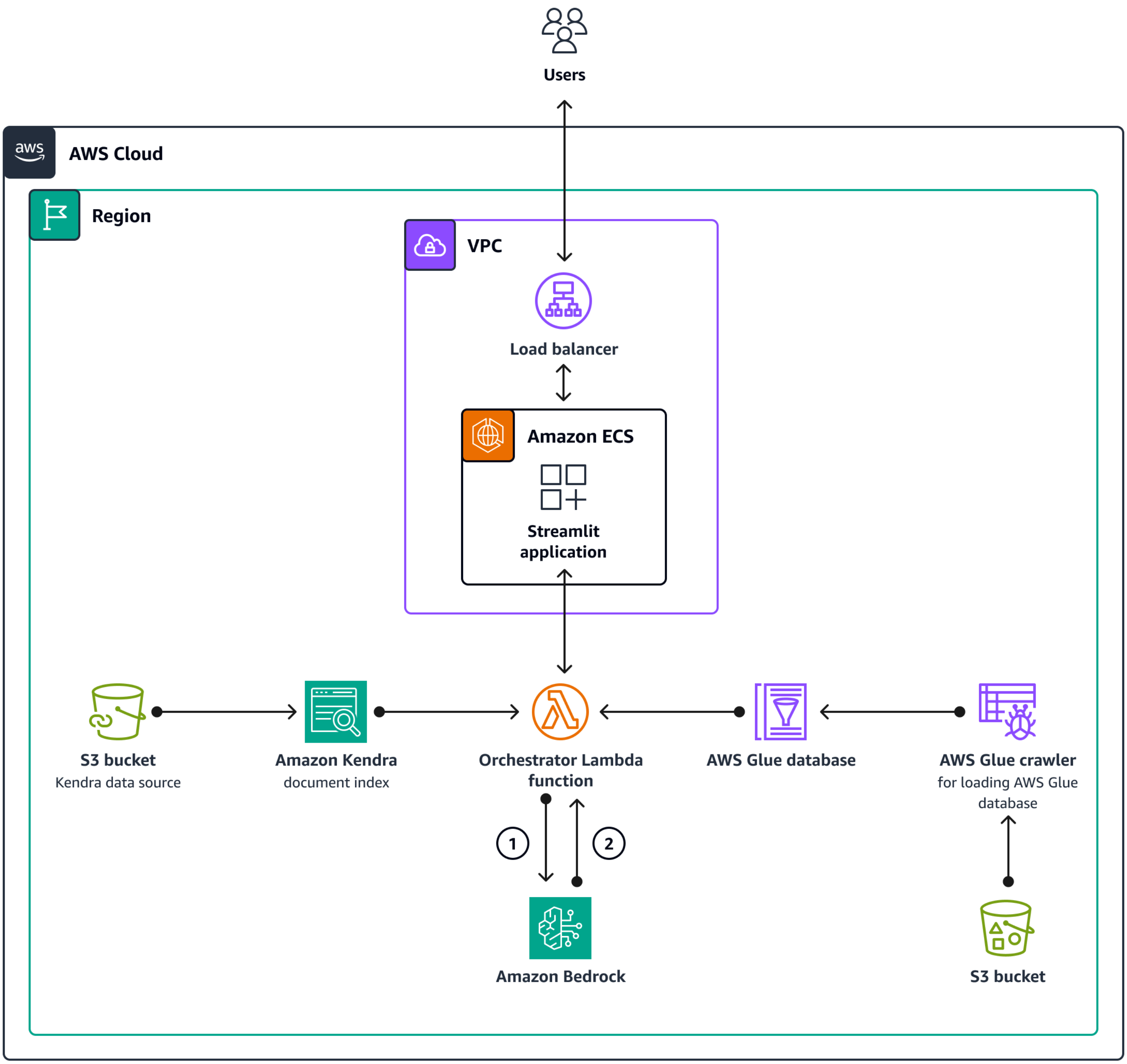
Architecture

Target technology stack

- Amazon Athena
- Amazon Bedrock
- Amazon Elastic Container Service (Amazon ECS)
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon Kendra
- Elastic Load Balancing

Target architecture

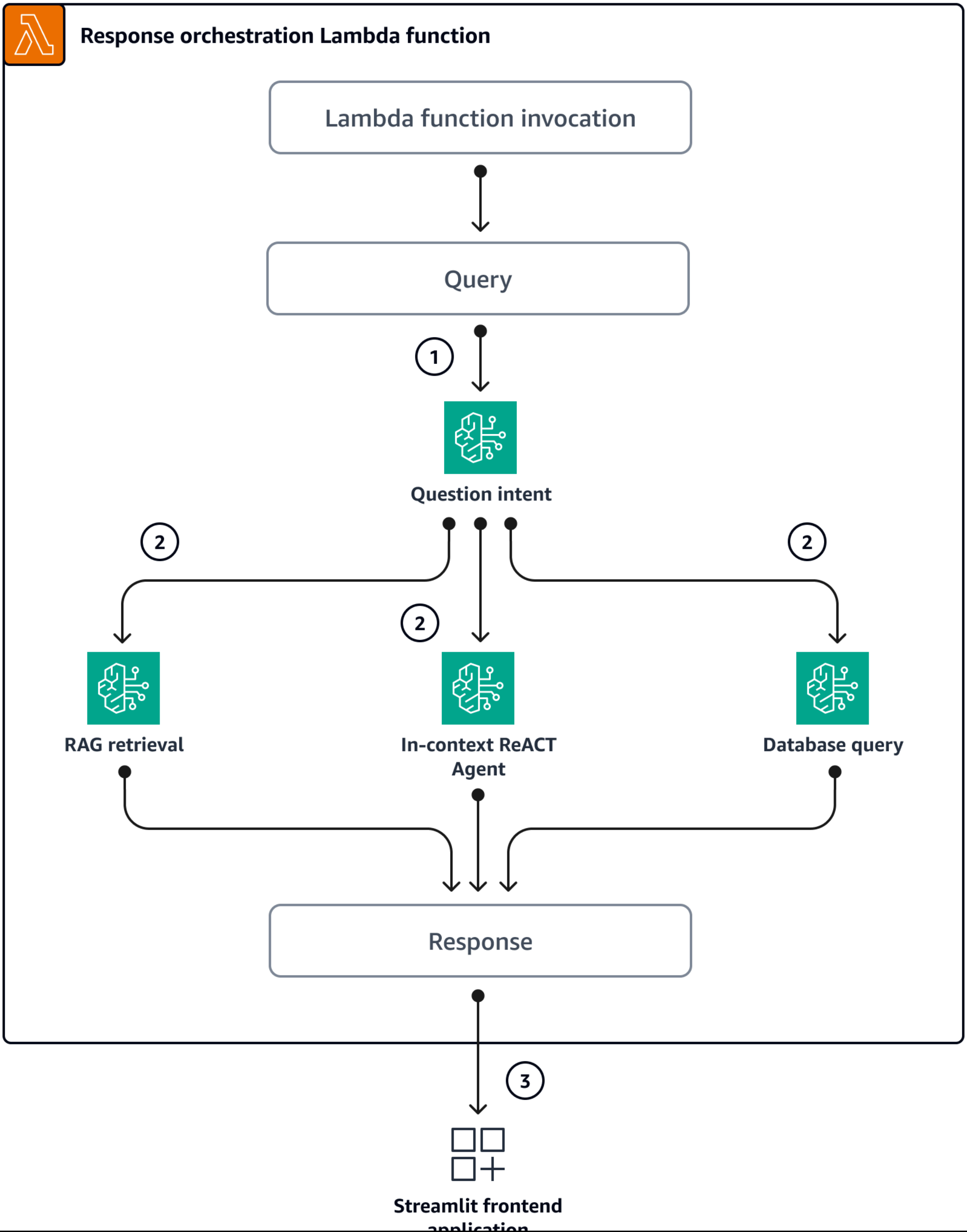
The AWS CDK code will deploy all the resources that are required to set up the chat-based assistant application in an AWS account. The chat-based assistant application shown in the following diagram is designed to answer SageMaker related queries from users. Users connect through an Application Load Balancer to a VPC that contains an Amazon ECS cluster hosting the Streamlit application. An orchestration Lambda function connects to the application. S3 bucket data sources provide data to the Lambda function through Amazon Kendra and AWS Glue. The Lambda function connects to Amazon Bedrock for answering queries (questions) from chat-based assistant users.



1. The orchestration Lambda function sends the LLM prompt request to the Amazon Bedrock model (Claude 2).
2. Amazon Bedrock sends the LLM response back to the orchestration Lambda function.

Logic flow within the orchestration Lambda function

When users ask a question through the Streamlit application, it invokes the orchestration Lambda function directly. The following diagram shows the logic flow when the Lambda function is invoked.


Response orchestration Lambda function
Lambda function invocation
Query
1
Question intent
2
2
2
RAG retrieval
In-context ReACT Agent
Database query
Response
3
Streamlit frontend application

- Step 1 – The input `query` (question) is classified into one of the three intents:
  - General SageMaker guidance questions
  - General SageMaker pricing (training/inference) questions
  - Complex questions related to SageMaker and pricing
- Step 2 – The input `query` initiates one of the three services:
  - `RAG Retrieval service`, which retrieves relevant context from the [Amazon Kendra](#) vector database and calls the LLM through [Amazon Bedrock](#) to summarize the retrieved context as the response.
  - `Database Query service`, which uses- the LLM, database metadata, and sample rows from relevant tables to convert the input `query` into a SQL query. Database Query service runs the SQL query against the SageMaker pricing database through [Amazon Athena](#) and summarizes the query results as the response.
  - `In-context ReACT Agent service`, which breaks down the input `query` into multiple steps before providing a response. The agent uses `RAG Retrieval service` and `Database Query service` as tools to retrieve relevant information during the reasoning process. After the reasoning and actions processes are complete, the agent generates the final answer as the response.
- Step 3 – The response from the orchestration Lambda function is sent to the Streamlit application as output.

Tools

AWS services

- [Amazon Athena](#) is an interactive query service that helps you analyze data directly in Amazon Simple Storage Service (Amazon S3) by using standard SQL.
- [Amazon Bedrock](#) is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API.
- [AWS Cloud Development Kit (AWS CDK)](#) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- [AWS Command Line Interface (AWS CLI)](#) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- [Amazon Elastic Container Service (Amazon ECS)](#) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.

- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams. This pattern uses an AWS Glue crawler and an AWS Glue Data Catalog table.
- Amazon Kendra is an intelligent search service that uses natural language processing and advanced machine learning algorithms to return specific answers to search questions from your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones.

Code repository

The code for this pattern is available in the GitHub genai-bedrock-chatbot repository.

The code repository contains the following files and folders:

- `assets` folder – The static assets the architecture diagram and the public dataset
- `code/lambda-container` folder – The Python code that is run in the Lambda function
- `code/streamlit-app` folder – The Python code that is run as the container image in Amazon ECS
- `tests` folder – The Python files that are run to unit test the AWS CDK constructs
- `code/code_stack.py` – The AWS CDK construct Python files used to create AWS resources
- `app.py` – The AWS CDK stack Python files used to deploy AWS resources in the target AWS account
- `requirements.txt` – The list of all Python dependencies that must be installed for AWS CDK
- `requirements-dev.txt` – The list of all Python dependencies that must be installed for AWS CDK to run the unit-test suite
- `cdk.json` – The input file to provide values required to spin up resources

Note: The AWS CDK code uses L3 (layer 3) constructs and AWS Identity and Access Management (IAM) policies managed by AWS for deploying the solution.

Best practices

- The code example provided here is for a proof-of-concept (PoC) or pilot demo only. If you want to take the code to Production, be sure to use the following best practices:
  - Amazon S3 access logging is enabled.
  - VPC Flow Logs is enabled.
  - The Amazon Kendra Enterprise Edition index is enabled.
- Set up monitoring and alerting for the Lambda function. For more information, see Monitoring and troubleshooting Lambda functions. For general best practices when working with Lambda functions, see the AWS documentation.

Epics

Set up AWS credentials on your local machine

Task	Description	Skills required

Export variables for the account and AWS Region where the stack will be deployed.	To provide AWS credentials for AWS CDK by using environment variables, run the following commands.
	`export CDK_DEFAULT_ACCOUNT=<12 Digit AWS Account Number>`
	`export CDK_DEFAULT_REGION=<region>`
		DevOps engineer, AWS DevOps

Set up the AWS CLI profile.	To set up the AWS CLI profile for the account, follow the instructions in the AWS documentation.	DevOps engineer, AWS DevOps

Set up your environment

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Clone the repo on your local machine.</td><td>To clone the repository, run the following command in your terminal.
<pre>git clone https://github.com/awslabs/genai-bedrock-chatbot.git</pre></td><td>DevOps engineer, AWS DevOps</td></tr>
<tr><td>Set up the Python virtual environment and install required dependencies.</td><td>To set up the Python virtual environment, run the following commands.
<pre>cd genai-bedrock-chatbot
python3 -m venv .venv
source .venv/bin/activate</pre>To set up the required dependencies, run the following command.
<pre>pip3 install -r requirements.txt</pre></td><td>DevOps engineer, AWS DevOps</td></tr>
<tr><td>Set up the AWS CDK environment and synthesize the AWS CDK code.</td><td>1. To set up the AWS CDK environment in your AWS account, run the following command.
<pre>cdk bootstrap aws://ACCOUNT-NUMBER/REGION</pre></td><td>DevOps engineer, AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	2. To convert the code to an AWS CloudFormation stack configuration, run the command `cdk synth`.	

Configure and deploy the chat-based assistant application

Task	Description	Skills required
Provision Claude model access.	To enable Anthropic Claude model access for your AWS account, follow the instructions in the Amazon Bedrock documentation.	AWS DevOps
Deploy resources in the account.	To deploy resources in the AWS account by using the AWS CDK, do the following: 1. In the root of the cloned repository, in the `cdk.json` file, provide inputs for the `logging` parameters. Example values are `INFO`, `DEBUG`, `WARN`, and `ERROR`. These values define log-level messages for the Lambda function and the Streamlit application. 2. The `app.py` file in the root of the cloned repository contains the AWS CloudFormation stack	AWS DevOps, DevOps engineer

Task	Description	Skills required
	name used for deploymen t. The default stack name is `chatbot-stack` . 3. To deploy resources, run the command `cdk deploy`. The `cdk deploy` command uses L3 constructs to create multiple Lambda functions for copying documents and CSV dataset files to S3 buckets. 4. After the command is complete, sign in to the AWS Management Console, open the CloudForm ation console, and [review that the stack deployed successfully](#). Upon successful deploymen t, you can access the chat-based assistant application by using the URL provided in the CloudFormation **Outputs** section.	

Task	Description	Skills required
Run the AWS Glue crawler and create the Data Catalog table.	An [AWS Glue crawler](#) is used to keep the data schema dynamic. The solution creates and updates partitions in the [AWS Glue Data Catalog table](#) by running the crawler on demand. After the CSV dataset files are copied into the S3 bucket, run the AWS Glue crawler and create the Data Catalog table schema for testing: 1. Navigate to the AWS Glue console. 2. In the navigation pane, under **Data Catalog**, choose **Crawlers**. 3. Select the crawler with suffix `sagemaker-pricing-crawler` . 4. Run the crawler. 5. After the crawler runs successfully, it creates an AWS Glue Data Catalog table. **Note** The AWS CDK code configures the AWS Glue crawler to run on demand, but you can	DevOps engineer, AWS DevOps

Task	Description	Skills required
	also [schedule](#) it to run periodically.	
Initiate document indexing.	After the files are copied into the S3 bucket, use Amazon Kendra to crawl and index them: 1. Navigate to the Amazon Kendra console. 2. Select the index with the suffix `chatbot-index` . 3. In the navigation pane, choose **Data sources**, and select the data source connector with the suffix `chatbot-index` . 4. Choose **Sync Now** to initiate the indexing process. **Note** The AWS CDK code configures the Amazon Kendra index sync to run on demand, but you can also run periodically by using the [Schedule parameter](#).	AWS DevOps, DevOps engineer

Clean up all AWS resources in the solution

Task	Description	Skills required
Remove the AWS resources.	After you test the solution, clean up the resources: 1. To remove AWS resources deployed by the solution, run the command `cdk destroy`. 2. Delete all objects from the two S3 buckets, and then remove the buckets. For more information, see Deleting a bucket.	DevOps engineer, AWS DevOps

Troubleshooting

Issue	Solution
AWS CDK returns errors.	For help with AWS CDK issues, see Troubleshooting common AWS CDK issues.

Related resources

- Amazon Bedrock:
  - Model access
  - Inference parameters for foundation models
- Building Lambda functions with Python
- Get started with the AWS CDK
- Working with the AWS CDK in Python
- Generative AI Application Builder on AWS

- LangChain documentation
- Streamlit documentation

Additional information

AWS CDK commands

When working with AWS CDK, keep in mind the following useful commands:

- Lists all stacks in the app

```
cdk ls
```

- Emits the synthesized AWS CloudFormation template

```
cdk synth
```

- Deploys the stack to your default AWS account and Region

```
cdk deploy
```

- Compares the deployed stack with the current state

```
cdk diff
```

- Opens the AWS CDK documentation

```
cdk docs
```

- Deletes the CloudFormation stack and removes AWS deployed resources

```
cdk destroy
```

Develop a fully automated chat-based assistant by using Amazon Bedrock agents and knowledge bases

Jundong Qiao, Shuai Cao, Noah Hamilton, Kiowa Jackson, Praveen Kumar Jeyarajan, and Kara Yang, Amazon Web Services

Summary

Many organizations face challenges when creating a chat-based assistant that is capable of orchestrating diverse data sources to offer comprehensive answers. This pattern presents a solution for developing a chat-based assistant that is capable of answering queries from both documentation and databases, with a straightforward deployment.

Starting with Amazon Bedrock, this fully managed generative artificial intelligence (AI) service provides a wide array of advanced foundation models (FMs). This facilitates the efficient creation of generative AI applications with a strong focus on privacy and security. In the context of documentation retrieval, the Retrieval Augmented Generation (RAG) is a pivotal feature. It uses knowledge bases to augment FM prompts with contextually relevant information from external sources. An Amazon OpenSearch Serverless index serves as the vector database behind the knowledge bases for Amazon Bedrock. This integration is enhanced through careful prompt engineering to minimize inaccuracies and make sure that responses are anchored in factual documentation. For database queries, the FMs of Amazon Bedrock transform textual inquiries into structured SQL queries, incorporating specific parameters. This enables the precise retrieval of data from databases managed by AWS Glue databases. Amazon Athena is used for these queries.

For handling more intricate queries, achieving comprehensive answers demands information sourced from both documentation and databases. Agents for Amazon Bedrock is a generative AI feature that helps you build autonomous agents that can understand complex tasks and break them down into simpler tasks for orchestration. The combination of insights retrieved from the simplified tasks, facilitated by Amazon Bedrock autonomous agents, enhances the synthesis of information, leading to more thorough and exhaustive answers. This pattern demonstrates how to build a chat-based assistant by using Amazon Bedrock and the related generative AI services and features within an automated solution.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Docker, installed
- AWS Cloud Development Kit (AWS CDK), installed and bootstrapped to the `us-east-1` or `us-west-2` AWS Regions
- AWS CDK Toolkit version 2.114.1 or later, installed
- AWS Command Line Interface (AWS CLI), installed and configured
- Python version 3.11 or later, installed
- In Amazon Bedrock, enable access to Claude 2, Claude 2.1, Claude Instant, and Titan Embeddings G1 – Text

Limitations

- This solution is deployed to a single AWS account.
- This solution can be deployed only in AWS Regions where Amazon Bedrock and Amazon OpenSearch Serverless are supported. For more information, see the documentation for Amazon Bedrock and Amazon OpenSearch Serverless.

Product versions

- Llama-index version 0.10.6 or later
- Sqlalchemy version 2.0.23 or later
- Opensearch-py version 2.4.2 or later
- Requests_aws4auth version 1.2.3 or later
- AWS SDK for Python (Boto3) version 1.34.57 or later

Architecture

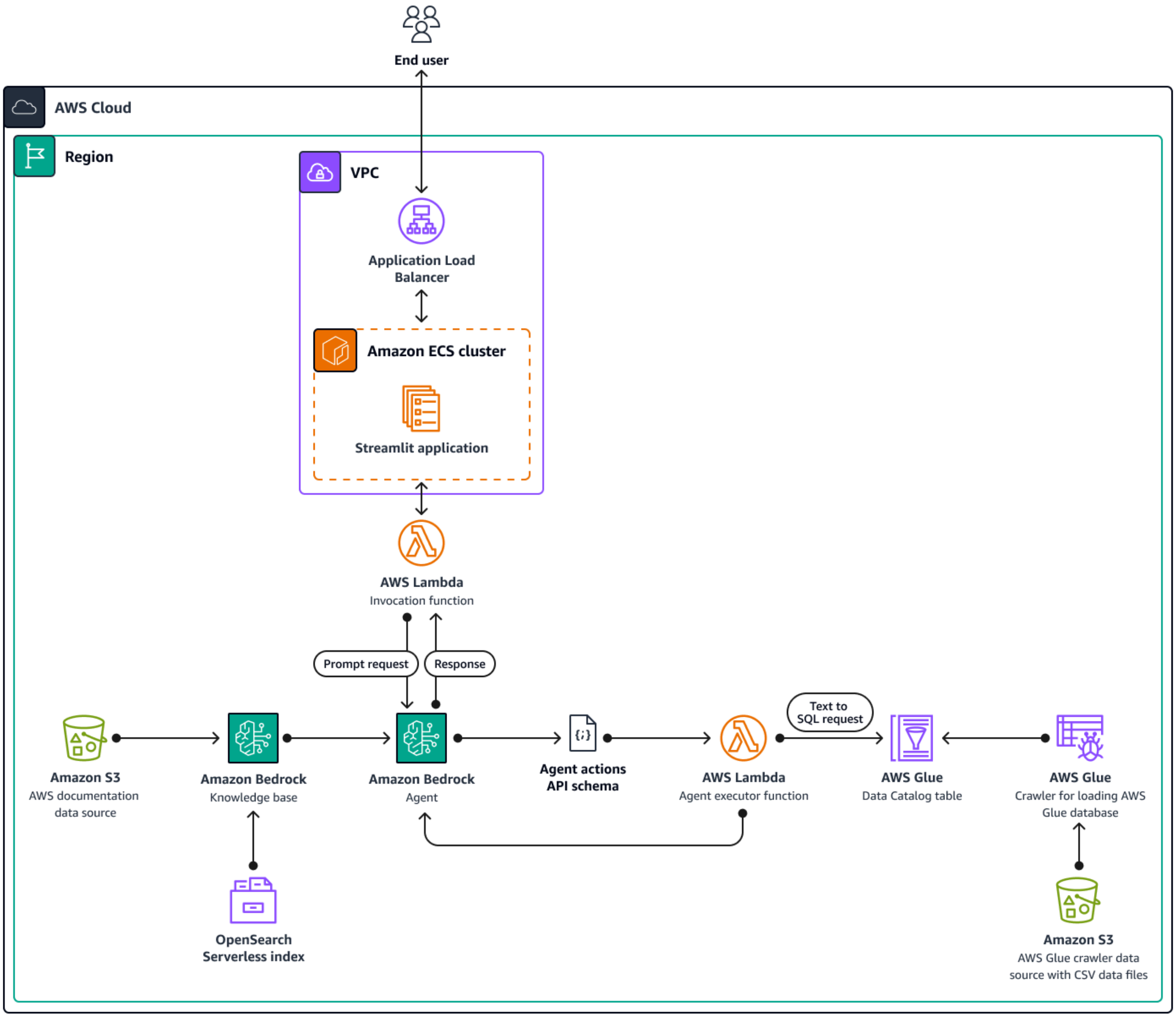
Target technology stack

The AWS Cloud Development Kit (AWS CDK) is an open source software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation. The AWS CDK stack used in this pattern deploys the following AWS resources:

- AWS Key Management Service (AWS KMS)

- Amazon Simple Storage Service (Amazon S3)
- AWS Glue Data Catalog, for the AWS Glue database component
- AWS Lambda
- AWS Identity and Access Management (IAM)
- Amazon OpenSearch Serverless
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon Elastic Container Service (Amazon ECS)
- AWS Fargate
- Amazon Virtual Private Cloud (Amazon VPC)
- Application Load Balancer

Target architecture



The diagram shows a comprehensive AWS cloud-native setup within a single AWS Region, using multiple AWS services. The primary interface for the chat-based assistant is a [Streamlit](https://streamlit.io/) application hosted on an Amazon ECS cluster. An [Application Load Balancer](https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html) manages accessibility. Queries made through this interface activate the `Invocation` Lambda function, which then interfaces with agents for Amazon Bedrock. This agent responds to user inquiries by either consulting the knowledge bases for Amazon Bedrock or by invoking an `Agent executor` Lambda function. This function triggers a set of actions associated with the agent, following a predefined API schema. The knowledge bases for Amazon Bedrock use an OpenSearch Serverless index as their vector database foundation. Additionally, the `Agent executor` function generates SQL queries that are executed against the AWS Glue database through Amazon Athena.

Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon Simple Storage Service (Amazon S3) by using standard SQL.
- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams. This pattern uses an AWS Glue crawler and an AWS Glue Data Catalog table.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon OpenSearch Serverless is an on-demand serverless configuration for Amazon OpenSearch Service. In this pattern, an OpenSearch Serverless index serves as a vector database for the knowledge bases for Amazon Bedrock.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Streamlit is an open source Python framework to create data applications.

Code repository

The code for this pattern is available in the GitHub genai-bedrock-agent-chatbot repository. The code repository contains the following files and folders:

- `assets` folder – The static assets, such as the architecture diagram and the public dataset.
- `code/lambdas/action-lambda` folder – The Python code for the Lambda function that acts as an action for the Amazon Bedrock agent.
- `code/lambdas/create-index-lambda` folder – The Python code for the Lambda function that creates the OpenSearch Serverless index.
- `code/lambdas/invoke-lambda` folder – The Python code for the Lambda function that invokes the Amazon Bedrock agent, which is called directly from the Streamlit application.
- `code/lambdas/update-lambda` folder – The Python code for the Lambda function that updates or deletes resources after the AWS resources are deployed through the AWS CDK.
- `code/layers/boto3_layer` folder – The AWS CDK stack that creates a Boto3 layer that is shared across all Lambda functions.
- `code/layers/opensearch_layer` folder – The AWS CDK stack that creates an OpenSearch Serverless layer that installs all dependencies to create the index.
- `code/streamlit-app` folder – The Python code that is run as the container image in Amazon ECS
- `code/code_stack.py` – The AWS CDK construct Python files that create AWS resources.
- `app.py` – The AWS CDK stack Python files that deploy AWS resources in the target AWS account.
- `requirements.txt` – The list of all Python dependencies that must be installed for the AWS CDK.
- `cdk.json` – The input file to provide the values that are required to create resources. Also, in the `context/config` fields, you can customize the solution accordingly. For more information about customization, see the Additional information section.

Best practices

- The code example provided here is for proof-of-concept (PoC) or pilot purposes only. If you want to take the code to production, be sure to use the following best practices:
  - Enable Amazon S3 access logging
  - Enable VPC Flow Logs

- Set up monitoring and alerting for the Lambda functions. For more information, see Monitoring and troubleshooting Lambda functions. For best practices, see the Best practices for working with AWS Lambda functions.

Epics

Set up AWS credentials on your local workstation

Task	Description	Skills required

Export variables for the account and Region.	To provide AWS credentials for the AWS CDK by using environment variables, run the following commands.
	`export CDK_DEFAULT_ACCOUNT=<12-digit AWS account number>`
	`export CDK_DEFAULT_REGION=<Region>`
		AWS DevOps, DevOps engineer

Set up the AWS CLI named profile.	To set up the AWS CLI named profile for the account, follow the instructions in Configuration and credential file settings.	AWS DevOps, DevOps engineer

Set up your environment

Task	Description	Skills required
Clone the repo to your local workstation.	To clone the repository, run the following command in your terminal. `git clone https://github.com/awslabs/`	DevOps engineer, AWS DevOps

Task	Description	Skills required
	`genai-bedrock-agent-chatbot.git`	

Set up the Python virtual environment.	To set up the Python virtual environment, run the following commands.
	`cd genai-bedrock-agent-chatbot`
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	To set up the required dependencies, run the following command.
	`pip3 install -r requirements.txt`
		DevOps engineer, AWS DevOps

Set up the AWS CDK environment.	To convert the code to an AWS CloudFormation template, run the command `cdk synth`.	AWS DevOps, DevOps engineer

Configure and deploy the application

Task	Description	Skills required
Deploy resources in the account.	To deploy resources in the AWS account by using the AWS CDK, do the following: 1. In the root of the cloned repository, in the `cdk.json` file, provide	DevOps engineer, AWS DevOps

Task	Description	Skills required
	inputs for the logging parameters. Example values are `INFO`, `DEBUG`, `WARN`, and `ERROR`. These values define log-level messages for the Lambda functions and the Streamlit application. 2. The `cdk.json` file in the root of the cloned repository contains the AWS CloudFormation stack name used for deploymen t. The default stack name is `chatbot-stack` . The default Amazon Bedrock agent name is `ChatbotBedrockAgent` , and the default Amazon Bedrock agent alias is `Chatbot_Agent` . 3. To deploy resources, run the command `cdk deploy`. The `cdk deploy` command uses layer-3 constructs to create multiple Lambda functions for copying documents and CSV dataset files to S3 buckets. It also deploys the Amazon Bedrock agent, knowledge bases, and	

Task	Description	Skills required
	`Action group` Lambda function for the Amazon Bedrock agent. 4. Sign in to the AWS Management Console, and then open the CloudFormation console at https://console.aws.amazon.com/cloudformation/. 5. Confirm that the stack deployed successfully. For instructions, see Reviewing your stack on the AWS CloudFormation console. After successful deployment, you can access the chat-based assistant application by using the URL provided on the **Outputs** tab in the CloudFormation console.	

Clean up all AWS resources in the solution

Task	Description	Skills required
Remove the AWS resources.	After you test the solution, to clean up the resources, run the command `cdk destroy`.	AWS DevOps, DevOps engineer

Related resources

AWS documentation

- Amazon Bedrock resources:
  - Model access
  - Inference parameters for foundation models
  - Agents for Amazon Bedrock
  - Knowledge bases for Amazon Bedrock
- Building Lambda functions with Python
- AWS CDK resources:
  - Get started with the AWS CDK
  - Troubleshooting common AWS CDK issues
  - Working with the AWS CDK in Python
- Generative AI Application Builder on AWS

Other AWS resources

- Vector Engine for Amazon OpenSearch Serverless

Other resources

- LlamaIndex documentation
- Streamlit documentation

Additional information

Customize the chat-based assistant with your own data

To integrate your custom data for deploying the solution, follow these structured guidelines. These steps are designed to ensure a seamless and efficient integration process, enabling you to deploy the solution effectively with your bespoke data.

For knowledge base data integration

Data preparation

1. Locate the `assets/knowledgebase_data_source/` directory.
2. Place your dataset within this folder.

Configuration adjustments

1. Open the `cdk.json` file.
2. Navigate to the `context/configure/paths/knowledgebase_file_name` field, and then update it accordingly.
3. Navigate to the `bedrock_instructions/knowledgebase_instruction` field, and then update it to accurately reflect the nuances and context of your new dataset.

For structural data integration

Data organization

1. Within the `assets/data_query_data_source/` directory, create a subdirectory, such as `tabular_data`.
2. Put your structured dataset (acceptable formats include CSV, JSON, ORC, and Parquet) into this newly created subfolder.
3. If you are connecting to an existing database, update the function `create_sql_engine()` in `code/lambda/action-lambda/build_query_engine.py` to connect to your database.

Configuration and code updates

1. In the `cdk.json` file, update the `context/configure/paths/athena_table_data_prefix` field to align with the new data path.
2. Revise `code/lambda/action-lambda/dynamic_examples.csv` by incorporating new text-to-SQL examples that correspond with your dataset.
3. Revise `code/lambda/action-lambda/prompt_templates.py` to mirror the attributes of your structured dataset.
4. In the `cdk.json` file, update the `context/configure/bedrock_instructions/action_group_description` field to explain the purpose and functionality of the `Action group` Lambda function.
5. In the `assets/agent_api_schema/artifacts_schema.json` file, explain the new functionalities of your `Action group` Lambda function.

General update

In the `cdk.json` file, in the `context/configure/bedrock_instructions/agent_instruction` section, provide a comprehensive description of the Amazon Bedrock agent's intended functionality and design purpose, taking into account the newly integrated data.

Document institutional knowledge from voice inputs by using Amazon Bedrock and Amazon Transcribe

Praveen Kumar Jeyarajan, Jundong Qiao, Rajiv Upadhyay, and Megan Wu, Amazon Web Services

Summary

Capturing institutional knowledge is paramount for ensuring organizational success and resilience. Institutional knowledge represents the collective wisdom, insights, and experiences accumulated by employees over time, often tacit in nature and passed down informally. This wealth of information encompasses unique approaches, best practices, and solutions to intricate problems that might not be documented elsewhere. By formalizing and documenting this knowledge, companies can preserve institutional memory, foster innovation, enhance decision-making processes, and accelerate learning curves for new employees. Additionally, it promotes collaboration, empowers individuals, and cultivates a culture of continuous improvement. Ultimately, harnessing institutional knowledge helps companies use their most valuable asset—the collective intelligence of their workforce—to navigate challenges, drive growth, and maintain competitive advantage in dynamic business environments.

This pattern explains how to capture institutional knowledge through voice recordings from senior employees. It uses Amazon Transcribe and Amazon Bedrock for systematic documentation and verification. By documenting this informal knowledge, you can preserve it and share it with subsequent cohorts of employees. This endeavor supports operational excellence and improves the effectiveness of training programs through the incorporation of practical knowledge acquired through direct experience.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Docker, installed
- AWS Cloud Development Kit (AWS CDK) version 2.114.1 or later, installed and bootstrapped to the `us-east-1` or `us-west-2` AWS Regions
- AWS CDK Toolkit version 2.114.1 or later, installed
- AWS Command Line Interface (AWS CLI), installed and configured

- Python version 3.12 or later, installed
- Permissions to create Amazon Transcribe, Amazon Bedrock, Amazon Simple Storage Service (Amazon S3), and AWS Lambda resources

Limitations

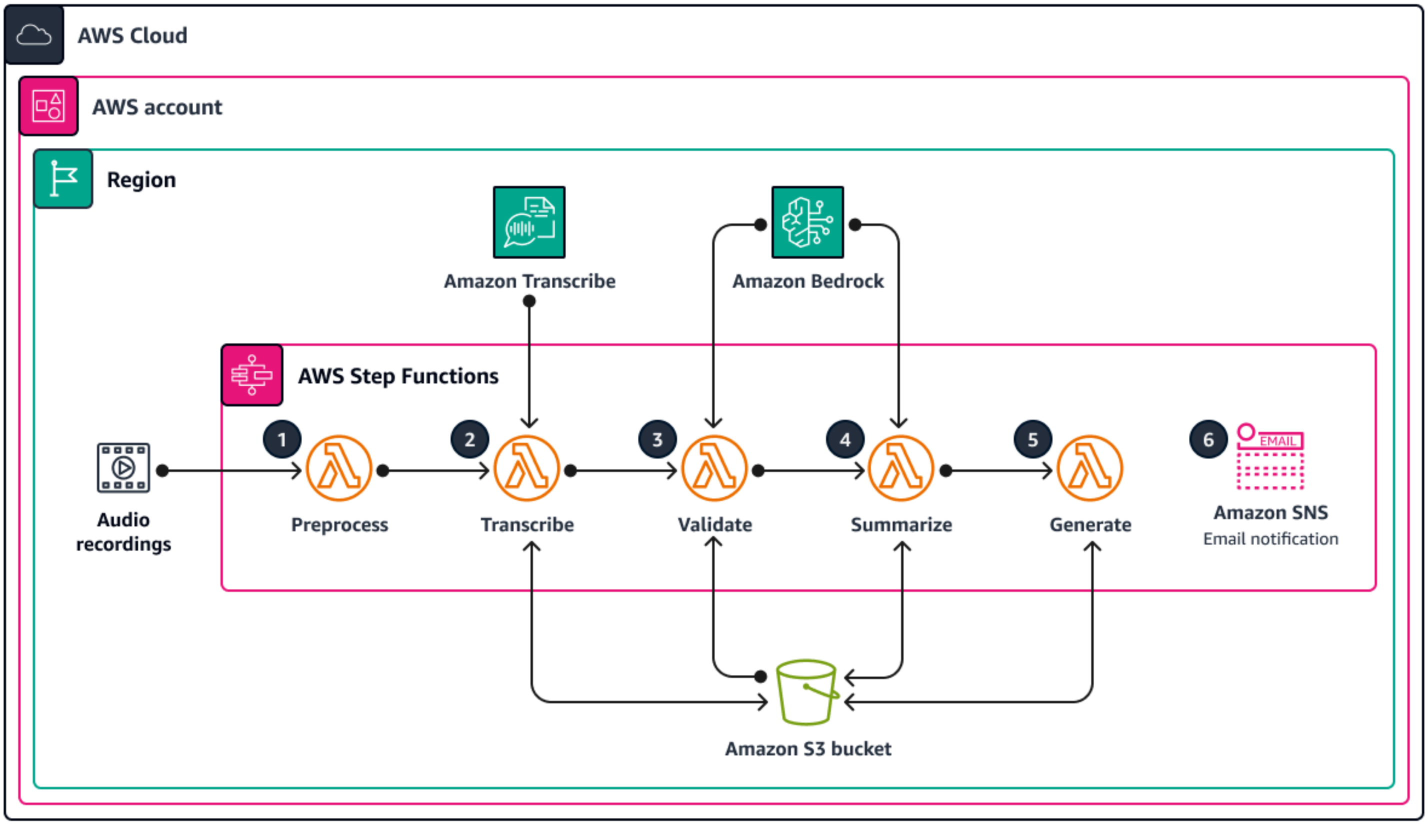
- This solution is deployed to a single AWS account.
- This solution can be deployed only in AWS Regions where Amazon Bedrock and Amazon Transcribe are available. For information about availability, see the documentation for Amazon Bedrock and Amazon Transcribe.
- The audio files must be in a format that Amazon Transcribe supports. For a list of supported formats, see Media formats in the Transcribe documentation.

Product versions

- AWS SDK for Python (Boto3) version 1.34.57 or later
- LangChain version 0.1.12 or later

Architecture

The architecture represents a serverless workflow on AWS. AWS Step Functions orchestrates Lambda functions for audio processing, text analysis, and document generation. The following diagram shows the Step Functions workflow, also known as a *state machine*.



Each step in the state machine is handled by a distinct Lambda function. The following are the steps in the document generation process:

1. The `preprocess` Lambda function validates the input passed to Step Functions and lists all of the audio files present in the provided Amazon S3 URI folder path. Downstream Lambda functions in the workflow use the file list to validate, summarize, and generate the document.
2. The `transcribe` Lambda function uses Amazon Transcribe to convert audio files into text transcripts. This Lambda function is responsible for initiating the transcription process and accurately transforming speech into text, which is then stored for subsequent processing.
3. The `validate` Lambda function analyzes the text transcripts, determining the relevance of the responses to the initial questions. By using a large language model (LLM) through Amazon Bedrock, it identifies and separates on-topic answers from off-topic responses.
4. The `summarize` Lambda function uses Amazon Bedrock to generate a coherent and concise summary of the on-topic answers.
5. The `generate` Lambda function assembles the summaries into a well-structured document. It can format the document according to predefined templates and include any additional necessary content or data.

6. If any of the Lambda functions fail, you receive an email notification through Amazon Simple Notification Service (Amazon SNS).

Throughout this process, AWS Step Functions makes sure that each Lambda function is initiated in the correct sequence. This state machine has the capacity for parallel processing to enhance efficiency. An Amazon S3 bucket acts as the central storage repository, supporting the workflow by managing the various media and document formats involved.

Tools

AWS services

- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- Amazon Transcribe is an automatic speech recognition service that uses machine learning models to convert audio to text.

Other tools

- LangChain is a framework for developing applications that are powered by large language models (LLMs).

Code repository

The code for this pattern is available in the GitHub genai-knowledge-capture repository.

The code repository contains the following files and folders:

- `assets` folder – The static assets for the solution, such as the architecture diagram and the public dataset
- `code/lambdas` folder – The Python code for all Lambda functions
  - `code/lambdas/generate` folder - The Python code that generates a document from the summarized data in the S3 bucket
  - `code/lambdas/preprocess` folder - The Python code that processes the inputs for the Step Functions state machine
  - `code/lambdas/summarize` folder - The Python code that summarizes the transcribed data by using Amazon Bedrock service
  - `code/lambdas/transcribe` folder - The Python code that converts speech data (audio file) into text by using Amazon Transcribe
  - `code/lambdas/validate` folder - The Python code that validates whether all answers pertain to the same topic
- `code/code_stack.py` – The AWS CDK construct Python file that is used to create AWS resources
- `app.py` – The AWS CDK app Python file that is used to deploy AWS resources in the target AWS account
- `requirements.txt` – The list of all Python dependencies that must be installed for the AWS CDK
- `cdk.json` – The input file to provide values that are required to create resources

Best practices

The code example provided is for proof-of-concept (PoC) or pilot purposes only. If you want to take the solution to production, use the following best practices:

- Enable [Amazon S3 access logging](#)
- Enable [VPC Flow Logs](#)

Epics

Set up AWS credentials on your local workstation

Task	Description	Skills required

Export variables for the account and AWS Region.	To provide AWS credentials for the AWS CDK by using environment variables, run the following commands.
	`export CDK_DEFAULT_ACCOUNT=<12-digit AWS account number>`
	`export CDK_DEFAULT_REGION=<Region>`
		AWS DevOps, DevOps engineer

Set up the AWS CLI named profile.	To set up the AWS CLI named profile for the account, follow the instructions in Configuration and credential file settings.	AWS DevOps, DevOps engineer

Set up your environment

Task	Description	Skills required
Clone the repo to your local workstation.	To clone the genai-knowledge-capture repository, run the following command in your terminal. `git clone https://github.com/aws-samples/genai-knowledge-capture`	AWS DevOps, DevOps engineer

Task	Description	Skills required
(Optional) Replace the audio files.	To customize the sample application to incorpora te your own data, do the following: 1. Navigate to the `assets/audio_samples` folder in the cloned repository. 2. Delete the folders containing the sample audio files. 3. Create a folder for each topic you want to analyze. 4. Transfer your audio files to their respective folders.	AWS DevOps, DevOps engineer

Set up the Python virtual environment.	To set up the Python virtual environment, run the following commands.
	
	`cd genai-knowledge-capture`
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	`pip install -r requirements.txt`
		AWS DevOps, DevOps engineer

Synthesize the AWS CDK code.	To convert the code to an AWS CloudFormation stack configuration, run the following command. `cdk synth`	AWS DevOps, DevOps engineer

Configure and deploy the solution

Task	Description	Skills required
Provision foundation model access.	Enable access to the Anthropic Claude 3 Sonnet model for your AWS account. For instructions, see Add model access in the Bedrock documentation.	AWS DevOps
Deploy resources in the account.	To deploy resources in the AWS account by using the AWS CDK, do the following: 1. (Optional) In the root of the cloned repository, in the `app.py` file, update the AWS CloudFormation stack name. The default stack name is `genai-knowledge-capture-stack` . 2. To deploy resources, run the command `cdk deploy`. The `cdk deploy` command uses layer-3 constructs to create a set of Lambda functions, an S3 bucket, an Amazon SNS topic, and a Step Functions state machine. Audio files in the `assets/audio_samples` folder are	AWS DevOps, DevOps engineer

Task	Description	Skills required
	copied into the S3 bucket during deployment. 3. Sign in to the AWS Management Console, and then open the CloudFormation console at https://console.aws.amazon.com/cloudformation/. 4. Confirm that the stack deployed successfully. For instructions, see Reviewing your stack on the AWS CloudFormation console.	
Subscribe to the Amazon SNS topic.	To subscribe to the Amazon SNS topic for notification, do the following: 1. In the CloudFormation console, in the navigate pane, choose **Stacks**. 2. Choose the `genai-knowledge-capture-stack` stack. 3. Choose the **Outputs** tab. 4. Find the Amazon SNS topic name with the key `SNSTopicName` . 5. Configure an email address to receive notifications by following the instructions in Subscribe an email address to an Amazon SNS topic.	General AWS

Test the solution

Task	Description	Skills required
Run the state machine.	1. Open the [Step Functions console](). 2. On the **State machines** page, choose **genai-knowledge-capture-stack-state-machine**. 3. Choose **Start execution**. 4. (Optional) In the **Name** box, enter a name for the execution. 5. In the **Input** area, enter the following JSON object by replacing the placeholder text, where: • `<Name>` is what you want to name the document. • `<S3 bucket name>` is the name of the Amazon S3 bucket that contains the audio files. • `<Folder path>` is the directory that contains the audio files. `{ "documentName": "<Name>", "audioFileFolderUri": "s3://<S3 bucket name>/<Folder path>" }`	App developer, General AWS

Task	Description	Skills required
	6. Choose **Start Execution**. 7. On the execution details page, review the results and wait for the execution to complete.	

Clean up all AWS resources in the solution

Task	Description	Skills required
Remove the AWS resources.	After you test the solution, clean up the resources: 1. Delete all objects from the S3 bucket, and then delete the bucket. For more information, see [Deleting a bucket](#). 2. From the cloned repository, run the command `cdk destroy`.	AWS DevOps, DevOps engineer

Related resources

AWS documentation

- Amazon Bedrock resources:
  - Model access
  - Inference parameters for foundation models
- AWS CDK resources:
  - Get started with the AWS CDK
  - Working with the AWS CDK in Python
  - Troubleshooting common AWS CDK issues

  - Toolkit commands
- AWS Step Functions resources:
  - Getting started with AWS Step Functions
  - Troubleshooting
- Building Lambda functions with Python
- Generative AI Application Builder on AWS

Other resources

- LangChain documentation

Generate personalized and re-ranked recommendations using Amazon Personalize

Mason Cahill, Matthew Chasse, and Tayo Olajide, Amazon Web Services

Summary

This pattern shows you how to use Amazon Personalize to generate personalized recommendations—including re-ranked recommendations—for your users based on the ingestion of real-time user-interaction data from those users. The example scenario used in this pattern is based on a pet adoption website that generates recommendations for its users based on their interactions (for example, what pets a user visits). By following the example scenario, you learn to use Amazon Kinesis Data Streams to ingest interaction data, AWS Lambda to generate recommendations and re-rank the recommendations, and Amazon Data Firehose to store the data in an Amazon Simple Storage Service (Amazon S3) bucket. You also learn to use AWS Step Functions to build a state machine that manages the solution version (that is, a trained model) that generates your recommendations.

Prerequisites and limitations

Prerequisites

- An active AWS account with a bootstrapped AWS Cloud Development Kit (AWS CDK)
- AWS Command Line Interface (AWS CLI) with configured credentials
- Python 3.9

Product versions

- Python 3.9
- AWS CDK 2.23.0 or later
- AWS CLI 2.7.27 or later

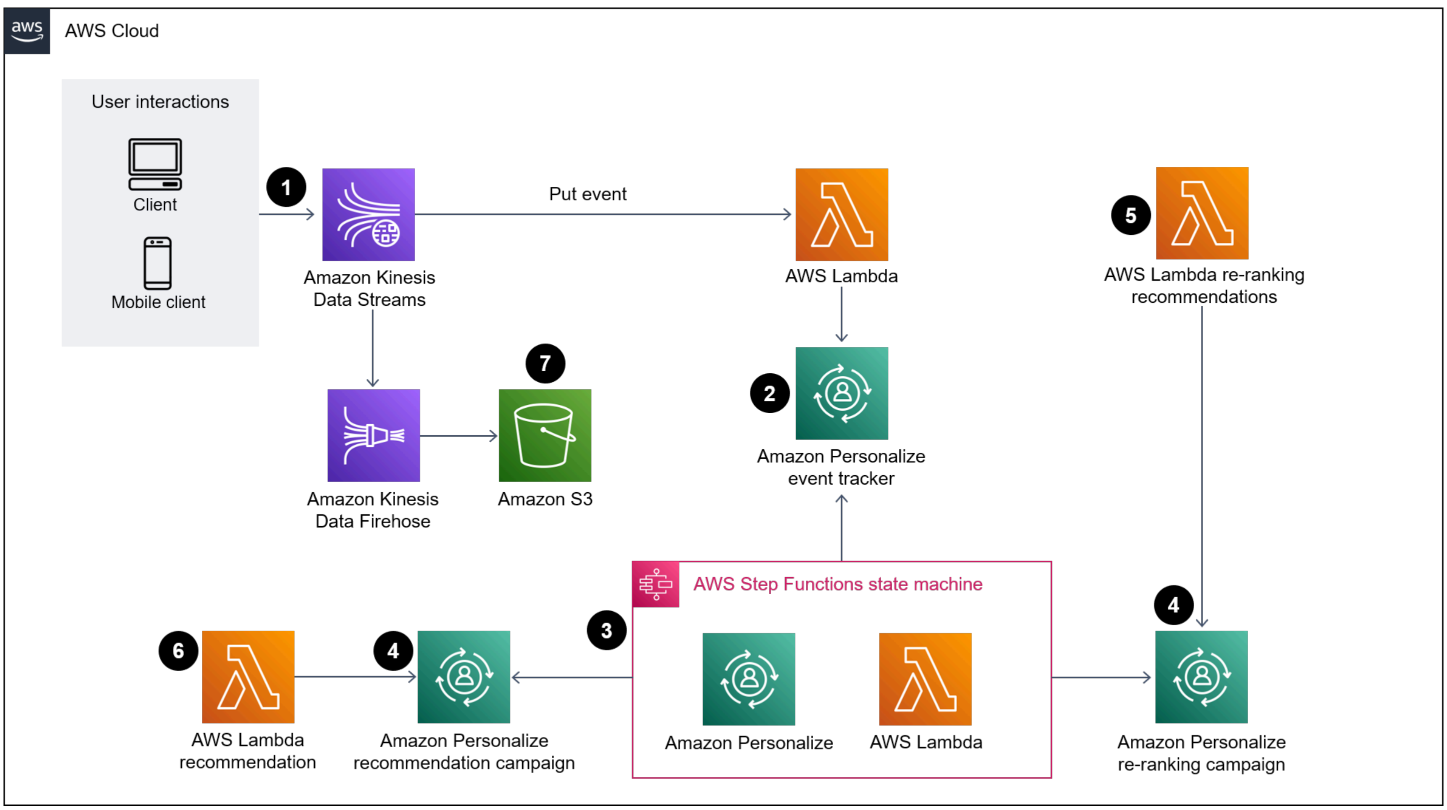
Architecture

Technology stack

- Amazon Data Firehose
- Amazon Kinesis Data Streams
- Amazon Personalize
- Amazon Simple Storage Service (Amazon S3)
- AWS Cloud Development Kit (AWS CDK)
- AWS Command Line Interface (AWS CLI)
- AWS Lambda
- AWS Step Functions

Target architecture

The following diagram illustrates a pipeline for ingesting real-time data into Amazon Personalize. The pipeline then uses that data to generate personalized and re-ranked recommendations for users.



The diagram shows the following workflow:

1. Kinesis Data Streams ingests real-time user data (for example, events like visited pets) for processing by Lambda and Firehose.

2. A Lambda function processes the records from Kinesis Data Streams and makes an API call to add the user-interaction in the record to an event tracker in Amazon Personalize.
3. A time-based rule invokes a Step Functions state machine and generates new solution versions for the recommendation and re-ranking models by using the events from the event tracker in Amazon Personalize.
4. Amazon Personalize campaigns are updated by the state machine to use the new solution version.
5. Lambda re-ranks the list of recommended items by calling the Amazon Personalize re-ranking campaign.
6. Lambda retrieves the list of recommended items by calling the Amazon Personalize recommendations campaign.
7. Firehose saves the events to an S3 bucket where they can be accessed as historical data.

Tools

AWS tools

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon Data Firehose helps you deliver real-time streaming data to other AWS services, custom HTTP endpoints, and HTTP endpoints owned by supported third-party service providers.
- Amazon Kinesis Data Streams helps you collect and process large streams of data records in real time.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Personalize is a fully managed machine learning (ML) service that helps you generate item recommendations for your users based on your data.
- AWS Step Functions is a serverless orchestration service that helps you combine Lambda functions and other AWS services to build business-critical applications.

Other tools

- pytest is a Python framework for writing small, readable tests.
- Python is a general-purpose computer programming language.

Code

The code for this pattern is available in the GitHub Animal Recommender repository. You can use the AWS CloudFormation template from this repository to deploy the resources for the example solution.

> **Note**
>
> The Amazon Personalize solution versions, event tracker, and campaigns are backed by custom resources (within the infrastructure) that expand on native CloudFormation resources.

Epics

Create the infrastructure

Task	Description	Skills required
Create an isolated Python environment.	**Mac/Linux setup** 1. To manually create a virtual environment, run the `$ python3 -m venv .venv` command from your terminal. 2. After the init process completes, run the `$ source .venv/bin/activate` command to activate the virtual environment. **Windows setup**	DevOps engineer

Task	Description	Skills required
	To manually create a virtual environment, run the `% .venv\Scripts\activate.bat` command from your terminal.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Synthesize the CloudFormation template.</td><td>1. To install the required dependencies, run the <code>$ pip install -r requirements.txt</code> command from your terminal.
2. In the AWS CLI, set the following environment variables:
• <code>export ACCOUNT_ID=123456789</code>
• <code>export CDK_DEPLOY_REGION=us-east-1</code>
• <code>export CDK_ENVIRONMENT=dev</code>
3. In the <code>config/{env}.yml</code> file, update <code>vpcId</code> to match your virtual private cloud (VPC) ID.
4. To synthesize the CloudFormation template for this code, run the <code>$ cdk synth</code> command.

Note
In step 2, <code>CDK_ENVIRONMENT</code> refers to the <code>config/{env}.yml</code> file.</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
Deploy resources and create infrastructure.	To deploy the solution resources, run the `./deploy.sh` command from your terminal. This command installs the required Python dependencies. A Python script creates an S3 bucket and an AWS Key Management Service (AWS KMS) key, and then adds the seed data for the initial model creations. Finally, the script runs `cdk deploy` to create the remaining infrastructure. **Note** The initial model training happens during stack creation. It can take up to two hours for the stack to finish getting created.	DevOps engineer

Related resources

- Animal Recommender (GitHub)
- AWS CDK Reference Documentation
- Boto3 Documentation
- Optimize personalized recommendations for a business metric of your choice with Amazon Personalize (AWS Machine Learning Blog)

Additional information

Example payloads and responses

Recommendation Lambda function

To retrieve recommendations, submit a request to the recommendation Lambda function with a payload in the following format:

```
{
  "userId": "3578196281679609099",
  "limit": 6
}
```

The following example response contains a list of animal groups:

```
[{"id": "1-domestic short hair-1-1"},
{"id": "1-domestic short hair-3-3"},
{"id": "1-domestic short hair-3-2"},
{"id": "1-domestic short hair-1-2"},
{"id": "1-domestic short hair-3-1"},
{"id": "2-beagle-3-3"},
```

If you leave out the `userId` field, the function returns general recommendations.

Re-ranking Lambda function

To use re-ranking, submit a request to the re-ranking Lambda function. The payload contains the `userId` of all the item IDs to be re-ranked and their metadata. The following example data uses the Oxford Pets classes for `animal_species_id` (1=cat, 2=dog) and integers 1-5 for `animal_age_id` and `animal_size_id`:

```
{
   "userId":"12345",
   "itemMetadataList":[
      {
         "itemId":"1",
         "animalMetadata":{
            "animal_species_id":"2",
            "animal_primary_breed_id":"Saint_Bernard",
            "animal_size_id":"3",
```

```
                "animal_age_id":"2"
            }
        },
        {
            "itemId":"2",
            "animalMetadata":{
                "animal_species_id":"1",
                "animal_primary_breed_id":"Egyptian_Mau",
                "animal_size_id":"1",
                "animal_age_id":"1"
            }
        },
        {
            "itemId":"3",
            "animalMetadata":{
                "animal_species_id":"2",
                "animal_primary_breed_id":"Saint_Bernard",
                "animal_size_id":"3",
                "animal_age_id":"2"
            }
        }
    ]
}
```


The Lambda function re-ranks these items, and then returns an ordered list that includes the item IDs and the direct response from Amazon Personalize. This is a ranked list of the animal groups that the items are in and their score. Amazon Personalize uses [User-Personalization](#) and [Personalized-Ranking](#) recipes to include a score for each item in the recommendations. These scores represent the relative certainty that Amazon Personalize has about which item the user will choose next. Higher scores represent greater certainty.


```
{
    "ranking":[
        "1",
        "3",
        "2"
    ],
    "personalizeResponse":{
        "ResponseMetadata":{
            "RequestId":"a2ec0417-9dcd-4986-8341-a3b3d26cd694",
            "HTTPStatusCode":200,
            "HTTPHeaders":{
```

```
            "date":"Thu, 16 Jun 2022 22:23:33 GMT",
            "content-type":"application/json",
            "content-length":"243",
            "connection":"keep-alive",
            "x-amzn-requestid":"a2ec0417-9dcd-4986-8341-a3b3d26cd694"
         },
         "RetryAttempts":0
      },
      "personalizedRanking":[
         {
            "itemId":"2-Saint_Bernard-3-2",
            "score":0.8947961
         },
         {
            "itemId":"1-Siamese-1-1",
            "score":0.105204
         }
      ],
      "recommendationId":"RID-d97c7a87-bd4e-47b5-a89b-ac1d19386aec"
   }
}
```


Amazon Kinesis payload

The payload to send to Amazon Kinesis has the following format:


```
{
    "Partitionkey": "randomstring",
    "Data": {
        "userId": "12345",
        "sessionId": "sessionId4545454",
        "eventType": "DetailView",
        "animalMetadata": {
            "animal_species_id": "1",
            "animal_primary_breed_id": "Russian_Blue",
            "animal_size_id": "1",
            "animal_age_id": "2"
        },
        "animal_id": "98765"

    }
}
```

Note

The `userId` field is removed for an unauthenticated user.

Streamline machine learning workflows from local development to scalable experiments by using SageMaker AI and Hydra

David Sauerwein, Marco Geiger, and Julian Ferdinand Grueber, Amazon Web Services

Summary

This pattern provides a unified approach to configuring and running machine learning (ML) algorithms from local testing to production on Amazon SageMaker AI. ML algorithms are the focus of this pattern, but its approach extends to feature engineering, inference, and whole ML pipelines. This pattern demonstrates the transition from local script development to SageMaker AI training jobs through a sample use case.

A typical ML workflow is to develop and test solutions on a local machine, run large scale experiments (for example, with different parameters) in the cloud, and deploy the approved solution in the cloud. Then, the deployed solution must be monitored and maintained. Without a unified approach to this workflow, developers often need to refactor their code at each stage. If the solution depends on a large number of parameters that might change at any stage of this workflow, it can become increasingly difficult to remain organized and consistent.

This pattern addresses these challenges. First, it eliminates the need for code refactoring between environments by providing a unified workflow that remains consistent whether running on local machines, in containers, or on SageMaker AI. Second, it simplifies parameter management through Hydra's configuration system, where parameters are defined in separate configuration files that can be easily modified and combined, with automatic logging of each run's configuration. For more details about how this pattern addresses these challenges, see [Additional information](#).

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS Identity and Access Management (IAM) [user role](#) for deploying and starting the SageMaker AI training jobs
- AWS Command Line Interface (AWS CLI) version 2.0 or later [installed](#) and [configured](#)

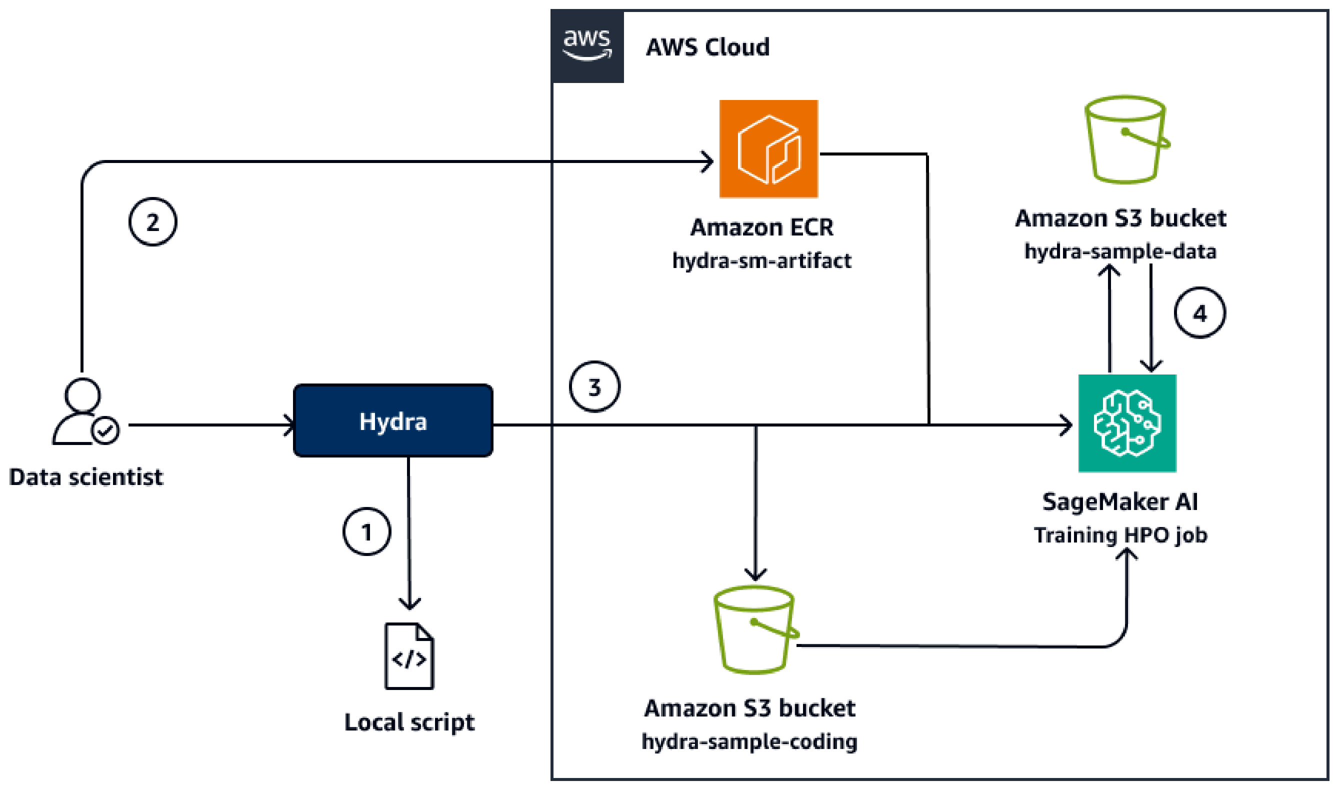
- Poetry version 1.8 or later, but earlier than 2.0, installed
- Docker installed
- Python version 3.10.x

Limitations

- The code currently only targets SageMaker AI training jobs. Extending it to processing jobs and whole SageMaker AI pipelines is straightforward.
- For a fully productionized SageMaker AI setup, additional details need to be in place. Examples could be custom AWS Key Management Service (AWS KMS) keys for compute and storage, or networking configurations. You can also configure these additional options by using Hydra in a dedicated subfolder of the `config` folder.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

The following diagram depicts the architecture of the solution.



The diagram shows the following workflow:

1. The data scientist can iterate over the algorithm at small scale in a local environment, adjust parameters, and test the training script rapidly without the need for Docker or SageMaker AI. (For more details, see the "Run locally for quick testing" task in Epics.)
2. Once satisfied with the algorithm, the data scientist builds and pushes the Docker image to the Amazon Elastic Container Registry (Amazon ECR) repository named `hydra-sm-artifact`. (For more details, see "Run workflows on SageMaker AI" in Epics.)
3. The data scientist initiates either SageMaker AI training jobs or hyperparameter optimization (HPO) jobs by using Python scripts. For regular training jobs, the adjusted configuration is written to the Amazon Simple Storage Service (Amazon S3) bucket named `hydra-sample-config`. For HPO jobs, the default configuration set located in the `config` folder is applied.
4. The SageMaker AI training job pulls the Docker image, reads the input data from the Amazon S3 bucket `hydra-sample-data`, and either fetches the configuration from the Amazon S3 bucket `hydra-sample-config` or uses the default configuration. After training, the job saves the output data to the Amazon S3 bucket `hydra-sample-data`.

Automation and scale

- For automated training, retraining, or inference, you can integrate the AWS CLI code with services like AWS Lambda, AWS CodePipeline, or Amazon EventBridge.
- Scaling can be achieved by changing configurations for instance sizes or by adding configurations for distributed training.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell. For this pattern, the AWS CLI is useful for both initial resource configuration and testing.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon SageMaker AI is a managed machine learning (ML) service that helps you build and train ML models and then deploy them into a production-ready hosted environment. SageMaker AI Training is a fully managed ML service within SageMaker AI that enables the training of ML models at scale. The tool can handle the computational demands of training models efficiently, making use of built-in scalability and integration with other AWS services. SageMaker AI Training also supports custom algorithms and containers, making it flexible for a wide range of ML workflows.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers. It was used in this pattern to ensure consistent environments across various stages, from development to deployment, and to package dependencies and code reliably. Docker's containerization allowed for easy scaling and version control across the workflow.

- Hydra is a configuration management tool that provides flexibility for handling multiple configurations and dynamic resource management. It is instrumental in managing environment configurations, allowing seamless deployment across different environments. For more details about Hydra, see Additional information.
- Python is a general-purpose computer programming language. Python was used to write the ML code and the deployment workflow.
- Poetry is a tool for dependency management and packaging in Python.

Code repository

The code for this pattern is available in the GitHub configuring-sagemaker-training-jobs-with-hydra repository.

Best practices

- Choose an IAM role for deploying and starting the SageMaker AI training jobs that follows the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.
- Use temporary credentials to access the IAM role in the terminal.

Epics

Set up the environment

Task	Description	Skills required

Create and activate the virtual environment.	To create and activate the virtual environment, run the following commands in the root of the repository:
	`poetry install`
	`poetry shell`
		General AWS

Task	Description	Skills required
Deploy the infrastructure.	To deploy the infrastructure using CloudFormation, run the following command: `aws cloudformation deploy --template-file infra/hydra-sagemaker-setup.yaml --stack-name hydra-sagemaker-setup  --capabilities CAPABILITY_NAMED_IAM`	General AWS, DevOps engineer
Download the sample data.	To download the input data from openml to your local machine, run the following command: `python scripts/download_data.py`	General AWS

Task	Description	Skills required
Run locally for quick testing.	To run the training code locally for testing, run the following command: `python mypackage/train.py data.train_data_path=data/train.csv evaluation.base_dir_path=data` The logs of all executions are stored by execution time in a folder called `outputs`. For more information, see the "Output" section in the GitHub repository. You can also perform multiple trainings in parallel, with different parameters, by using the `--multirun` functionality. For more details, see the Hydra documentation.	Data scientist

Run workflows on SageMaker AI

Task	Description	Skills required
Set the environment variables.	To run your job on SageMaker AI, set the following environment variables, providing your AWS Region and your AWS account ID:	General AWS

Task	Description	Skills required
	```export ECR_REPO_NAME=hydra-sm-artifact export image_tag=latest export AWS_REGION="<your_aws_region>" # for instance, us-east-1 export ACCOUNT_ID="<your_account_id>" export BUCKET_NAME_DATA=hydra-sample-data-$ACCOUNT_ID export BUCKET_NAME_CONFIG=hydra-sample-config-$ACCOUNT_ID export AWS_DEFAULT_REGION=$AWS_REGION export ROLE_ARN=arn:aws:iam::${ACCOUNT_ID}:role/hydra-sample-sagemaker export INPUT_DATA_S3_PATH=s3://$BUCKET_NAME_DATA/hydra-on-sm/input/ export OUTPUT_DATA_S3_PATH=s3://$BUCKET_NAME_DATA/hydra-on-sm/output/```	

Task	Description	Skills required
Create and push Docker image.	To create the Docker image and push it to the Amazon ECR repository, run the following command: `chmod +x scripts/create_and_push_image.sh scripts/create_and_push_image.sh $ECR_REPO_NAME $image_tag $AWS_REGION $ACCOUNT_ID` This task assumes that you have valid credentials in your environment. The Docker image is pushed to the Amazon ECR repository specified in the environment variable in the previous task and is used to activate the SageMaker AI container in which the training job will run.	ML engineer, General AWS

Task	Description	Skills required
Copy input data to Amazon S3.	The SageMaker AI training job needs to pick up the input data. To copy the input data to the Amazon S3 bucket for data, run the following command: `aws s3 cp data/train.csv "${INPUT_DATA_S3_PATH}train.csv"`	Data engineer, General AWS
Submit SageMaker AI training jobs.	To simplify the execution of your scripts, specify default configuration parameters in the `default.yaml` file. In addition to ensuring consistency across runs, this approach also offers the flexibility to easily override default settings as needed. See the following example: `python scripts/start_sagemaker_training_job.py sagemaker.role_arn=$ROLE_ARN sagemaker.config_s3_bucket=$BUCKET_NAME_CONFIG sagemaker.input_data_s3_path=$INPUT_DATA_S3_PATH sagemaker.output_data_s3_path=$OUTPUT_DATA_S3_PATH`	General AWS, ML engineer, Data scientist

Task	Description	Skills required
Run SageMaker AI hyperparameter tuning.	Running SageMaker AI hyperparameter tuning is similar to submitting a SageMaker AI training job. However, the execution script differs in some important ways as you can see in the start_sagemaker_hpo_job.py file. The hyperparameters to be tuned must be passed through the boto3 payload, not a channel to the training job. To start the hyperparameter optimization (HPO) job, run the following commands: `python scripts/start_sagemaker_hpo_job.py sagemaker.role_arn=$ROLE_ARN sagemaker.config_s3_bucket=$BUCKET_NAME_CONFIG sagemaker.input_data_s3_path=$INPUT_DATA_S3_PATH sagemaker.output_data_s3_path=$OUTPUT_DATA_S3_PATH`	Data scientist

Troubleshooting

Issue	Solution
Expired token	Export fresh AWS credentials.
Lack of IAM permissions	Make sure that you export the credentia ls of an IAM role that has all the required IAM permissions to deploy the CloudForm ation template and to start the SageMaker AI training jobs.

Related resources

- Train a model with Amazon SageMaker AI (AWS documentation)
- What is Hyperparameter Tuning?

Additional information

This pattern addresses the following challenges:

Consistency from local development to at-scale deployment – With this pattern, developers can use the same workflow, regardless of whether they're using local Python scripts, running local Docker containers, conducting large experiments on SageMaker AI, or deploying in production on SageMaker AI. This consistency is important for the following reasons:

- **Faster iteration** – It allows for fast, local experimentation without the need for major adjustments when scaling up.
- **No refactoring** – Transitioning to larger experiments on SageMaker AI is seamless, requiring no overhaul of the existing setup.
- **Continuous improvement** – Developing new features and continuously improving the algorithm is straightforward because the code remains the same across environments.

Configuration management – This pattern makes use of Hydra, a configuration management tool, to provide the following benefits:

- Parameters are defined in configuration files, separate from the code.
- Different parameter sets can be swapped or combined easily.
- Experiment tracking is simplified because each run's configuration is logged automatically.
- Cloud experiments can use the same configuration structure as local runs, ensuring consistency.

With Hydra, you can manage configuration effectively, enabling the following features:

- **Divide configurations** – Break your project configurations into smaller, manageable pieces that can be independently modified. This approach makes it easier to handle complex projects.
- **Adjust defaults easily** – Change your baseline configurations quickly, making it simpler to test new ideas.
- **Align CLI inputs and config files** – Combine command line inputs with your configuration files smoothly. This approach reduces clutter and confusion, making your project more manageable over time.

Train and deploy a custom GPU-supported ML model on Amazon SageMaker

Ankur Shukla, Amazon Web Services

Summary

Training and deploying a graphics processing unit (GPU)-supported machine learning (ML) model requires an initial setup and initialization of certain environment variables to fully unlock the benefits of NVIDIA GPUs. However, it can be time-consuming to set up the environment and make it compatible with Amazon SageMaker architecture on the Amazon Web Services (AWS) Cloud.

This pattern helps you train and build a custom GPU-supported ML model using Amazon SageMaker. It provides steps to train and deploy a custom CatBoost model built on an open-source Amazon reviews dataset. You can then benchmark its performance on a `p3.16xlarge` Amazon Elastic Compute Cloud (Amazon EC2) instance.

This pattern is useful if your organization wants to deploy existing GPU-supported ML models on SageMaker. Your data scientists can follow the steps in this pattern to create NVIDIA GPU-supported containers and deploy ML models on those containers.

Prerequisites and limitations

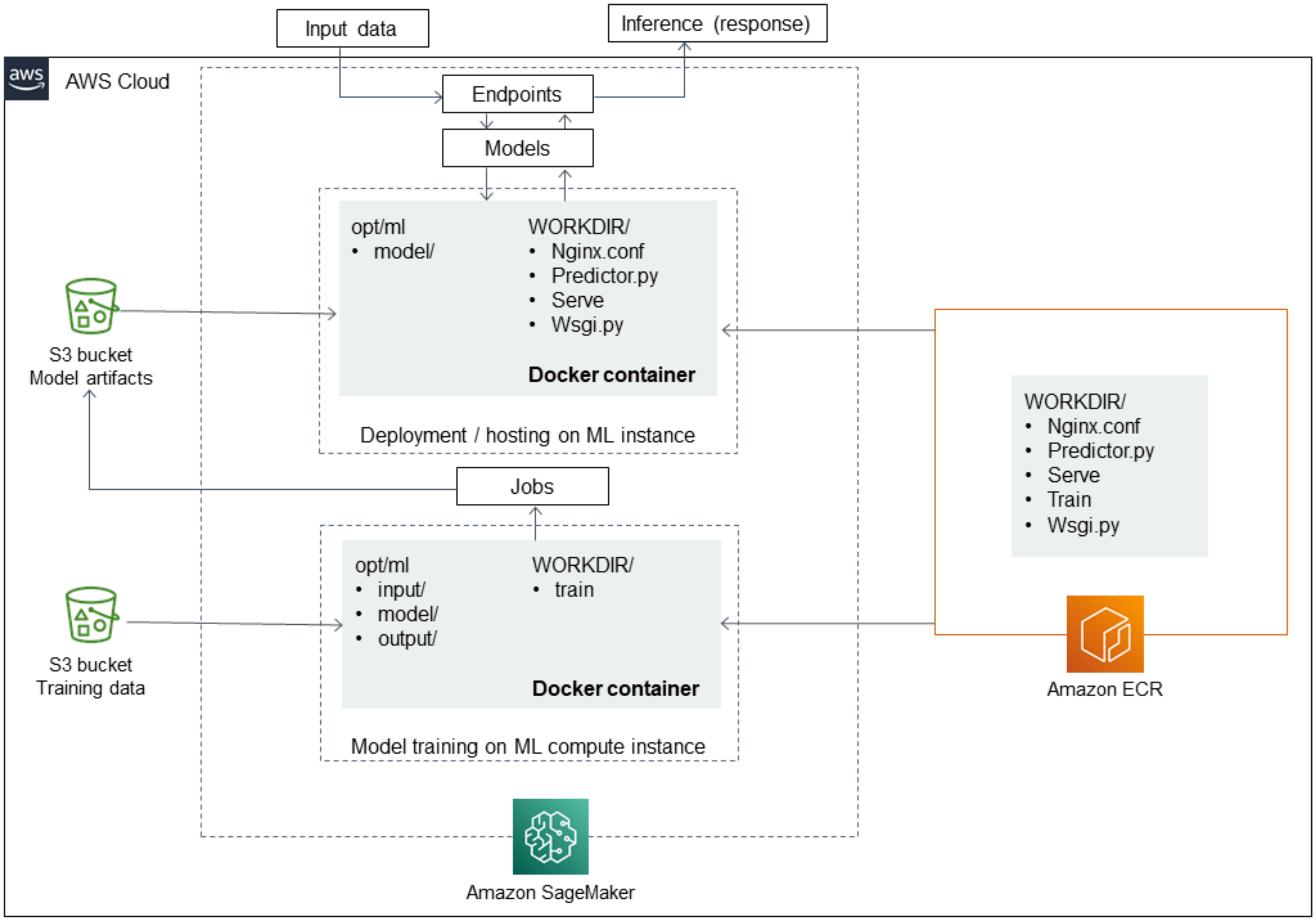
Prerequisites

- An active AWS account.
- An Amazon Simple Storage Service (Amazon S3) source bucket to store the model artifacts and predictions.
- An understanding of SageMaker notebook instances and Jupyter notebooks.
- An understanding of how to create an AWS Identity and Access Management (IAM) role with basic SageMaker role permissions, S3 bucket access and update permissions, and additional permissions for Amazon Elastic Container Registry (Amazon ECR).

Limitations

- This pattern is intended for supervised ML workloads with a train and deploy code written in Python.

Architecture



Technology stack

- SageMaker
- Amazon ECR

Tools

Tools

- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.
- Amazon SageMaker – SageMaker is a fully managed ML service.

- Docker – Docker is a software platform for building, testing, and deploying applications quickly.
- Python – Python is a programming language.

Code

The code for this pattern is available on the GitHub Implementing a review classification model with Catboost and SageMaker repository.

Epics

Prepare the data

Task	Description	Skills required
Create an IAM role and attach the required policies.	Sign in to the AWS Management Console, open the IAM console, and create a new IAM role. Attach the following policies to the IAM role: • `AmazonEC2ContainerRegistryFullAccess` • `AmazonS3FullAccess` • `AmazonSageMakerFullAccess` For more information about this, see Create a notebook instance in the Amazon SageMaker documentation.	Data scientist
Create the SageMaker notebook instance.	Open the SageMaker console, choose **Notebook instances**, and then choose **Create notebook instance**. For **IAM role**, choose the IAM role that	Data scientist

Task	Description	Skills required
	you created earlier. Configure the notebook instance according to your requirements and then choose **Create notebook instance**. For detailed steps and instructions, see [Create a notebook instance](#) in the Amazon SageMaker documentation.	
Clone the repository.	Open the terminal in the SageMaker notebook instance and clone the GitHub [Implementing a review classification model with Catboost and SageMaker](#) repository by running the following command: `git clone https://github.com/aws-samples/review-classification-using-catboost-sagemaker.git`	
Start the Jupyter notebook.	Start the `Review classification model with Catboost and SageMaker.ipynb` Jupyter notebook, which contains the predefined steps.	Data scientist

Feature engineering

Task	Description	Skills required
Run commands in Jupyter notebook.	Open the Jupyter notebook and run the commands from the following stories to prepare the data to train your ML model.	Data scientist

Read the data from the S3 bucket.	<pre>import pandas as pd
	import csv
	fname = 's3://ama
	zon-reviews-pds/ts
	v/amazon_reviews_u
	s_Digital_Video_Do
	wnload_v1_00.tsv.gz'
	df = pd.read_csv(fname,
	sep='\t',delimiter
	='\t',error_bad_li
	nes=False)</pre>
		Data scientist

Preprocess the data.	<pre>import numpy as np
	def pre_process(df):
	
	   df.fillna(value={'
	review_body': '',
	 'review_headline':
	 ''}, inplace=True)
	  df.fillna(value={'v
	erified_purchase':
	 'Unk'}, inplace=True)
	
	  df.fillna(0,
	 inplace=True)
	 return df
	df = pre_process(df)
	df.review_date =
	 pd.to_datetime(df.
	review_date)</pre>
		Data scientist

Task	Description	Skills required
	`df['target'] = np.where(df['star_rating']>=4,1,0)` **Note** This code replaces null values in the `'review_body'` with an empty string and replaces the `'verified_purchase'` column with `'Unk'`, which means "unknown."	

Task	Description	Skills required

Split the data into training, validation, and test datasets.	To keep the distribution of the target label identical across the split sets, you must stratify the sampling by using the [scikit-learn library](https://scikit-learn.org/).
	`from sklearn.model_selection import StratifiedShuffleSplit`
	`sss = StratifiedShuffleSplit(n_splits=2, test_size=0.10, random_state=0)`
	`sss.get_n_splits(df, df['target'])`
	`for train_index, test_index in sss.split(df, df['target']):`
	`    X_train_vallid , X_test = df.iloc[train_index], df.iloc[test_index]`
	
	`sss.get_n_splits(X_train_vallid, X_train_vallid['target'])`
	`for train_index, test_index in sss.split(X_train_vallid, X_train_vallid['target']):`
	`    X_train , X_valid = X_train_vallid.iloc[train_index], X_train_vallid.iloc[test_index]`
		Data scientist

Build, run, and push the Docker image to Amazon ECR

Task	Description	Skills required
Prepare and push the Docker image.	In the Jupyter notebook, run the commands from the following stories to prepare the Docker image and push it to Amazon ECR.	ML engineer

Create a repository in Amazon ECR.	<pre>%%sh
	
	algorithm_name=s
	agemaker-catboost-
	github-gpu-img
	
	chmod +x code/train
	chmod +x code/serve
	
	account=$(aws sts get-
	caller-identity --
	query Account --output
	 text)
	
	# Get the region
	 defined in the current
	 configuration (default
	 to us-west-2 if none
	 defined)
	region=$(aws configure
	 get region)
	region=${region:-us-
	east-1}
	
	fullname="${accou
	nt}.dkr.ecr.${regi
	on}.amazonaws.com/
	${algorithm_name}:
	latest"
	
	aws ecr create-re
	pository --repository-</pre>
		ML engineer

Task	Description	Skills required
	`name "${algorithm_name}" > /dev/nul`	
Build a Docker image locally.	```docker build -t "${algorithm_name}" .``` ```docker tag ${algorithm_name} ${fullname}```	ML engineer
Run the Docker image and push it to Amazon ECR.	`docker push ${fullname}`	ML engineer

Training

Task	Description	Skills required
Create a SageMaker hyperparameter tuning job.	In the Jupyter notebook, run the commands from the following stories to create a SageMaker hyperparameter tuning job using your Docker image.	Data scientist

Create a SageMaker estimator .	Create a [SageMaker estimator](https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-train-model.html) by using the Docker image's name.
	`import sagemaker as sage`
	`from time import gmtime, strftime`
	`sess = sage.Session()`
	`from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, HyperparameterTuner`
		Data scientist

Task	Description	Skills required
	```account = sess.boto_session.client('sts').get_caller_identity()['Account'] region = sess.boto_session.region_name image = '{}.dkr.ecr.{}.amazonaws.com/sagemaker-catboost-github-gpu-img:latest'.format(account, region) tree_hpo = sage.estimator.Estimator(image, role, 1, 'ml.p3.16xlarge', train_volume_size = 100, output_path="s3://{}/sagemaker/DEMO-GPU-Catboost/output".format(bucket), sagemaker_session=sess)```	

Task	Description	Skills required

Create an HPO job.	Create a hyperparameter optimization (HPO) tuning job with parameter ranges and pass the train and validation sets as parameters to the function.
	``
	hyperparameter_ranges = {'iterations': IntegerParameter(80000, 130000),
	
	 'max_depth': IntegerParameter(6, 10),
	
	 'max_ctr_complexity': IntegerParameter(4, 10),
	
	    'learning_rate': ContinuousParameter(0.01, 0.5)}
	
	objective_metric_name = 'auc'
	metric_definitions = [{'Name': 'auc',
	
	 'Regex': 'auc: ([0-9\\.]+)'}]
	
	tuner = HyperparameterTuner(tree_hpo,
	
	    objective_metric_name,
	
	    hyperparameter_ranges,
	``
		Data scientist

Task	Description	Skills required

	``
	    metric_definitions
	,
	
	    objective_type='Ma
	ximize',
	
	    max_jobs=50,
	
	    max_parallel_jobs=
	2)
	``	

Run the HPO job.	```
	train_location =
	 's3://'+bucket+'/s
	agemaker/DEMO-GPU-
	Catboost/data/train/'
	valid_location =
	 's3://'+bucket+'/s
	agemaker/DEMO-GPU-
	Catboost/data/valid/'
	
	tuner.fit({'train':
	 train_location,
	           'validati
	on': valid_location
	   })
	``
		Data scientist

Receive the best performing training job.	``` import sagemaker as sage from time import gmtime, strftime sess = sage.Session() best_job =tuner.be st_training_job() ```	Data scientist

Batch transform

Task	Description	Skills required
Create a SageMaker batch transform job on test data for model prediction.	In the Jupyter notebook, run the commands from the following stories to create the model from your SageMaker hyperparameter tuning job and submit a SageMaker batch transform job on the test data for model predictio n.	Data scientist

Create the SageMaker model.	Create a model in SageMaker model using the best training job.
	
	`attached_estimator = sage.estimator.Estimator.attach(best_job)`
	
	`output_path ='s3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/test-predictions/'`
	`input_path  ='s3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/test/'`
	
	`transformer = attached_estimator.transformer(instance_count=1,`
	
	`   instance_type='ml.p3.16xlarge',`
		Data scientist

Task	Description	Skills required

	``
	   assemble_with='Lin
	e',
	
	 accept='text/csv',
	
	 max_payload=1,
	
	   output_path=output
	_path,
	                    env
	 = {'SAGEMAKER_MODEL_
	SERVER_TIMEOUT' :
	 '3600' })
	``	

Create batch transform job.	Create batch transform job on the test data set. ``` transformer.transf orm(input_path, content_type='text/ csv', split_type='Line') ```	Data scientist

Analyze the results

Task	Description	Skills required
Read the results and evaluate the model's performance.	In the Jupyter notebook, run the commands from the following stories to read	Data scientist

Task	Description	Skills required
	the results and evaluate the performance of the model on Area Under the ROC Curve (ROC-AUC) and Area Under the Precision Recall Curve (PR-AUC) model metrics. For more information about this, see Amazon Machine Learning key concepts in the Amazon Machine Learning (Amazon ML) documentation.	

Read the batch transform job results.	Read the batch transform job results into a data frame.
	
	`file_name = 's3://'+bucket+'/sagemaker/DEMO-GPU-Catboost/data/test-predictions/file_1.out'`
	
	`results = pd.read_csv(file_name, names=['review_id','target','score'] ,sep='\t',escapechar ='\\' , quoting=csv.QUOTE_NONE, lineterminator='\n',quotechar='"').dropna()`
		Data scientist

Task	Description	Skills required

Evaluate the performance metrics.	Evaluate the performance of the model on ROC-AUC and PR-AUC.
	
	``
	from sklearn import
	 metrics
	import matplotlib
	import pandas as pd
	matplotlib.use('agg',
	 warn=False, force=Tru
	e)
	from matplotlib import
	 pyplot as plt
	
	%matplotlib inline
	
	def analyze_results(la
	bels, predictions):
	    precision, recall,
	 thresholds = metrics.p
	recision_recall_cu
	rve(labels, predictio
	ns)
	    auc = metrics.a
	uc(recall, precision)
	
	    fpr, tpr, _ =
	 metrics.roc_curve(
	labels, predictions)
	    roc_auc_score =
	 metrics.roc_auc_sc
	ore(labels, predictio
	ns)
	
	    print('Neural-
	Nets: ROC auc=%.3f' %
	 (roc_auc_score))
	
	    plt.plot(fpr, tpr,
	 label="data 1, auc=" +
	 str(roc_auc_score))
	``
		Data scientist

Task	Description	Skills required

	<pre> plt.xlabel('1-Spec
	ificity')
	    plt.ylabel('Sensit
	ivity')
	    plt.legend(loc=4)
	    plt.show()
	
	    lr_precision,
	 lr_recall, _ =
	 metrics.precision_
	recall_curve(labels,
	 predictions)
	    lr_auc = metrics.a
	uc(lr_recall,
	 lr_precision)
	    # summarize scores
	    print('Neural-
	Nets: PR auc=%.3f' %
	 (lr_auc))
	    # plot the precision
	-recall curves
	    no_skill = len(label
	s[labels==1.0]) /
	 len(labels)
	    plt.plot([0, 1],
	 [no_skill, no_skill]
	, linestyle='--',
	 label='No Skill')
	
	    plt.plot(lr_recall
	, lr_precision,
	 marker='.', label='Ne
	ural-Nets')
	    # axis labels
	    plt.xlabel('Recall
	')
	    plt.ylabel('Precis
	ion')
	    # show the legend
	    plt.legend()
	    # show the plot</pre>	

Task	Description	Skills required

	`plt.show()`
	
	`return auc`
	
	`analyze_results(results['target'].values,results['score'].values)`	

Related resources

- [Train and host Scikit-Learn models in Amazon SageMaker by building a Scikit Docker container](https://)

Additional information

The following list shows the different elements of the Dockerfile that is run in the *Build, run, and push the Docker image into Amazon ECR* epic.

Install Python with aws-cli.

```
FROM amazonlinux:1

 RUN yum update -y && yum install -y python36 python36-devel python36-libs python36-
tools python36-pip && \
 yum install gcc tar make wget util-linux kmod man sudo git -y && \
 yum install wget -y && \
 yum install aws-cli -y && \
 yum install nginx -y && \
yum install gcc-c++.noarch -y && yum clean all
```

Install the Python packages

```
RUN pip-3.6 install --no-cache-dir --upgrade pip && \pip3 install --no-cache-dir --
upgrade setuptools && \
pip3 install Cython && \
```

```
pip3 install --no-cache-dir numpy==1.16.0 scipy==1.4.1 scikit-learn==0.20.3
 pandas==0.24.2 \
flask gevent gunicorn boto3 s3fs  matplotlib joblib catboost==0.20.2
```

Install CUDA and CuDNN

```
RUN wget https://developer.nvidia.com/compute/cuda/9.0/Prod/local_installers/
cuda_9.0.176_384.81_linux-run \
&& chmod u+x cuda_9.0.176_384.81_linux-run \
&& ./cuda_9.0.176_384.81_linux-run --tmpdir=/data --silent --toolkit --override \
&& wget https://custom-gpu-sagemaker-image.s3.amazonaws.com/installation/cudnn-9.0-
linux-x64-v7.tgz \
&& tar -xvzf cudnn-9.0-linux-x64-v7.tgz \
&& cp /data/cuda/include/cudnn.h /usr/local/cuda/include \
&& cp /data/cuda/lib64/libcudnn* /usr/local/cuda/lib64 \

&& chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn* \
&&  rm -rf /data/*
```

Create the required directory structure for SageMaker

```
RUN mkdir /opt/ml /opt/ml/input /opt/ml/input/config /opt/ml/input/data /opt/ml/input/
data/training /opt/ml/model /opt/ml/output /opt/program
```

Set the NVIDIA environment variables

```
ENV PYTHONPATH=/opt/program
ENV PYTHONUNBUFFERED=TRUE
ENV PYTHONDONTWRITEBYTECODE=TRUE
ENV PATH="/opt/program:${PATH}"

# Set NVIDIA mount environments
ENV LD_LIBRARY_PATH=/usr/local/nvidia/lib:/usr/local/nvidia/lib64:$LD_LIBRARY_PATH
ENV NVIDIA_VISIBLE_DEVICES="all"
ENV NVIDIA_DRIVER_CAPABILITIES="compute,utility"
ENV NVIDIA_REQUIRE_CUDA "cuda>=9.0"
```

Copy training and inference files into the Docker image

```
COPY code/* /opt/program/
WORKDIR /opt/program
```

Translate natural language into query DSL for OpenSearch and Elasticsearch queries

Tabby Ward, Nicholas Switzer, and Breanne Warner, Amazon Web Services

Summary

This pattern demonstrates how to use large language models (LLMs) to convert natural language queries into query domain-specific language (query DSL), which makes it easier for users to interact with search services such as OpenSearch and Elasticsearch without extensive knowledge of the query language. This resource is particularly valuable for developers and data scientists who want to enhance search-based applications with natural language querying capabilities, ultimately improving user experience and search functionality.

The pattern illustrates techniques for prompt engineering, iterative refinement, and incorporation of specialized knowledge, all of which are crucial in synthetic data generation. Although this approach focuses primarily on query conversion, it implicitly demonstrates the potential for data augmentation and scalable synthetic data production. This foundation could be extended to more comprehensive synthetic data generation tasks, to highlight the power of LLMs in bridging unstructured natural language inputs with structured, application-specific outputs.

This solution doesn't involve migration or deployment tools in the traditional sense. Instead, it focuses on demonstrating a proof of concept (PoC) for converting natural language queries to query DSL by using LLMs.

- The pattern uses a Jupyter notebook as a step-by-step guide for setting up the environment and implementing the text-to-query conversion.
- It uses Amazon Bedrock to access LLMs, which are crucial for interpreting natural language and generating appropriate queries.
- The solution is designed to work with Amazon OpenSearch Service. You can follow a similar process for Elasticsearch, and the generated queries could potentially be adapted for similar search engines.

[Query DSL](#) is a flexible, JSON-based search language that's used to construct complex queries in both Elasticsearch and OpenSearch. It enables you to specify queries in the query parameter of search operations, and supports various query types. A DSL query includes leaf queries and compound queries. Leaf queries search for specific values in certain fields and encompass full-text, term-level, geographic, joining, span, and specialized queries. Compound queries act as wrappers for multiple leaf or compound clauses, and combine their results or modify their behavior. Query DSL supports the creation of sophisticated searches, ranging from simple, match-all queries to complex, multi-clause queries that produce highly specific results. Query DSL is particularly valuable for projects that require advanced search capabilities, flexible query construction, and JSON-based query structures.

This pattern uses techniques such as few-shot prompting, system prompts, structured output, prompt chaining, context provision, and task-specific prompts for text-to-query DSL conversion. For definitions and examples of these techniques, see the Additional information section.

Prerequisites and limitations

Prerequisites

To effectively use the Jupyter notebook for converting natural language queries into query DSL queries, you need:

- **Familiarity with Jupyter notebooks**. Basic understanding of how to navigate and run code in a Jupyter notebook environment.
- **Python environment**. A working Python environment, preferably Python 3.x, with the necessary libraries installed.
- **Elasticsearch or OpenSearch knowledge**. Basic knowledge of Elasticsearch or OpenSearch, including its architecture and how to perform queries.
- **AWS account**. An active AWS account to access Amazon Bedrock and other related services.
- **Libraries and dependencies**. Installation of specific libraries mentioned in the notebook, such as `boto3` for AWS interaction, and any other dependencies required for LLM integration.
- **Model access within Amazon Bedrock**. This pattern uses three Claude LLMs from Anthropic. Open the Amazon Bedrock console and choose **Model access**. On the next screen, choose **Enable specific models** and select these three models:
  - Claude 3 Sonnet
  - Claude 3.5 Sonnet
  - Claude 3 Haiku
- **Proper IAM policies and IAM role**. To run the notebook in an AWS account, your AWS Identity and Access Management (IAM) role requires the `SagemakerFullAccess` policy as well

as the policy that's provided in the Additional information section, which you can name `APGtext2querydslpolicy`. This policy includes subscribing to the three Claude models listed.

Having these prerequisites in place ensures a smooth experience when you work with the notebook and implement the text-to-query functionality.

Limitations

- **Proof of concept status**. This project is primarily intended as a proof of concept (PoC). It demonstrates the potential of using LLMs to convert natural language queries into query DSL, but it might not be fully optimized or production-ready.
- **Model limitations**:

  Context window constraints. When using the LLMs that are available on Amazon Bedrock, be aware of the context window limitations:

  Claude models (as of September 2024):

  - Claude 3 Opus: 200,000 tokens
  - Claude 3 Sonnet: 200,000 tokens
  - Claude 3 Haiku: 200,000 tokens

  Other models on Amazon Bedrock might have different context window sizes. Always check the latest documentation for the latest information.

  Model availability. The availability of specific models on Amazon Bedrock can vary. Make sure that you have access to the required models before you implement this solution.
- **Additional limitations**
  - **Query complexity**. The effectiveness of the natural language to query DSL conversion might vary depending on the complexity of the input query.
  - **Version compatibility**. The generated query DSL might need adjustments based on the specific version of Elasticsearch or OpenSearch that you use.
  - **Performance**. This pattern provides a PoC implementation, so query generation speed and accuracy might not be optimal for large-scale production use.
  - **Cost**. Using LLMs in Amazon Bedrock incurs costs. Be aware of the pricing structure for your chosen model. For more information, see Amazon Bedrock pricing.

- **Maintenance**. Regular updates to the prompts and model selection might be necessary to keep up with advancements in LLM technology and changes in query DSL syntax.

Product versions

This solution was tested in Amazon OpenSearch Service. If you want to use Elasticsearch, you might have to make some changes to replicate the exact functionality of this pattern.

- **OpenSearch version compatibility**. OpenSearch maintains backward compatibility within major versions. For example:
  - OpenSearch 1.x clients are generally compatible with OpenSearch 1.x clusters.
  - OpenSearch 2.x clients are generally compatible with OpenSearch 2.x clusters.

  However, it's always best to use the same minor version for both client and cluster when possible.
- **OpenSearch API compatibility**. OpenSearch maintains API compatibility with Elasticsearch OSS 7.10.2 for most operations. However, some differences exist, especially in newer versions.
- **OpenSearch upgrade considerations**:
  - Direct downgrades are not supported. Use snapshots for rollback if needed.
  - When you upgrade, check the compatibility matrix and release notes for any breaking changes.

Elasticsearch considerations

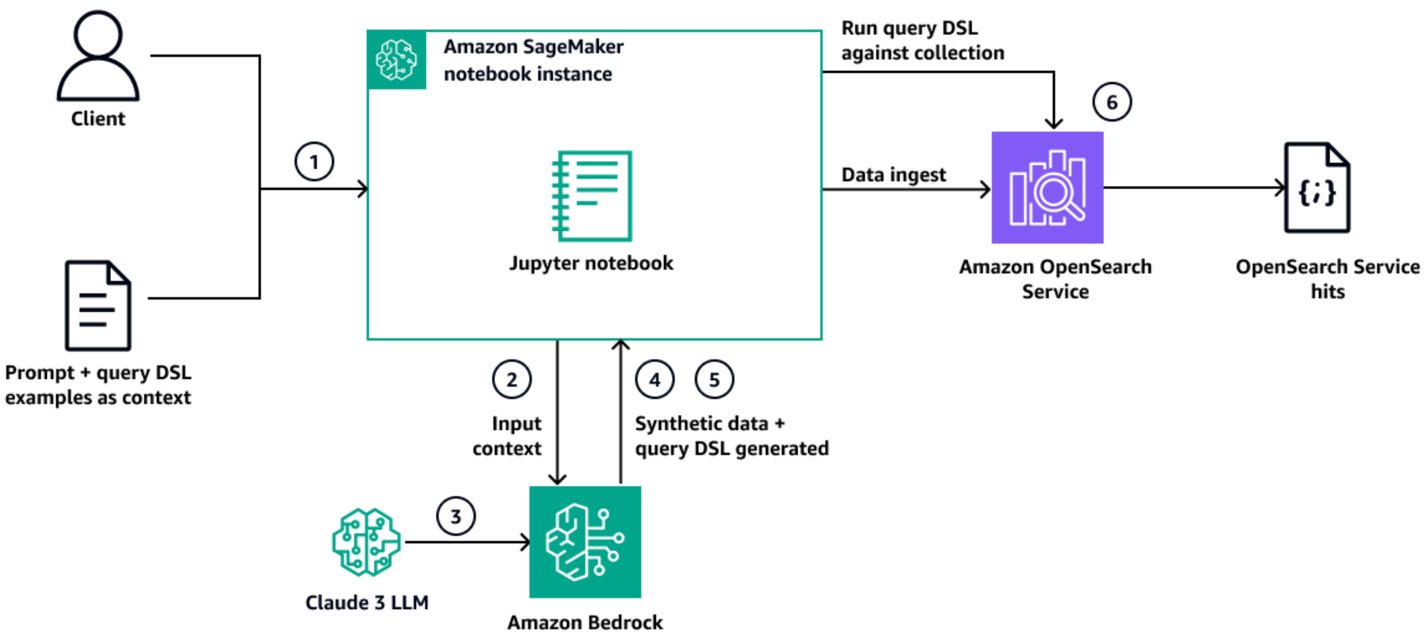
- **Elasticsearch version**. The major version of Elasticsearch you're using is crucial, because query syntax and features can change between major versions. Currently, the latest stable version is Elasticsearch 8.x. Make sure that your queries are compatible with your specific Elasticsearch version.
- **Elasticsearch query DSL library version**. If you're using the Elasticsearch query DSL Python library, make sure that its version matches your Elasticsearch version. For example:
  - For Elasticsearch 8.x, use an `elasticsearch-dsl` version that's greater or equal to 8.0.0 but smaller than 9.0.0.
  - For Elasticsearch 7.x, use an `elasticsearch-dsl` version that's greater or equal to 7.0.0 but smaller than 8.0.0.
- **Client library version**. Whether you're using the official Elasticsearch client or a language-specific client, make sure that it's compatible with your Elasticsearch version.

- **Query DSL version**. Query DSL evolves with Elasticsearch versions. Some query types or parameters might be deprecated or introduced in different versions.
- **Mapping version**. The way you define mappings for your indexes and change between versions. Always check the mapping documentation for your specific Elasticsearch version.
- **Analysis tools versions**. If you're using analyzers, tokenizers, or other text analysis tools, their behavior or availability might change between versions.

Architecture

Target architecture

The following diagram illustrates the architecture for this pattern.



where:

1. User input and system prompt with few-shot prompting examples. The process begins with a user who provides a natural language query or a request for schema generation.
2. Amazon Bedrock. The input is sent to Amazon Bedrock, which serves as the interface to access the Claude LLM.
3. Claude 3 Sonnet LLM. Amazon Bedrock uses Claude 3 Sonnet from the Claude 3 family of LLMs to process the input. It interprets and generates the appropriate Elasticsearch or OpenSearch query DSL. For schema requests, it generates synthetic Elasticsearch or OpenSearch mappings.

4. Query DSL generation. For natural language queries, the application takes the LLM's output and formats it into a valid Elasticsearch or OpenSearch Service query DSL.
5. Synthetic data generation. The application also takes schemas to create synthetic Elasticsearch or OpenSearch data to be loaded into an OpenSearch Serverless collection for testing.
6. OpenSearch or Elasticsearch. The generated Query DSL is queried against an OpenSearch Serverless collection on all indexes. JSON output contains the relevant data and number of *hits* from the data that resides in the OpenSearch Serverless collection.

Automation and scale

The code that's provided with this pattern is built strictly for PoC purposes. The following list provides a few suggestions for automating and scaling the solution further and moving the code to production. These enhancements are outside the scope of this pattern.

- Containerization:
  - Dockerize the application to ensure consistency across different environments.
  - Use container orchestration platforms such as Amazon Elastic Container Service (Amazon ECS) or Kubernetes for scalable deployments.
- Serverless architecture:
  - Convert the core functionality into AWS Lambda functions.
  - Use Amazon API Gateway to create RESTful endpoints for the natural language query input.
- Asynchronous processing:
  - Implement Amazon Simple Queue Service (Amazon SQS) to queue incoming queries.
  - Use AWS Step Functions to orchestrate the workflow of processing queries and generating query DSL.
- Caching:
  - Implement a mechanism to cache the prompts.
- Monitoring and logging:
  - Use Amazon CloudWatch for monitoring and alerting.
  - Implement centralized logging with Amazon CloudWatch Logs or Amazon OpenSearch Service for log analytics.
- Security enhancements:
  - Implement IAM roles for fine-grained access control.

  - Use AWS Secrets Manager to securely store and manage API keys and credentials.
- Multi-Region deployment:
  - Consider deploying the solution across multiple AWS Regions for improved latency and disaster recovery.
  - Use Amazon Route 53 for intelligent request routing.

By implementing these suggestions, you can transform this PoC into a robust, scalable, and production-ready solution. We recommend that you thoroughly test each component and the entire system before full deployment.

Tools

Tools

- Amazon SageMaker AI notebooks are fully managed Jupyter notebooks for machine learning development. This pattern uses notebooks as an interactive environment for data exploration, model development, and experimentation in Amazon SageMaker AI. Notebooks provide seamless integration with other SageMaker AI features and AWS services.
- Python is a general-purpose computer programming language. This pattern uses Python as the core language to implement the solution.
- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API. Amazon Bedrock provides access to LLMs for natural language processing. This pattern uses Anthropic Claude 3 models.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services, including Amazon Bedrock.
- Amazon OpenSearch Service is a managed service that helps you deploy, operate, and scale OpenSearch Service clusters in the AWS Cloud. This pattern uses OpenSearch Service as the target system for generating query DSL.

Code repository

The code for this pattern is available in the GitHub Prompt Engineering Text-to-QueryDSL Using Claude 3 Models repository. The example uses a health social media app that creates posts for users and user profiles associated with the health application.

Best practices

When working with this solution, consider the following:

- The need for proper AWS credentials and permissions to access Amazon Bedrock
- Potential costs associated with using AWS services and LLMs
- The importance of understanding query DSL to validate and potentially modify the generated queries

Epics

Set up the environment and prepare data

Task	Description	Skills required
Set up the development environment.	**Note** For detailed instructions and code for this and the other steps in this pattern, see the comprehensive walkthrough in the GitHub repository. 1. Install the required Python packages, including `boto3`, `numpy`, `awscli`, `opensearch-py` , and `requests-aws4auth` by using **pip**. 2. Import the necessary modules such as `boto3`, `json`, `os`, `opensearch` from `opensearchpy` ,	Python, pip, AWS SDK

Task	Description	Skills required
	`RequestsHttpConnection` from `Opensearchpy`, `bulk` from `opensearchpy.helpers`, `sagemaker`, `time`, `random`, `re`, and `AWS4Auth` from `requests_aws4auth`.	
Set up AWS access.	Set up the Amazon Bedrock client and SageMaker AI session. Retrieve the Amazon Resource Name (ARN) for the SageMaker AI execution role for later use in creating the OpenSearch Serverless collection.	IAM, AWS CLI, Amazon Bedrock, Amazon SageMaker
Load health app schemas.	Read and parse JSON schemas for health posts and user profiles from predefined files. Convert schemas to strings for later use in prompts.	DevOps engineer, General AWS, Python, JSON

Generate synthetic data

Task	Description	Skills required
Create an LLM-based data generator.	Implement the **generate_data()** function to call the Amazon Bedrock Converse API with Claude 3 models. Set up model IDs for Sonnet, Sonnet 3.5, and Haiku:	Python, Amazon Bedrock API, LLM prompting

Task	Description	Skills required
	```model_id_sonnet3_5 = "anthropic.claude-3-5-sonnet-20240620-v1:0" model_id_sonnet = "anthropic.claude-3-sonnet-20240229-v1:0" model_id_haiku = "anthropic.claude-3-haiku-20240307-v1:0"```	
Create synthetic health posts.	Use the **generate_data()** function with a specific message prompt to create synthetic health post entries based on the provided schema. The function call looks like this: ```health_post_data = generate_data(bedrock_rt, model_id_sonnet, system_prompt, message_healthpost, inference_config)```	Python, JSON
Create synthetic user profiles.	Use the **generate_data()** function with a specific message prompt to create synthetic user profile entries based on the provided schema. This is similar to health posts generation, but uses a different prompt.	Python, JSON

Set up OpenSearch and ingest data

Task	Description	Skills required
Set up an OpenSearch Serverless collection.	Use Boto3 to create an OpenSearch Serverless collection with appropriate encryption, network, and access policies. The collection creation looks like this: `collection = aoss_client.create_collection(name=es_name, type='SEARCH')` For more information about OpenSearch Serverless, see the [AWS documentation]().	OpenSearch Serverless, IAM
Define OpenSearch indexes.	Create indiexes for health posts and user profiles by using the OpenSearch client, based on the predefined schema mappings. The index creation looks like this: `response_health = oss_client.indices.create(healthpost_index, body=healthpost_body)`	OpenSearch, JSON
Load data into OpenSearch.	Run the **ingest_data()** function to bulk insert the synthetic health posts and user profiles into their respective OpenSearch	Python, OpenSearch API, bulk data operations

Task	Description	Skills required
	indexes. The function uses the bulk helper from `opensearch-py` : `success, failed = bulk(oss_client, actions)`	

Generate and run queries

Task	Description	Skills required
Design few-shot prompt examples.	Generate example queries and corresponding natural language questions by using Claude 3 models to serve as few-shot examples for query generation. The system prompt includes these examples: `system_prompt_query_generation = [{"text": f"""You are an expert query dsl generator. ... Examples: {example_prompt} ..."""}]`	LLM prompting, query DSL
Create a text-to-query DSL converter.	Implement the system prompt, which includes schemas, data, and few-shot examples, for query generation. Use the system prompt to convert natural language	Python, Amazon Bedrock API, LLM prompting

Task	Description	Skills required
	queries to query DSL. The function call looks like this: ```query_response = generate_data(bedrock_rt, model_id, system_prompt_query_generation, query, inference_config)```	
Test query DSL on OpenSearch.	Run the **query_oss()** function to run the generated query DSL against the OpenSearch Serverless collection and return results. The function uses the OpenSearch client's search method: ```response = oss_client.search(index="_all", body=temp)```	Python, OpenSearch API, query DSL

Test and evaluate

Task	Description	Skills required
Create a test query set.	Use Claude 3 to generate a diverse set of test questions based on the synthetic data and schemas: ```test_queries = generate_data(bedrock_rt, model_id_sonnet, query_sys```	LLM prompting

Task	Description	Skills required
	`tem_prompt, query_prompt, inference_config)`	
Assess the accuracy of the query DSL conversion.	Test the generated query DSL by running queries against OpenSearch and analyzing the returned results for relevance and accuracy. This involves running the query and inspecting the hits: `output = query_oss(response1) print("Response after running query against Opensearch") print(output)`	Python, data analysis, query DSL
Benchmark Claude 3 models.	Compare the performance of different Claude 3 models (Haiku, Sonnet, Sonnet 3.5) for query generation in terms of accuracy and latency. To compare, change the `model_id` when you call **generate_data()** and measure execution time.	Python, performance benchmarking

Clean up and document

Task	Description	Skills required
Develop a cleanup process.	Delete all indexes from the OpenSearch Serverless collection after use.	Python, AWS SDK, OpenSearch API

Related resources

- Query DSL (OpenSearch documentation)
- Amazon OpenSearch Service documentation
- OpenSearch Serverless collections
- Amazon Bedrock documentation
- Amazon SageMaker AI documentation
- AWS SDK for Python (Boto3) documentation

Additional information

IAM policy

Here's the `APGtext2querydslpolicy` policy for the IAM role used in this pattern:

```
{
  "Version": "2012-10-17",
  "Statement": [
    { "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel",
        "bedrock:InvokeModelWithResponseStream"
      ],
      "Resource": "*"
    },
    { "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::sagemaker-*",
        "arn:aws:s3:::sagemaker-*/*"
      ]
    },
    { "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
```

```
          "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
      },
      { "Effect": "Allow",
        "Action": [
          "ec2:CreateNetworkInterface",
          "ec2:DescribeNetworkInterfaces",
          "ec2:DeleteNetworkInterface"
        ],
        "Resource": "*"
      },
      { "Effect": "Allow",
        "Action": [
          "aoss:*"
        ],
        "Resource": "*"
      },
      { "Effect": "Allow",
        "Action": [
          "iam:PassRole",
          "sagemaker:*"
        ],
        "Resource": [
          "arn:aws:iam::*:role/*", "*"
        ],
        "Condition": {
          "StringEquals": {
            "iam:PassedToService": "sagemaker.amazonaws.com"
            }
        }
      },
      { "Effect": "Allow",
        "Action": [
          "codecommit:GetBranch",
          "codecommit:GetCommit",
          "codecommit:GetRepository",
          "codecommit:ListBranches",
          "codecommit:ListRepositories"
        ],
        "Resource": "*"
      },
      { "Effect": "Allow",
        "Action": [
```

```
        "aws-marketplace:Subscribe"
      ],
      "Resource": "*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws-marketplace:ProductId": [
            "prod-6dw3qvchef7zy",
            "prod-m5ilt4siql27k",
            "prod-ozonys2hmmpeu"
          ]
        }
      }
    },
    { "Effect": "Allow",
      "Action": [
        "aws-marketplace:Unsubscribe",
        "aws-marketplace:ViewSubscriptions"
      ],
      "Resource": "*"
    },
    { "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
}
```

Prompt techniques with Anthropic Claude 3 models

This pattern demonstrates the following prompting techniques for text-to-query DSL conversion using Claude 3 models.

- **Few-shot prompting:** Few-shot prompting is a powerful technique for improving the performance of Claude 3 models on Amazon Bedrock. This approach involves providing the model with a small number of examples that demonstrate the desired input/output behavior before asking it to perform a similar task. When you use Claude 3 models on Amazon Bedrock, few-shot prompting can be particularly effective for tasks that require specific formatting, reasoning patterns, or domain knowledge. To implement this technique, you typically structure your prompt with two main components: the example section and the actual query. The example section contains one or more input/output pairs that illustrate the task, and the query section presents the new input for which you want a response. This method helps Claude 3 understand the context and expected output format, and often results in a more accurate and consistent response.

Example:

```
"query": {
  "bool": {
    "must": [
      {"match": {"post_type": "recipe"}},
      {"range": {"likes_count": {"gte": 100}}},
      {"exists": {"field": "media_urls"}}
    ]
  }
}
Question: Find all recipe posts that have at least 100 likes and include media URLs.
```

- **System prompts:** In addition to few-shot prompting, Claude 3 models on Amazon Bedrock also support the use of system prompts. System prompts are a way to provide overall context, instructions, or guidelines to the model before presenting it with specific user inputs. They are particularly useful for setting the tone, defining the model's role, or establishing constraints for the entire conversation. To use a system prompt with Claude 3 on Amazon Bedrock, you include it in the `system` parameter of your API request. This is separate from the user messages and applies to the entire interaction. Detailed system prompts are used to set context and provide guidelines for the model.

  Example:

```
You are an expert query dsl generator. Your task is to take an input question and
 generate a query dsl to answer the question. Use the schemas and data below to
 generate the query.

Schemas: [schema details]
Data: [sample data]
Guidelines:
- Ensure the generated query adheres to DSL query syntax
- Do not create new mappings or other items that aren't included in the provided
 schemas.
```

- **Structured output**: You can instruct the model to provide output in specific formats, such as JSON or within XML tags.

Example:

```
Put the query in json tags
```

- **Prompt chaining**: The notebook uses the output of one LLM call as input for another, such as using generated synthetic data to create example questions.
- **Context provision**: Relevant context, including schemas and sample data, is provided in the prompts.

  Example:

```
Schemas: [schema details]
Data: [sample data]
```

- **Task-specific prompts**: Different prompts are crafted for specific tasks, such as generating synthetic data, creating example questions, and converting natural language queries to query DSL.

  Example for generating test questions:

```
Your task is to generate 5 example questions users can ask the health app based on
 provided schemas and data. Only include the questions generated in the response.
```

Use Amazon Q Developer as a coding assistant to increase your productivity

Ram Kandaswamy, Amazon Web Services

Summary

This pattern uses a tic-tac-toe game to demonstrate how you can apply Amazon Q Developer across a range of development tasks. It generates code for a tic-tac-toe game as a single-page application (SPA), enhances its UI, and creates scripts to deploy the application on AWS.

Amazon Q Developer functions as a coding assistant to help accelerate software development workflows and enhance productivity for both developers and non-developers. Regardless of your technical expertise, it helps you create architectures and design solutions for business problems, bootstraps your working environment, helps you implement new features, and generates test cases for validation. It uses natural language instructions and AI capabilities to ensure consistent, high-quality code and to mitigate coding challenges regardless of your programming skills.

The key advantage of Amazon Q Developer is its ability to liberate you from repetitive coding tasks. When you use the `@workspace` annotation, Amazon Q Developer ingests and indexes all code files, configurations, and project structure in your integrated development environment (IDE), and provides tailored responses to help you focus on creative problem-solving. You can dedicate more time to designing innovative solutions and enhancing the user experience. If you aren't technical, you can use Amazon Q Developer to streamline workflows and collaborate more effectively with the development team. The Amazon Q Developer **Explain code** feature offers detailed instructions and summaries, so you can navigate complex code bases.

Furthermore, Amazon Q Developer provides a language-agnostic approach that helps junior and mid-level developers expand their skill sets. You can concentrate on core concepts and business logic instead of language-specific syntax. This reduces the learning curve when you switch technologies.

Prerequisites and limitations

Prerequisites

- IDE (for example, WebStorm or Visual Studio Code) with the Amazon Q Developer plugin installed. For instructions, see [Installing the Amazon Q Developer extension or plugin in your IDE] in the Amazon Q Developer documentation.

- An active AWS account set up with Amazon Q Developer. For instructions, see Getting started in the Amazon Q Developer documentation.
- **npm** installed. For instructions, see the npm documentation. This pattern was tested with npm version 10.8.
- AWS Command Line Interface (AWS CLI) installed. For instructions, see the AWS CLI documentation.

Limitations

- Amazon Q Developer can perform only one development task at a time.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Tools

- This pattern requires an IDE such as Visual Studio Code or WebStorm. For a list of supported IDEs, see the Amazon Q Developer documentation.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.

Best practices

See Best coding practices with Amazon Q Developer in AWS Prescriptive Guidance. In addition:

- When you provide prompts to Amazon Q Developer, make sure that your instructions are clear and unambiguous. Add code snippets and annotations such as `@workspace` to the prompt to provide more context for your prompts.
- Include relevant libraries and import them to avoid conflicts or incorrect guesses by the system.
- If the code generated isn't accurate or as expected, use the **Provide feedback & regenerate** option. Try breaking the prompts into smaller instructions.

Epics

Set up the working environment

Task	Description	Skills required
Create a new project.	To create a new project in your working environment, run the following command and accept the default settings for all questions: `npx create-next-app@latest`	App developer, Programmer, Software developer
Test the base application.	Run the following command and confirm that the base application loads successfully in the browser: `npm run dev`	App developer, Programmer, Software developer

Clean up the base code.	Navigate to the `page.tsx` file in the `src/app` folder and delete the default content to get a blank page. After deletion, the file should look like this:
	`export default function Home() {`
	`  return (<div></div>`
	`      );`
	`}`
		App developer, Programmer, Software developer

Use Amazon Q Developer to design a tic-tac-toe game project

Task	Description	Skills required
Get an overview of steps.	1. In your IDE, open your project, and choose the Amazon Q icon to open the chat panel. 2. In the Amazon Q Developer chat panel, ask for an overview of creating a single-page application (SPA). For example: `I would like to create a single-page application involving Next.js React framework for tic-tac-toe game. What are the steps?`	App developer, Programmer, Software developer
Generate code for tic-tac-toe.	In the chat panel, start a development task by using the `/dev` command followed by the description of the task. For example: `/dev Create a React-based single-page application written in TypeScript for a tic-tac-toe game with the following specifications: 1. Design an aesthetically pleasing interface with the game grid`	App developer, Programmer, Software developer

Task	Description	Skills required
	``` centered vertically and horizontally on the page. 2. Include a heading and clear instructions on how to play the game. 3. Implement color-cod ing for X and O marks to distinguish them easily. ``` Amazon Q Developer generates code based on your instructions.	
Inspect and accept the generated code.	Visually inspect the code, and choose **Accept code** to automatically replace the `page.tsx` file. If you run into issues, choose **Provide feedback & regenerate** and describe the issue you encountered.	App developer, Programmer, Software developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Fix lint errors.</td><td>The example tic-tac-toe game includes a grid. The code that Amazon Q Developer generates might use the default type <code>any</code>. You can add type safety by prompting Amazon Q Developer as follows:<pre><code>/dev Ensure proper
 TypeScript typing
 for the onSquare Click
 event handler
to resolve any 'any'
 type issues.</code></pre></td><td>App developer, Programmer, Software developer</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add visual appeal.</td><td>You can break the original requirement into smaller fragments. For example, you can improve the game UI with the following prompts in the dev tasks. This prompt enhances Cascading Style Sheets (CSS) styles and exports the app for deployment.<pre><code>/dev Debug and fix any
 CSS issues to correctly
 display the game grid
 and overall layout.

Simplify the code by
 removing game history
 functionality and
 related components.

Implement static file
 export to an 'out'
 directory for easy
 deployment. The
 solution
should be fully
 functional, visually
 appealing, and free of
 typing errors or layout
 issues.</code></pre></td><td>App developer, Programmer, Software developer</td></tr>
</table>

Task	Description	Skills required

Test again.	1. Now that you have completed the development lifecycle, test the code to confirm that it works as expected. To run the application locally, use the command:
	`npm run dev`
	2. If the application works as expected, use the `build` command to export the entire application to the output folder in preparation for deployment:
	`npm run build`
		App developer, Programmer, Software developer

Deploy the application to the AWS Cloud

Task	Description	Skills required

Create folders and files for deployment.	In the project in your working environment, create a deployment folder and two files inside it: `pushtos3.sh` and `cloudformation.yml` :
	`mkdir deployment && cd deployment`
	`touch pushtos3.sh && chmod +x pushtos3.sh`
		App developer, Programmer, Software developer

Task	Description	Skills required
	`touch cloudformation.yml`	

Task	Description	Skills required
Generate automation code.	1. In the chat panel for Amazon Q Developer, provide the following prompt: ```/dev Generate a Cloudformation template that creates two resources: tictactoe artifact bucket and CloudFront. CloudFront should be configured with this bucket as origin. Add cache policy appropriate for Amazon S3 and default root object as index.html. Ensure that origin access control is used and no public bucket is created. Output all the resources and their ARNs.``` 2. Review and accept the generated code. The `cloudformation.yml` file that you created previously should now be populated with an AWS CloudFormation script that creates the resources for the AWS Cloud.	AWS administrator, AWS DevOps, App developer

Task	Description	Skills required
Generate script content.	To create a deployment script, use the following prompt: `/dev Modify the pushtos3 shell script so that it can use AWS CLI commands to create a CloudFormation stack named tictactoe-stack if it does not exist already, and use cloudformation.yml as the source template. Wait for the stack to complete and sync the contents from the out folder to the S3 bucket. Perform invalidation of the CloudFront origin.`	App developer, Programmer, Software developer
Deploy the application to the AWS Cloud.	1. Configure the working environment with valid AWS credentials. 2. Run the shell script to deploy the fully functiona l tic-tac-toe game to the AWS Cloud.	AWS administrator, AWS DevOps, Cloud architect, App developer

Troubleshooting

Issue	Solution
The build doesn't create a single-page application or export it to the output folder.	Look at the contents of the `next.conf ig.mjs` file.

Issue	Solution

	If the code has the following default configuration:
	 `const nextConfig = {};`
	 modify it as follows:
	 `const nextConfig = {`
	 `  output: 'export',`
	 `  distDir: 'out',`
	 `};`

Related resources

- Creating a new React project (React documentation)
- Amazon Q Developer overview (AWS documentation)
- Amazon Q Developer best practices (AWS Prescriptive Guidance)
- Installing, configuring, and using Amazon Q Developer with JetBrains IDEs (YouTube video)
- Installing Amazon Q for the command line (AWS documentation)

Use SageMaker Processing for distributed feature engineering of terabyte-scale ML datasets

Chris Boomhower, Amazon Web Services

Summary

Many terabyte-scale or larger datasets often consist of a hierarchical folder structure, and the files in the dataset sometimes share interdependencies. For this reason, machine learning (ML) engineers and data scientists must make thoughtful design decisions to prepare such data for model training and inference. This pattern demonstrates how you can use manual macrosharding and microsharding techniques in combination with Amazon SageMaker Processing and virtual CPU (vCPU) parallelization to efficiently scale feature engineering processes for complicated big data ML datasets.

This pattern defines *macrosharding* as the splitting of data directories across multiple machines for processing, and *microsharding* as the splitting of data on each machine across multiple processing threads. The pattern demonstrates these techniques by using Amazon SageMaker with sample time-series waveform records from the PhysioNet MIMIC-III dataset. By implementing the techniques in this pattern, you can minimize the processing time and costs for feature engineering while maximizing resource utilization and throughput efficiency. These optimizations rely on distributed SageMaker Processing on Amazon Elastic Compute Cloud (Amazon EC2) instances and vCPUs for similar, large datasets, regardless of data type.

Prerequisites and limitations

Prerequisites

- Access to SageMaker notebook instances or SageMaker Studio, if you want to implement this pattern for your own dataset. If you are using Amazon SageMaker for the first time, see Get started with Amazon SageMaker in the AWS documentation.
- SageMaker Studio, if you want to implement this pattern with the PhysioNet MIMIC-III sample data.
- The pattern uses SageMaker Processing, but doesn't require any experience running SageMaker Processing jobs.

Limitations

- This pattern is well suited to ML datasets that include interdependent files. These interdependencies benefit the most from manual macrosharding and running multiple, single-instance SageMaker Processing jobs in parallel. For datasets where such interdependencies do not exist, the `ShardedByS3Key` feature in SageMaker Processing might be a better alternative to macrosharding, because it sends sharded data to multiple instances that are managed by the same Processing job. However, you can implement this pattern's microsharding strategy in both scenarios to best utilize instance vCPUs.

Product versions

- Amazon SageMaker Python SDK version 2

Architecture

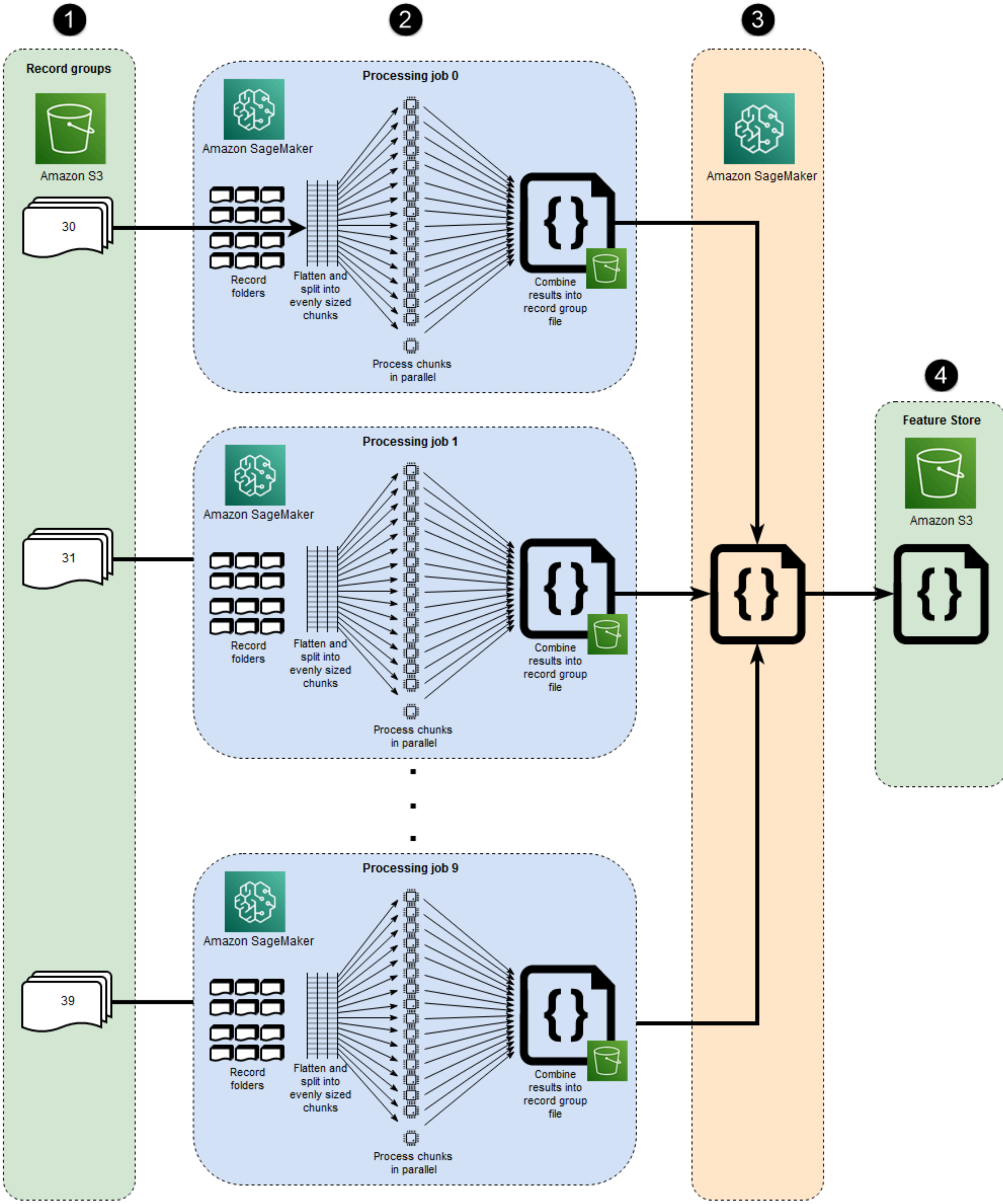
Target technology stack

- Amazon Simple Storage Service (Amazon S3)
- Amazon SageMaker

Target architecture

Macrosharding and distributed EC2 instances

The 10 parallel processes represented in this architecture reflect the structure of the MIMIC-III dataset. (Processes are represented by ellipses for diagram simplification.) A similar architecture applies to any dataset when you use manual macrosharding. In the case of MIMIC-III, you can use the dataset's raw structure to your advantage by processing each patient group folder separately, with minimal effort. In the following diagram, the record groups block appears on the left (1). Given the distributed nature of the data, it makes sense to shard by patient group.


1
2
3
4
Record groups
Amazon S3
30
31
39
Processing job 0
Processing job 1
Processing job 9
Amazon SageMaker
Record folders
Flatten and split into evenly sized chunks
Process chunks in parallel
Combine results into record group file
Amazon SageMaker
Feature Store
Amazon S3

However, manually sharding by patient group means that a separate Processing job is required for each patient group folder, as you can see in the middle section of the diagram (2), instead of a single Processing job with multiple EC2 instances. Because MIMIC-III's data includes both binary waveform files and matching text-based header files, and there is a required dependency on the wfdb library for binary data extraction, all the records for a specific patient must be made available on the same instance. The only way to be certain that each binary waveform file's associated header file is also present is to implement manual sharding to run each shard within its own Processing job, and to specify `s3_data_distribution_type='FullyReplicated'` when you define the Processing job input. Alternatively, if all data were available in a single directory and no dependencies existed between files, a more suitable option might be to launch a single Processing job with multiple EC2 instances and `s3_data_distribution_type='ShardedByS3Key'` specified. Specifying `ShardedByS3Key` as the Amazon S3 data distribution type directs SageMaker to manage data sharding automatically across instances.

Launching a Processing job for each folder is a cost-efficient way to preprocess the data, because running multiple instances concurrently saves time. For additional cost and time savings, you can use microsharding within each Processing job.

Microsharding and parallel vCPUs

Within each Processing job, the grouped data is further divided to maximize use of all available vCPUs on the SageMaker fully managed EC2 instance. The blocks in the middle section of the diagram (2) depict what happens within each primary Processing job. The contents of the patient record folders are flattened and divided evenly based on the number of available vCPUs on the instance. After the folder contents are divided, the evenly sized set of files are distributed across all vCPUs for processing. When processing is complete, the results from each vCPU are combined into a single data file for each Processing job.

In the attached code, these concepts are represented in the following section of the `src/feature-engineering-pass1/preprocessing.py` file.

```
def chunks(lst, n):
    """
    Yield successive n-sized chunks from lst.

    :param lst: list of elements to be divided
    :param n: number of elements per chunk
    :type lst: list
    :type n: int
```

```
    :return: generator comprising evenly sized chunks
    :rtype: class 'generator'
    """
    for i in range(0, len(lst), n):
        yield lst[i:i + n]


# Generate list of data files on machine
data_dir = input_dir
d_subs = next(os.walk(os.path.join(data_dir, '.')))[1]
file_list = []
for ds in d_subs:
    file_list.extend(os.listdir(os.path.join(data_dir, ds, '.')))
dat_list = [os.path.join(re.split('_|\.', f)[0].replace('n', ''), f[:-4]) for f in
 file_list if f[-4:] == '.dat']

# Split list of files into sub-lists
cpu_count = multiprocessing.cpu_count()
splits = int(len(dat_list) / cpu_count)
if splits == 0: splits = 1
dat_chunks = list(chunks(dat_list, splits))

# Parallelize processing of sub-lists across CPUs
ws_df_list = Parallel(n_jobs=-1, verbose=0)(delayed(run_process)(dc) for dc in
 dat_chunks)

# Compile and pickle patient group dataframe
ws_df_group = pd.concat(ws_df_list)
ws_df_group = ws_df_group.reset_index().rename(columns={'index': 'signal'})
ws_df_group.to_json(os.path.join(output_dir, group_data_out))
```

A function, `chunks`, is first defined to consume a given list by dividing it into evenly sized chunks of length `n` and by returning these results as a generator. Next, the data is flattened across patient folders by compiling a list of all binary waveform files that are present. After this is done, the number of vCPUs available on the EC2 instance is obtained. The list of binary waveform files is evenly divided across these vCPUs by calling `chunks`, and then each waveform sublist is processed on its own vCPU by using [joblib's Parallel class](). Results are automatically combined into a single list of dataframes by the Processing job, which SageMaker then processes further before writing it to Amazon S3 upon job completion. In this example, there are 10 files written to Amazon S3 by the Processing jobs (one for each job).

When all the initial Processing jobs are complete, a secondary Processing job, which is shown in the block to the right of the diagram (3) combines the output files produced by each primary Processing job and writes the combined output to Amazon S3 (4).

Tools

Tools

- Python – The sample code used for this pattern is Python (version 3).
- SageMaker Studio – Amazon SageMaker Studio is a web-based, integrated development environment (IDE) for machine learning that lets you build, train, debug, deploy, and monitor your machine learning models. You run SageMaker Processing jobs by using Jupyter notebooks inside SageMaker Studio.
- SageMaker Processing – Amazon SageMaker Processing provides a simplified way to run your data processing workloads. In this pattern, the feature engineering code is implemented at scale by using SageMaker Processing jobs.

Code

The attached .zip file provides the complete code for this pattern. The following section describes the steps to build the architecture for this pattern. Each step is illustrated by sample code from the attachment.

Epics

Set up your SageMaker Studio environment

Task	Description	Skills required
Access Amazon SageMaker Studio.	Onboard to SageMaker Studio in your AWS account by following the direction s provided in the Amazon SageMaker documentation.	Data scientist, ML engineer
Install the wget utility.	Install *wget* if you onboarded with a new SageMaker Studio configuration or if you've	Data scientist, ML engineer

Task	Description	Skills required
	never used these utilities in SageMaker Studio before. To install, open a terminal window in the SageMaker Studio console and run the following command: `sudo yum install wget`	

Download and unzip the sample code.	Download the `attachments.zip` file in the *Attachments* section. In a terminal window, navigate to the folder where you downloaded the file and extract its contents:
	
	`unzip attachment.zip`
	
	Navigate to the folder where you extracted the .zip file, and extract the contents of the `Scaled-Processing.zip` file.
	
	`unzip Scaled-Processing.zip`
		Data scientist, ML engineer

Task	Description	Skills required
Download the sample dataset from physionet.org and upload it to Amazon S3.	Run the `get_data.ipynb` Jupyter notebook within the folder that contains the `Scaled-Processing` files. This notebook downloads a sample MIMIC-III dataset from physionet.org and uploads it to your SageMaker Studio session bucket in Amazon S3.	Data scientist, ML engineer

Configure the first preprocessing script

Task	Description	Skills required

Flatten the file hierarchy across all subdirectories.	In large datasets such as MIMIC-III, files are often distributed across multiple subdirectories even within a logical parent group. Your script should be configure d to flatten all group files across all subdirectories, as the following code demonstra tes.
	`# Generate list of .dat files on machine`
	`data_dir = input_dir`
	`d_subs = next(os.walk(os.path.join(data_dir, '.')))[1]`
	`file_list = []`
	`for ds in d_subs:`
	`    file_list.extend(os.listdir(os.path.`
		Data scientist, ML engineer

Task	Description	Skills required
	``` join(data_dir, ds, '.'))) dat_list = [os.path. join(re.split('_|\\ .', f)[0].replace('n', ''), f[:-4]) for f in file_list if f[-4:] == '.dat'] ``` **Note** The example code snippets in this epic are from the `src/feature-engineering-pass1/preprocessing.py` file, which is provided in the attachment.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Divide files into subgroups based on vCPU count.</td><td>Files should be divided into evenly sized subgroups, or chunks, depending on the number of vCPUs present on the instance that runs the script. For this step, you can implement code similar to the following.
<pre><code># Split list of files
 into sub-lists
cpu_count = multiproc
essing.cpu_count()
splits = int(len(d
at_list) / cpu_count)
if splits == 0: splits =
 1
dat_chunks = list(chun
ks(dat_list, splits))</code></pre></td><td>Data scientist, ML engineer</td></tr>
<tr><td>Parallelize processing of subgroups across vCPUs.</td><td>Script logic should be configured to process all subgroups in parallel. To do this, use the Joblib library's <code>Parallel</code> class and <code>delayed</code> method as follows.
<pre><code># Parallelize processin
g of sub-lists across
 CPUs
ws_df_list = Parallel(
n_jobs=-1, verbose=0)
(delayed(run_process)
(dc) for dc in dat_chunk
s)</code></pre></td><td>Data scientist, ML engineer</td></tr>
</table>

Task	Description	Skills required

Save single file group output to Amazon S3.	When parallel vCPU processing is complete, the results from each vCPU should be combined and uploaded to the file group's S3 bucket path. For this step, you can use code similar to the following.
	`# Compile and pickle patient group dataframe`
	`ws_df_group = pd.concat(ws_df_list)`
	`ws_df_group = ws_df_group.reset_index().rename(columns={'index': 'signal'})`
	`ws_df_group.to_json(os.path.join(output_dir, group_data_out))`
		Data scientist, ML engineer

Configure the second preprocessing script

Task	Description	Skills required
Combine data files produced across all Processing jobs that ran the first script.	The previous script outputs a single file for each SageMaker Processing job that processes a group of files from the dataset. Next, you need to combine these output files into a single object and write a single output dataset to Amazon S3. This is demonstra	Data scientist, ML engineer

Task	Description	Skills required

	ted in the `src/feature-engineering-pass1p5/preprocessing.py` file, which is provided in the attachment, as follows.
	``
	def write_parquet(wavs_df, path):
	    """
	    Write waveform summary dataframe to S3 in parquet format.
	
	    :param wavs_df: waveform summary dataframe
	    :param path: S3 directory prefix
	    :type wavs_df: pandas dataframe
	    :type path: str
	    :return: None
	    """
	    extra_args = {"ServerSideEncryption": "aws:kms"}
	    wr.s3.to_parquet(
	        df=wavs_df,
	        path=path,
	        compression='snappy',
	        s3_additional_kwargs=extra_args)
	
	def combine_data():
	    """
	    Get combined data and write to parquet.
	``	

Task	Description	Skills required

	``
	    :return: waveform
	 summary dataframe
	    :rtype: pandas
	 dataframe
	    """
	    wavs_df = get_data(
	)
	    wavs_df = normalize
	_signal_names(wavs
	_df)
	    write_parquet(wavs
	_df, "s3://{}/{}/
	{}".format(buck
	et_xform, dataset_p
	refix, pass1p5ou
	t_data))
	
	    return wavs_df
	
	wavs_df = combine_d
	ata()
	``	

Run Processing jobs

Task	Description	Skills required
Run the first Processing job.	To perform macrosharding, run a separate Processing job for each file group. Microsharding is performed inside each Processing job, because each job runs your first script. The following code demonstrates how to launch a Processing job for each file group directory in the	Data scientist, ML engineer

Task	Description	Skills required
	following snippet (included in `notebooks/FeatExtract_Pass1.ipynb`).	

```
pat_groups = list(range(30,40))
ts = str(int(time.time()))

for group in pat_groups:
    sklearn_processor = SKLearnProcessor(framework_version='0.20.0',
                                         role=role,
                                          instance_type='ml.m5.4xlarge',
                                          instance_count=1,
                                          volume_size_in_gb=5)
    sklearn_processor.run(
        code='../src/feature-engineering-pass1/preprocessing.py',
        job_name='-'.join(['scaled-processing-p1', str(group), ts]),
        arguments=[
            "input_path", "/opt/ml/processing/input",
```

Task	Description	Skills required

	``
	            "output_p
	ath", "/opt/ml/
	processing/output",
	            "group_da
	ta_out", "ws_df_gr
	oup.json"
	        ],
	        inputs=
	        [
	            Processin
	gInput(
	
	 source=f's3://{ses
	s.default_bucket()}/
	data_inputs/{group}',
	
	 destination='/opt/ml/
	processing/input',
	
	 s3_data_distributi
	on_type='FullyRepl
	icated'
	            )
	        ],
	        outputs=
	        [
	            Processin
	gOutput(
	
	 source='/opt/ml/pr
	ocessing/output',
	
	 destination=f's3:/
	/{sess.default_buc
	ket()}/data_outputs/
	{group}'
	            )
	        ],
	        wait=False
	    )
	``	

Task	Description	Skills required

Run the second Processing job.	To combine the outputs generated by the first set of processing jobs and perform any additional computations for preprocessing, you run your second script by using a single SageMaker Processing job. The following code demonstrates this (included in `notebooks/FeatExtract_Pass1p5.ipynb`).
	``
	ts = str(int(time.time()))
	bucket = sess.default_bucket()
	
	sklearn_processor = SKLearnProcessor(framework_version='0.20.0',
	         role=role,
	            instance_type='ml.t3.2xlarge',
	            instance_count=1,
	            volume_size_in_gb=5)
	sklearn_processor.run(
	    code='../src/feature-engineering-pass1p5/preprocessing.py',
	``
		Data scientist, ML engineer

Task	Description	Skills required

	``
	    job_name='-'.join(
	['scaled-processing',
	 'p1p5', ts]),
	    arguments=['bucket
	', bucket,
	                'pass1out
	_prefix', 'data_out
	puts',
	                'pass1out
	_data', 'ws_df_gr
	oup.json',
	                'pass1p5o
	ut_data', 'waveform
	_summary.parquet',
	                'statsdat
	a_name', 'signal_s
	tats.csv'],
	    wait=True
	)
	``	

Related resources

- Onboard to Amazon SageMaker Studio Using Quick Start (SageMaker documentation)
- Process Data (SageMaker documentation)
- Data Processing with scikit-learn (SageMaker documentation)
- joblib.Parallel documentation
- Moody, B., Moody, G., Villarroel, M., Clifford, G. D., & Silva, I. (2020). MIMIC-III Waveform Database (version 1.0). *PhysioNet*.
- Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., & Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. Scientific Data, 3, 160035.
- MIMIC-III Waveform Database license

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Visualize AI/ML model results using Flask and AWS Elastic Beanstalk

Chris Caudill and Durga Sury, Amazon Web Services

Summary

Visualizing output from artificial intelligence and machine learning (AI/ML) services often requires complex API calls that must be customized by your developers and engineers. This can be a drawback if your analysts want to quickly explore a new dataset.

You can enhance the accessibility of your services and provide a more interactive form of data analysis by using a web-based user interface (UI) that enables users to upload their own data and visualize the model results in a dashboard.

This pattern uses [Flask] and [Plotly] to integrate Amazon Comprehend with a custom web application and visualize sentiments and entities from user-provided data. The pattern also provides the steps to deploy an application by using AWS Elastic Beanstalk. You can adapt the application by using [Amazon Web Services (AWS) AI services] or with a custom trained model hosted on an endpoint (for example, an [Amazon SageMaker endpoint]).

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured on your local machine. For more information about this, see Configuration basics in the AWS CLI documentation. You can also use an AWS Cloud9 integrated development environment (IDE); for more information about this, see Python tutorial for AWS Cloud9 and Previewing running applications in the AWS Cloud9 IDE in the AWS Cloud9 documentation.

  Notice: AWS Cloud9 is no longer available to new customers. Existing customers of AWS Cloud9 can continue to use the service as normal. Learn more
- An understanding of Flask's web application framework. For more information about Flask, see the Quickstart in the Flask documentation.

- Python version 3.6 or later, installed and configured. You can install Python by following the instructions from [Setting up your Python development environment]() in the AWS Elastic Beanstalk documentation.
- Elastic Beanstalk Command Line Interface (EB CLI), installed and configured. For more information about this, see [Install the EB CLI]() and [Configure the EB CLI]() from the AWS Elastic Beanstalk documentation.

Limitations

- This pattern's Flask application is designed to work with .csv files that use a single text column and are restricted to 200 rows. The application code can be adapted to handle other file types and data volumes.
- The application doesn't consider data retention and continues to aggregate uploaded user files until they are manually deleted. You can integrate the application with Amazon Simple Storage Service (Amazon S3) for persistent object storage or use a database such as Amazon DynamoDB for serverless key-value storage.
- The application only considers documents in the English language. However, you can use Amazon Comprehend to detect a document's primary language. For more information about the supported languages for each action, see [API reference]() in the Amazon Comprehend documentation.
- A troubleshooting list that contains common errors and their solutions is available in the *Additional information* section.

Architecture

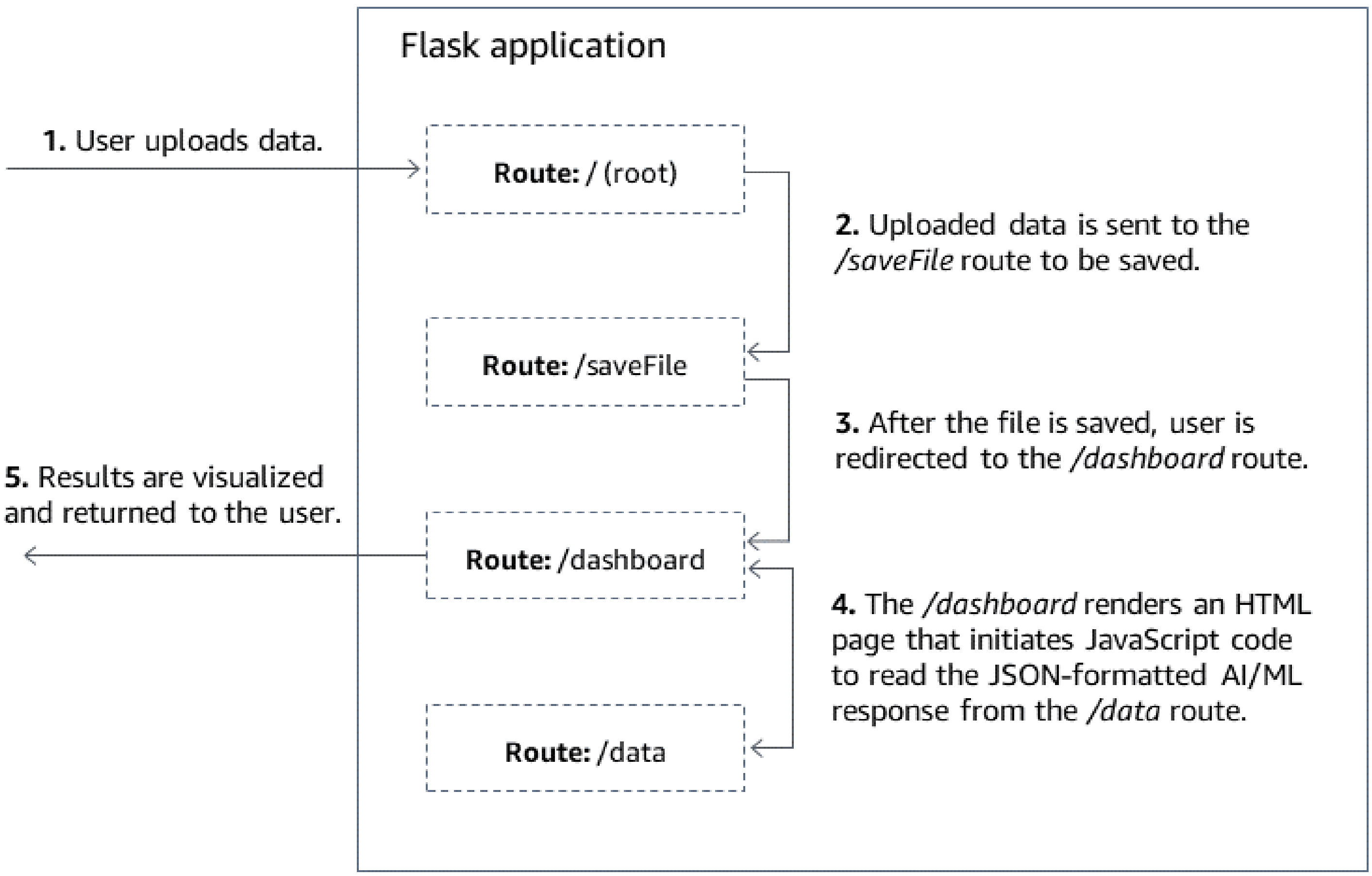
Flask application architecture

Flask is a lightweight framework for developing web applications in Python. It is designed to combine Python's powerful data processing with a rich web UI. The pattern's Flask application shows you how to build a web application that enables users to upload data, sends the data to Amazon Comprehend for inference, and then visualizes the results. The application has the following structure:

- `static` – Contains all the static files that support the web UI (for example, JavaScript, CSS, and images)
- `templates` – Contains all of the application's HTML pages

- `userData` – Stores uploaded user data
- `application.py` – The Flask application file
- `comprehend_helper.py` – Functions to make API calls to Amazon Comprehend
- `config.py` – The application configuration file
- `requirements.txt` – The Python dependencies required by the application

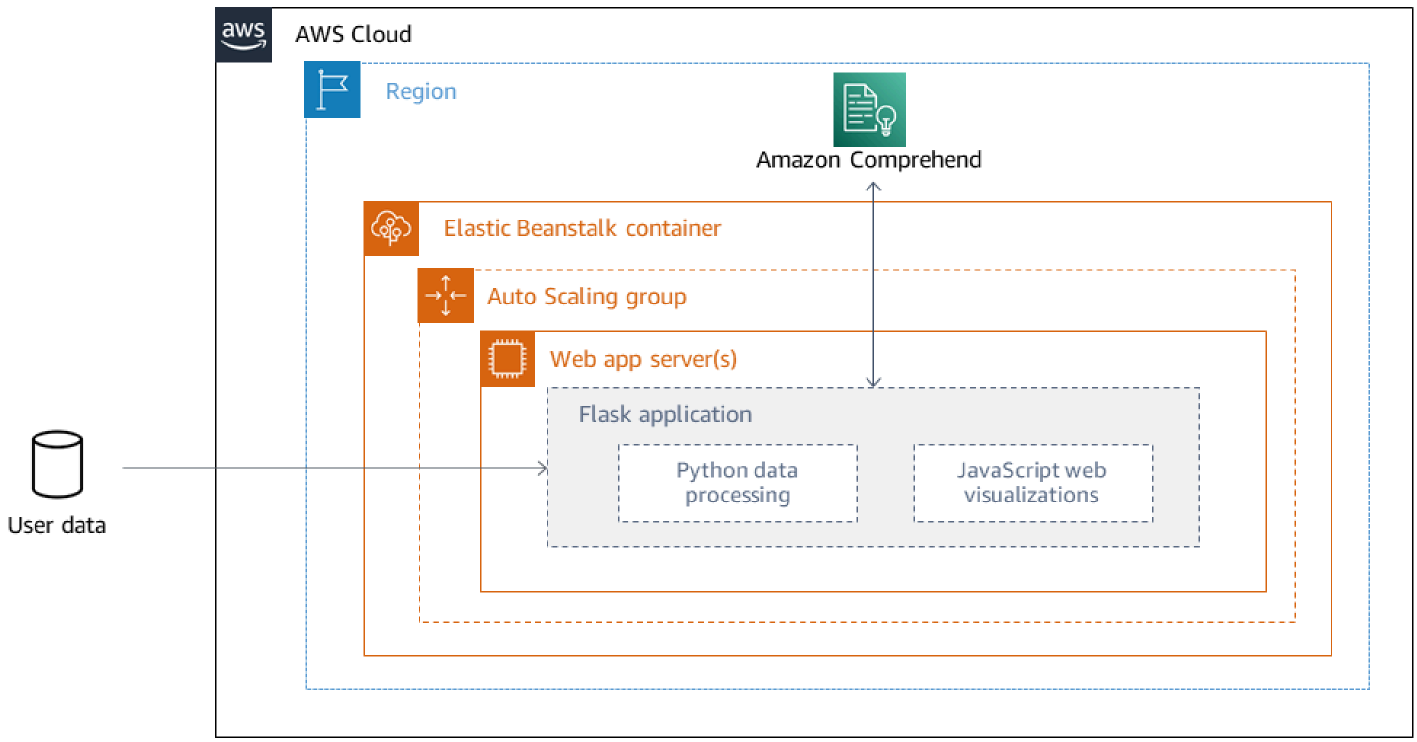
The `application.py` script contains the web application's core functionality, which consists of four Flask routes. The following diagram shows these Flask routes.



- / is the application's root and directs users to the `upload.html` page (stored in the `templates` directory).
- `/saveFile` is a route that is invoked after a user uploads a file. This route receives a `POST` request via an HTML form, which contains the file uploaded by the user. The file is saved in the `userData` directory and the route redirects users to the `/dashboard` route.

- `/dashboard` sends users to the `dashboard.html` page. Within this page's HTML, it runs the JavaScript code in `static/js/core.js` that reads data from the `/data` route and then builds visualizations for the page.
- `/data` is a JSON API that presents the data to be visualized in the dashboard. This route reads the user-provided data and uses the functions in `comprehend_helper.py` to send the user data to Amazon Comprehend for sentiment analysis and named entity recognition (NER). Amazon Comprehend's response is formatted and returned as a JSON object.

Deployment architecture



Design considerations

For more information about design considerations for applications deployed using Elastic Beanstalk on the AWS Cloud, see in the AWS Elastic Beanstalk documentation.

Technology stack

- Amazon Comprehend
- Elastic Beanstalk
- Flask

Automation and scale

Elastic Beanstalk deployments are automatically set up with load balancers and auto scaling groups. For more configuration options, see Configuring Elastic Beanstalk environments in the AWS Elastic Beanstalk documentation.

Tools

- AWS Command Line Interface (AWS CLI) is a unified tool that provides a consistent interface for interacting with all parts of AWS.
- Amazon Comprehend uses natural language processing (NLP) to extract insights about the content of documents without requiring special preprocessing.
- AWS Elastic Beanstalk helps you quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications.
- Elastic Beanstalk CLI (EB CLI) is a command line interface for AWS Elastic Beanstalk that provides interactive commands to simplify creating, updating, and monitoring environments from a local repository.
- The Flask framework performs data processing and API calls using Python and offers interactive web visualization with Plotly.

Code

The code for this pattern is available in the GitHub Visualize AI/ML model results using Flask and AWS Elastic Beanstalk repository.

Epics

Set up the Flask application

Task	Description	Skills required
Clone the GitHub repository.	Pull the application code from the GitHub Visualize AI/ML model results using Flask and AWS Elastic Beanstalk repository by running the following command:	Developer

Task	Description	Skills required
	`git clone git@github.com:aws-samples/aws-comprehend-elasticbeanstalk-for-flask.git` **Note** Make sure that you configure your SSH keys with GitHub.	

Install the Python modules.	After you clone the repository, a new local `aws-comprehend-elasticbeanstalk-for-flask` directory is created. In that directory, the `requirements.txt` file contains the Python modules and versions that run the application. Use the following commands to install the modules:
	
	`cd aws-comprehend-elasticbeanstalk-for-flask`
	
	`pip install -r requirements.txt`
		Python developer

Task	Description	Skills required

Test the application locally.	Start the Flask server by running the following command:
	`python application.py`
	This returns information about the running server. You should be able to access the application by opening a browser and visiting http://localhost:5000
	Note
	If you're running the application in an AWS Cloud9 IDE, you need to replace the `application.run()` command in the `application.py` file with the following line:
	`application.run(host=os.getenv('IP', '0.0.0.0'),port=int(os.getenv('PORT', 8080)))`
	You must revert this change before deployment.
		Python developer

Deploy the application to Elastic Beanstalk

Task	Description	Skills required
Launch the Elastic Beanstalk application.	To launch your project as an Elastic Beanstalk application, run the following command from your application's root directory: `eb init -p python-3.6 comprehend_flask --region us-east-1` **Important** • `comprehend_flask` is the name of the Elastic Beanstalk application and can be changed according to your requirements. • You can replace the AWS Region with a Region of your choice. The default Region in AWS CLI is used if you don't specify a Region. • The application was built with Python version 3.6. You might encounter errors if you use other Python versions. Run the `eb init -i` command for more	Architect, Developer

Task	Description	Skills required
	deployment configuration options.	
Deploy the Elastic Beanstalk environment.	Run the following command from the application's root directory: `eb create comprehend-flask-env` **Note** `comprehend-flask-env` is the name of the Elastic Beanstalk environment and can be changed according to your requirements. The name can only contain letters, numbers, and dashes.	Architect, Developer

Task	Description	Skills required

Authorize your deployment to use Amazon Comprehend.	Although your application might be successfully deployed, you should also provide your deployment with access to Amazon Comprehend. `ComprehendFullAccess` is an AWS managed policy that provides the deployed application with permissions to make API calls to Amazon Comprehend.
	
	Attach the `ComprehendFullAccess` policy to `aws-elasticbeanstalk-ec2-role` (this role is automatically created for your deployment's Amazon Elastic Compute Cloud (Amazon EC2) instances) by running the following command:
	
	`aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/ComprehendFullAccess --role-name aws-elasticbeanstalk-ec2-role`
	
	Important
	
	`aws-elasticbeanstalk-`
		Developer, Security architect

Task	Description	Skills required
	`ec2-role` is created when your applicati on deploys. You must complete the deployment process before you can attach the AWS Identity and Access Management (IAM) policy.	
Visit your deployed applicati on.	After your application successfully deploys, you can visit it by running the `eb open` command. You can also run the `eb status` command to receive details about your deploymen t. The deployment URL is listed under `CNAME`.	Architect, Developer

(Optional) Customize the application to your ML model

Task	Description	Skills required
Authorize Elastic Beanstalk to access the new model.	Make sure that Elastic Beanstalk has the required access permissions for your new model endpoint. For example, if you use an Amazon SageMaker endpoint, your deployment needs to have permission to invoke the endpoint.	Developer, Security architect

Task	Description	Skills required
	For more information about this, see InvokeEndpoint in the Amazon SageMaker documentation.	
Send the user data to a new model.	To change the underlying ML model in this application, you must change the following files: • `comprehend_helper.py` – This is the Python script that connects with Amazon Comprehend, processes the response, and returns the final result to the application. In this script, you can either route the data to another AI service on the AWS Cloud or you can send the data to a custom model endpoint. We recommend that you also format the results in this script for logical separation and the reusability of this pattern. • `application.py` – If you change the name of the `comprehend_helper.py` script or functions, you need to update the application `application.py` script to reflect those changes.	Data scientist

Task	Description	Skills required
Update the dashboard visualizations.	Typically, incorporating a new ML model means that visualizations must be updated to reflect the new results. These changes are made in the following files: • `templates/dashboard.html` – The prebuilt application only accounts for two basic visualizations. The entire layout of the page can be adjusted in this file. • `static/js/core.js` – This script captures the formatted output of the Flask server's `/data` route and uses Plotly to create visualizations. You can add or update the page's charts.	Web developer

(Optional) Deploy the updated application

Task	Description	Skills required
Update your application's requirements file.	Before sending changes to Elastic Beanstalk, update the `requirements.txt` file to reflect any new Python modules by running the following command in your application's root directory:	Python developer

Task	Description	Skills required
	`pip freeze > requirements.txt`	
Redeploy the Elastic Beanstalk environment.	To ensure that your application changes are reflected in your Elastic Beanstalk deployment, navigate to your application's root directory and run the following command: `eb deploy` This sends the most recent version of the application's code to your existing Elastic Beanstalk deployment.	Systems administrator, Architect

Related resources

- Call an Amazon SageMaker model endpoint using Amazon API Gateway and AWS Lambda
- Deploying a Flask application to Elastic Beanstalk
- EB CLI command reference
- Setting up your Python development environment

Additional information

Troubleshooting list

The following are six common errors and their solutions.

Error 1

```
Unable to assume role "arn:aws:iam::xxxxxxxxxx:role/aws-elasticbeanstalk-ec2-role".
 Verify that the role exists and is configured correctly.
```

Solution: If this error occurs when you run `eb create`, create a sample application on the Elastic Beanstalk console to create the default instance profile. For more information about this, see [Creating an Elastic Beanstalk environment](#) in the AWS Elastic Beanstalk documentation.

Error 2

```
Your WSGIPath refers to a file that does not exist.
```

Solution: This error occurs in deployment logs because Elastic Beanstalk expects the Flask code to be named `application.py`. If you chose a different name, run `eb config` and edit the WSGIPath as shown in the following code sample:

```
aws:elasticbeanstalk:container:python:
     NumProcesses: '1'
     NumThreads: '15'
     StaticFiles: /static/=static/
     WSGIPath: application.py
```

Make sure that you replace `application.py` with your file name.

You can also leverage Gunicorn and a Procfile. For more information about this approach, see [Configuring the WSGI server with a Procfile](#) in the AWS Elastic Beanstalk documentation.

Error 3

```
Target WSGI script '/opt/python/current/app/application.py' does not contain WSGI
 application 'application'.
```

Solution: Elastic Beanstalk expects the variable that represents your Flask application to be named `application`. Make sure that the `application.py` file uses `application` as the variable name:

```
application = Flask(__name__)
```

Error 4

```
The EB CLI cannot find your SSH key file for keyname
```

Solution: Use the EB CLI to specify which key pair to use or to create a key pair for your deployment's EC2 instances. To resolve the error, run `eb init -i` and one of the options will ask:

```
Do you want to set up SSH for your instances?
```

Respond with Y to either create a key pair or specify an existing key pair.

Error 5

I've updated my code and redeployed but my deployment is not reflecting my changes.

Solution: If you're using a Git repository with your deployment, make sure that you add and commit your changes before redeploying.

Error 6

You are previewing the Flask application from an AWS Cloud9 IDE and run into errors.

Solution: For more information about this, see Previewing running applications in the AWS Cloud9 IDE in the AWS Cloud9 documentation.

Natural language processing using Amazon Comprehend

By choosing to use Amazon Comprehend, you can detect custom entities in individual text documents by running real-time analysis or asynchronous batch jobs. Amazon Comprehend also enables you to train custom entity recognition and text classification models that can be used in real time by creating an endpoint.

This pattern uses asynchronous batch jobs to detect sentiments and entities from an input file that contains multiple documents. The sample application provided by this pattern is designed for users to upload a .csv file containing a single column with one text document per row. The `comprehend_helper.py` file in the GitHub Visualize AI/ML model results using Flask and AWS Elastic Beanstalk repository reads the input file and sends the input to Amazon Comprehend for processing.

BatchDetectEntities

Amazon Comprehend inspects the text of a batch of documents for named entities and returns the detected entity, location, type of entity, and a score that indicates Amazon Comprehend's level of confidence. A maximum of 25 documents can be sent in one API call, with each document smaller than 5,000 bytes in size. You can filter the results to show only certain entities based on the use case. For example, you could skip the `'quantity'` entity type and set a threshold score for the detected entity (for example, 0.75). We recommend that you explore the results for your specific use case before choosing a threshold value. For more information about this, see BatchDetectEntities in the Amazon Comprehend documentation.

BatchDetectSentiment

Amazon Comprehend inspects a batch of incoming documents and returns the prevailing sentiment for each document (`POSITIVE`, `NEUTRAL`, `MIXED`, or `NEGATIVE`). A maximum of 25 documents can be sent in one API call, with each document smaller than 5,000 bytes in size. Analyzing the sentiment is straightforward and you choose the sentiment with the highest score to be displayed in the final results. For more information about this, see BatchDetectSentiment in the Amazon Comprehend documentation.

Flask configuration handling

Flask servers use a series of configuration variables to control how the server runs. These variables can contain debug output, session tokens, or other application settings. You can also define custom variables that can be accessed while the application is running. There are multiple approaches for setting configuration variables.

In this pattern, the configuration is defined in `config.py` and inherited within `application.py`.

- **Note**

  `config.py` contains the configuration variables that are set up on the application's startup. In this application, a `DEBUG` variable is defined to tell the application to run the server in debug mode. : Debug mode should not be used when running an application in a production environment. `UPLOAD_FOLDER` is a custom variable that is defined to be referenced later in the application and inform it where uploaded user data should be stored.
- `application.py` initiates the Flask application and inherits the configuration settings defined in `config.py`. This is performed by the following code:

```
application = Flask(__name__)
application.config.from_pyfile('config.py')
```

More patterns

- Automate AWS infrastructure operations by using Amazon Bedrock
- Deploy agentic systems on Amazon Bedrock with the CrewAI framework by using Terraform
- Deploy a ChatOps solution to manage SAST scan results by using Amazon Q Developer in chat applications custom actions and AWS CloudFormation
- Generate data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Generate Db2 z/OS data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Give SageMaker notebook instances temporary access to a CodeCommit repository in another AWS account
- Manage AWS Organizations policies as code by using AWS CodePipeline and Amazon Bedrock
- Modernize the CardDemo mainframe application by using AWS Transform
- Perform advanced analytics using Amazon Redshift ML
- Streamline Amazon EC2 compliance management with Amazon Bedrock agents and AWS Config
- Streamline Amazon Lex bot development and deployment by using an automated workflow
- Troubleshoot states in AWS Step Functions by using Amazon Bedrock

Analytics

Topics

- Analyze Amazon Redshift data in Microsoft SQL Server Analysis Services
- Analyze and visualize nested JSON data with Amazon Athena and Amazon QuickSight
- Automate data ingestion from AWS Data Exchange into Amazon S3
- Automate encryption enforcement in AWS Glue using an AWS CloudFormation template
- Build a data pipeline to ingest, transform, and analyze Google Analytics data using the AWS DataOps Development Kit
- Build a video processing pipeline by using Amazon Kinesis Video Streams and AWS Fargate
- Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue
- Build an enterprise data mesh with Amazon DataZone, AWS CDK, and AWS CloudFormation
- Calculate value at risk (VaR) by using AWS services
- Configure cross-account access to a shared AWS Glue Data Catalog using Amazon Athena
- Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL
- Convert the Teradata RESET WHEN feature to Amazon Redshift SQL
- Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code
- Enforce tagging of Amazon EMR clusters at launch
- Ensure Amazon EMR logging to Amazon S3 is enabled at launch
- Generate test data using an AWS Glue job and Python
- Cost-effectively ingest IoT data directly into Amazon S3 using AWS IoT Greengrass
- Launch a Spark job in a transient EMR cluster using a Lambda function
- Migrate Apache Cassandra workloads to Amazon Keyspaces by using AWS Glue
- Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator
- Migrate Oracle Business Intelligence 12c to the AWS Cloud from on-premises servers
- Migrate an ELK Stack to Elastic Cloud on AWS
- Migrate data to the AWS Cloud by using Starburst
- Optimize the ETL ingestion of input file size on AWS
- Orchestrate an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions

- Perform advanced analytics using Amazon Redshift ML
- Query Amazon DynamoDB tables with SQL by using Amazon Athena
- Access, query, and join Amazon DynamoDB tables using Athena
- Set up a minimum viable data space to share data between organizations
- Set up language-specific sorting for Amazon Redshift query results using a scalar Python UDF
- Subscribe a Lambda function to event notifications from S3 buckets in different AWS Regions
- Three AWS Glue ETL job types for converting data to Apache Parquet
- Visualize Amazon Redshift audit logs using Amazon Athena and Amazon QuickSight
- Visualize IAM credential reports for all AWS accounts using Amazon QuickSight
- More patterns

Analyze Amazon Redshift data in Microsoft SQL Server Analysis Services

Sunil Vora, Amazon Web Services

Summary

This pattern describes how to connect and analyze Amazon Redshift data in Microsoft SQL Server Analysis Services, by using the Intellisoft OLE DB Provider or CData ADO.NET Provider for database access.

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. SQL Server Analysis Services is an online analytical processing (OLAP) tool that you can use to analyze data from data marts and data warehouses such as Amazon Redshift. You can use SQL Server Analysis Services to create OLAP cubes from your data for rapid, advanced data analysis.

Prerequisites and limitations

Assumptions

- This pattern describes how to set up SQL Server Analysis Services and Intellisoft OLE DB Provider or CData ADO.NET Provider for Amazon Redshift on an Amazon Elastic Compute Cloud (Amazon EC2) instance. Alternatively, you can install both on a host in your corporate data center.

Prerequisites

- An active AWS account
- An Amazon Redshift cluster with credentials

Architecture

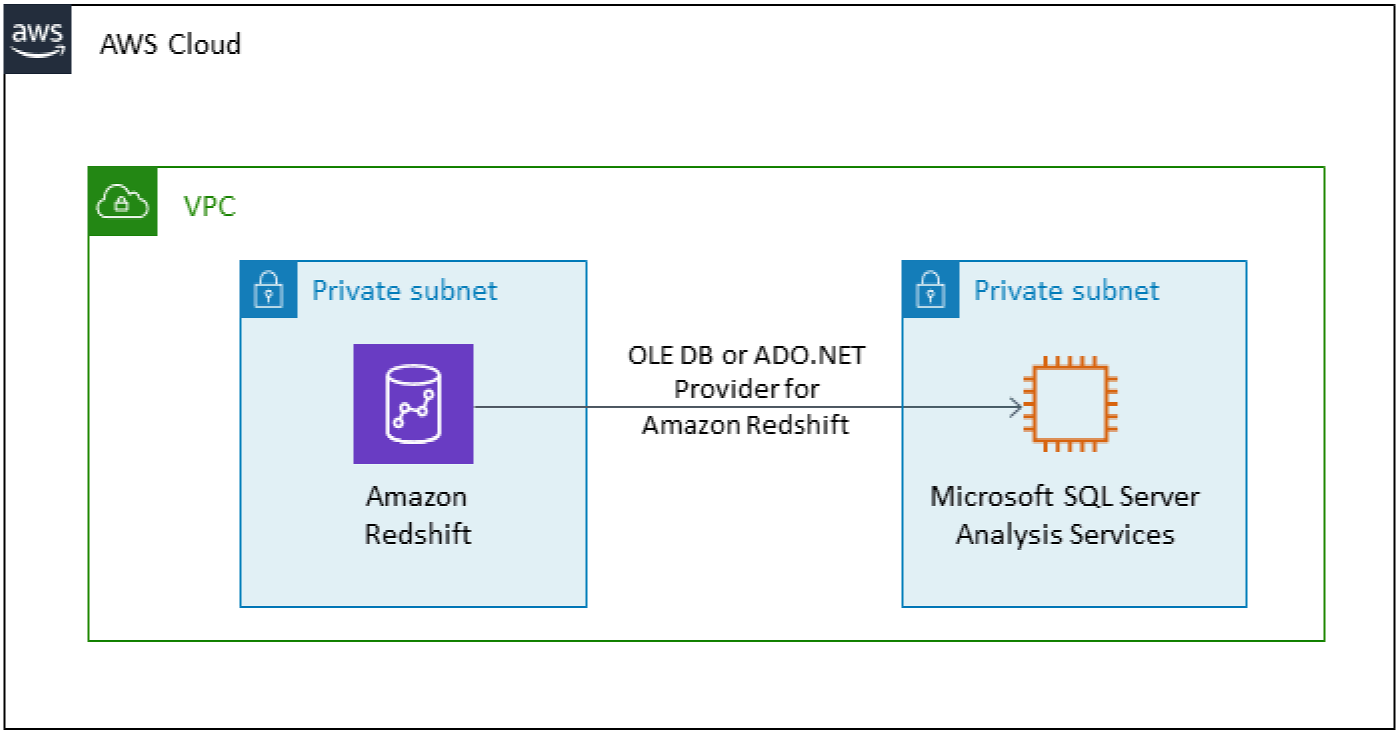
Source technology stack

- An Amazon Redshift cluster

Target technology stack

- Microsoft SQL Server Analysis Services

Source and target architecture



Tools

- Microsoft Visual Studio 2019 (Community Edition)
- Intellisoft OLE DB Provider for Amazon Redshift (Trial) or CData ADO.NET Provider for Amazon Redshift (Trial)

Epics

Analyze tables

Task	Description	Skills required
Analyze the tables and data to be imported.	Identify the Amazon Redshift tables to be imported and their sizes.	DBA

Set up EC2 instance and install tools

Task	Description	Skills required
Set up an EC2 instance.	In your AWS account, create an EC2 instance in a private or public subnet.	Systems administrator
Install tools for database access.	Download and install the Intellisoft OLE DB Provider for Amazon Redshift (or CData ADO.NET Provider for Amazon Redshift).	Systems administrator
Install Visual Studio.	Download and install Visual Studio 2019 (Community Edition).	Systems administrator
Install extensions.	Install the **Microsoft Analysis Services Projects** extension in Visual Studio.	Systems administrator
Create a project.	Create a new tabular model project in Visual Studio to store your Amazon Redshift data. In Visual Studio, choose the **Analysis Services Tabular Project** option when creating your project.	DBA

Create data source and import tables

Task	Description	Skills required
Create an Amazon Redshift data source.	Create an Amazon Redshift data source by using the Intellisoft OLE DB Provider	Amazon Redshift, DBA

Task	Description	Skills required
	for Amazon Redshift (or CData ADO.NET Provider for Amazon Redshift) and your Amazon Redshift credentials.	
Import tables.	Select and import tables and views from Amazon Redshift into your SQL Server Analysis Services project.	Amazon Redshift, DBA

Clean up after migration

Task	Description	Skills required
Delete the EC2 instance.	Delete the EC2 instance you launched previously.	Systems administrator

Related resources

- Amazon Redshift (AWS documentation)
- Install SQL Server Analysis Services (Microsoft documentation)
- Tabular Model Designer (Microsoft documentation)
- Overview of OLAP cubes for advanced analytics (Microsoft documentation)
- Microsoft Visual Studio 2019 (Community Edition)
- Intellisoft OLE DB Provider for Amazon Redshift (Trial)
- CData ADO.NET Provider for Amazon Redshift (Trial)

Analyze and visualize nested JSON data with Amazon Athena and Amazon QuickSight

Anoop Singh, Amazon Web Services

Summary

This pattern explains how to translate a nested, JSON-formatted data structure into a tabular view by using Amazon Athena, and then visualize the data in Amazon QuickSight.

You can use JSON-formatted data for API-powered data feeds from operational systems to create data products. This data can also help you understand your customers and their interactions with your products better, so you can tailor user experiences and predict outcomes.

Prerequisites and limitations

Prerequisites

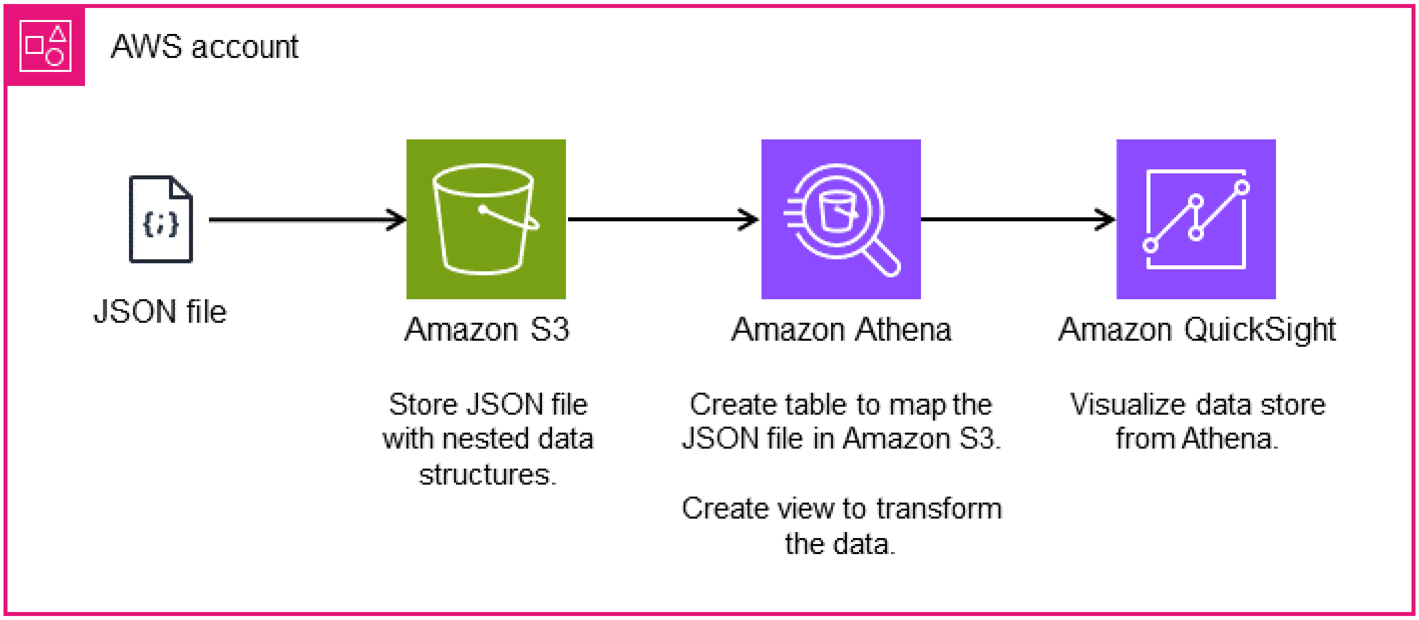
- An active AWS account
- A JSON file that represents a nested data structure (this pattern provides a sample file)

Limitations:

- JSON features integrate well with existing SQL-oriented functions in Athena. However, they aren't ANSI SQL compatible, and the JSON file is expected to carry each record on a separate line. You might need to use the `ignore.malformed.json` property in Athena to indicate if malformed JSON records should be turned into null characters or generate errors. For more information, see Best practices for reading JSON data in the Athena documentation.
- This pattern considers only simple and small amounts of JSON-formatted data. If you want to use these concepts at scale, consider applying data partitioning and consolidate your data into larger files.

Architecture

The following diagram shows the architecture and workflow for this pattern. The nested data structures are stored in Amazon Simple Storage Service (Amazon S3) in JSON format. In Athena, the JSON data is mapped to an Athena data structure. You then create a view to analyze the data, and visualize the data structure in QuickSight.



Tools

AWS services

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data. This pattern uses Amazon S3 to store the JSON file.
- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL. This pattern uses Athena to query and transform the JSON data. With a few actions in the AWS Management Console, you can point Athena at your data in Amazon S3 and use standard SQL to run one-time queries. Athena is serverless, so there is no infrastructure to set up or manage, and you pay only for the queries that you run. Athena scales automatically and runs queries in parallel, so results are fast, even with large datasets and complex queries.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data on a single dashboard. QuickSight lets you easily create and publish interactive dashboards that include machine learning (ML) insights. You can access these dashboards from any device, and embed them into your applications, portals, and websites.

Example code

The following JSON file provides a nested data structure that you can use in this pattern.


```
{
  "symbol": "AAPL",
  "financials": [
    {
      "reportDate": "2017-03-31",
      "grossProfit": 20591000000,
      "costOfRevenue": 32305000000,
      "operatingRevenue": 52896000000,
      "totalRevenue": 52896000000,
      "operatingIncome": 14097000000,
      "netIncome": 11029000000,
      "researchAndDevelopment": 2776000000,
      "operatingExpense": 6494000000,
      "currentAssets": 101990000000,
      "totalAssets": 334532000000,
      "totalLiabilities": 200450000000,
      "currentCash": 15157000000,
      "currentDebt": 13991000000,
      "totalCash": 67101000000,
      "totalDebt": 98522000000,
      "shareholderEquity": 134082000000,
      "cashChange": -1214000000,
      "cashFlow": 12523000000,
      "operatingGainsLosses": null
    }
  ]
}
```


Epics

Set up an S3 bucket

Task	Description	Skills required
Create an S3 bucket.	To create a bucket to store the JSON file, sign in to the AWS Management Console, open the [Amazon S3 console](https://console.aws.amazon.com/s3/), and then choose **Create**	Systems administrator

Task	Description	Skills required
	bucket. For more information, see Creating a bucket in the Amazon S3 documentation.	
Add the nested JSON data.	Upload your JSON file to the S3 bucket. For a sample JSON file, see the previous section. For instructions, see Uploading objects in the Amazon S3 documentation.	Systems administrator

Analyze data in Athena

Task	Description	Skills required

Create a table for mapping the JSON data.	1. Open the Athena console.
	2. Create a database by following the instructions in the Athena documentation.
	3. From the **Database** menu, choose the database that you created.
	4. In the query editor, enter a `CREATE TABLE` statement such as the following:
	
	`CREATE EXTERNAL TABLE financials_json (`
	`symbol string,`
	`financials array<`
	`struct<reportdate: string,`
	`grossprofit: bigint,`
		Developer

Task	Description	Skills required

	``
	                totalreve
	nue: bigint,
	                totalcash
	: bigint,
	                totaldebt
	: bigint,
	                researcha
	nddevelopment:
	 bigint>>
	)
	ROW FORMAT SERDE
	 'org.openx.data.js
	onserde.JsonSerDe'
	 LOCATION 's3://s3b
	ucket-for-athena/'
	``
	where `LOCATION` specifies the location of the S3 bucket that contains the JSON file.
	5. Choose **Run** to create the table.
	
	For more information about creating tables, see the [Athena documentation].	

Task	Description	Skills required
Create a view for data analysis.	1. Open the [Athena console](). 2. Create a database by following the instructions in the [Athena documentation](). 3. From the **Database** menu, choose the database that you created. 4. In the query editor, enter a `CREATE VIEW` statement such as the following: `CREATE OR REPLACE VIEW financial_json_view AS SELECT symbol, financials[1].reportdate one_report_date, -- indexes start with 1 financials[1].totalrevenue one_total_revenue, financials[1].reportdate another_report_date, financials[1].totalrevenue another_total_revenue FROM financials_json where symbol='AAPL' ORDER BY 1` 5. Choose **Run** to create the view.	Developer

Task	Description	Skills required
	For more information about creating views, see the Athena documentation.	
Analyze and validate the data.	1. Open the Athena console. 2. In the query editor, run queries by using the view that you created in the previous step. 3. Validate the data against the JSON file, to confirm that column names and data types are mapped correctly.	Developer

Visualize data in QuickSight

Task	Description	Skills required
Set up Athena as a data source in QuickSight.	1. Open the QuickSight console. 2. Choose **Datasets**, **New dataset**. 3. Choose Athena as a data source. 4. Choose the database that includes the view that you created. 5. Choose the view that you want to create a dataset for.	Systems administrator

Task	Description	Skills required
	6. On the **Finish data set creation** page, choose **Directly query your data**. 7. Choose **Visualize**.	
Visualize data in QuickSight.	1. After you visualize the dataset, choose the visuals from the left pane, and choose fields for the dataset. For more informati on, see the tutorial in the QuickSight documentation. 2. Save the changes to the analysis. 3. Choose **Publish dashboard** to publish the visuals that you created.	Data analyst

Related resources

- Amazon Athena documentation
- Amazon QuickSight tutorials
- Working with nested JSON (blog post)

Automate data ingestion from AWS Data Exchange into Amazon S3

Adnan Alvee and Manikanta Gona, Amazon Web Services

Summary

This pattern provides an AWS CloudFormation template that enables you to automatically ingest data from AWS Data Exchange into your data lake in Amazon Simple Storage Service (Amazon S3).

AWS Data Exchange is a service that makes it easy to securely exchange file-based data sets in the AWS Cloud. AWS Data Exchange data sets are subscription-based. As a subscriber, you can also access data set revisions as providers publish new data.

The AWS CloudFormation template creates an event in Amazon CloudWatch Events and an AWS Lambda function. The event watches for any updates to the data set you have subscribed to. If there is an update, CloudWatch initiates a Lambda function, which copies the data over to the S3 bucket you specify. When the data has been copied successfully, Lambda sends you an Amazon Simple Notification Service (Amazon SNS) notification.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Subscription to a data set in AWS Data Exchange

Limitations

- The AWS CloudFormation template must be deployed separately for each subscribed data set in AWS Data Exchange.

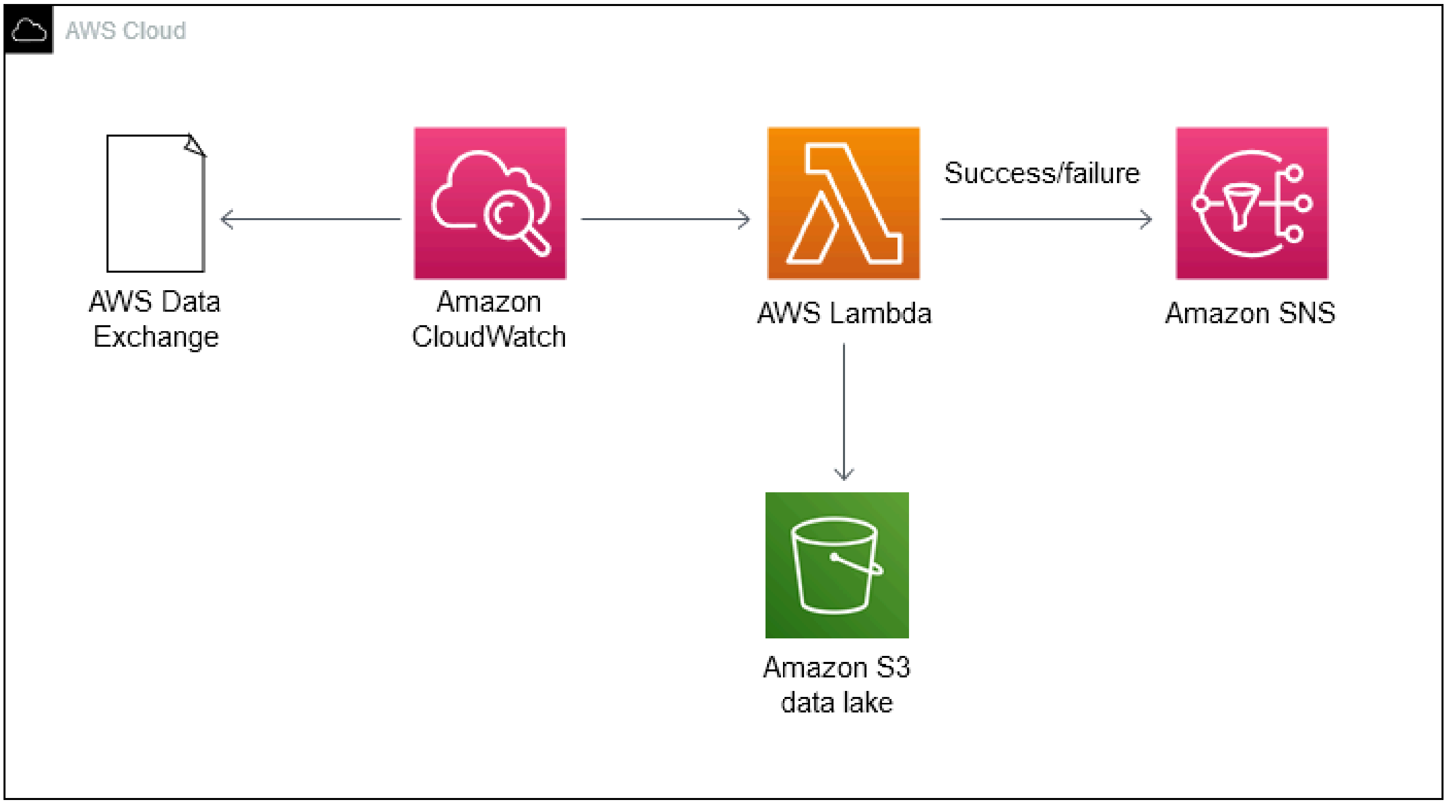
Architecture

Target technology stack

- AWS Lambda
- Amazon S3

- AWS Data Exchange
- Amazon CloudWatch
- Amazon SNS

Target architecture



Automation and scale

You can use the AWS CloudFormation template multiple times for the data sets you want to ingest into the data lake.

Tools

- AWS Data Exchange makes it easy for AWS customers to securely exchange file-based data sets in the AWS Cloud. As a subscriber, you can find and subscribe to hundreds of products from qualified data providers. Then, you can quickly download the data set or copy it to Amazon S3 for use across a variety of AWS analytics and machine learning services. Anyone with an AWS account can be an AWS Data Exchange subscriber.
- AWS Lambda lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per

second. You pay only for the compute time you consume; there is no charge when your code isn't running. With Lambda, you can run code for virtually any type of application or backend service with zero administration. Lambda runs your code on a high-availability compute infrastructure and manages all the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring, and logging.

- Amazon S3 provides storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. Using simple rules that you can quickly set up, you can match events and route them to one or more target functions or streams. CloudWatch Events becomes aware of operational changes as they occur. It responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information. You can also use CloudWatch Events to schedule automated actions that self-initiate at certain times using **cron** or **rate** expressions.
- Amazon Simple Notification Service (Amazon SNS) enables applications, end-users, and devices to instantly send and receive notifications from the cloud. Amazon SNS provides topics (communication channels) for high-throughput, push-based, many-to-many messaging. Using Amazon SNS topics, publishers can distribute messages to a large number of subscribers for parallel processing, including Amazon Simple Queue Service (Amazon SQS) queues, Lambda functions, and HTTP/S webhooks. You can also use Amazon SNS to send notifications to end users using mobile push, SMS, and email.

Epics

Subscribe to a data set

Task	Description	Skills required
Subscribe to a data set.	In the AWS Data Exchange console, subscribe to a dataset. For instructions, see Subscribing to data products on AWS Data Exchange in the AWS documentation.	General AWS

Task	Description	Skills required
Note the data set attributes.	Note the AWS Region, ID, and revision ID for the data set. You will need this for the AWS CloudFormation template in the next step.	General AWS

Deploy the AWS CloudFormation template

Task	Description	Skills required
Create an S3 bucket and folder.	If you already have a data lake in Amazon S3, create a folder to store the data to ingest from AWS Data Exchange. If you are deploying the template for testing purposes, create a new S3 bucket, and note the bucket name and folder prefix for the next step.	General AWS
Deploy the AWS CloudForm ation template.	Deploy the AWS CloudForm ation template that's provided as an attachment to this pattern. For instructions, see the AWS CloudFormation documentation. Configure the following parameters to correspond to your AWS account, data set, and S3 bucket settings: **Dataset AWS Region**, **Dataset ID**, **Revision ID**, **S3 Bucket Name** (for example, `DOC-`	General AWS

Task	Description	Skills required
	`EXAMPLE-BUCKET`), **Folder Prefix** (for example, `myfolder/`), and **Email for SNS Notification**. You can set the **Dataset Name** parameter to any name. When you deploy the template, it runs a Lambda function to automatically ingest the first set of data available in the data set. Subsequent ingestion then takes place automatically, as new data arrives in the data set.	

Related resources

- Subscribing to data products on AWS Data Exchange (AWS Data Exchange documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automate encryption enforcement in AWS Glue using an AWS CloudFormation template

Diogo Guedes, Amazon Web Services

Summary

This pattern shows you how to set up and automate encryption enforcement in AWS Glue by using an AWS CloudFormation template. The template creates all the required configurations and resources for enforcing encryption. These resources include an initial configuration, a preventive control created by an Amazon EventBridge rule, and an AWS Lambda function.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Permissions to deploy the CloudFormation template and its resources

Limitations

This security control is regional. You must deploy the security control in each AWS Region where you want to set up encryption enforcement in AWS Glue.

Architecture

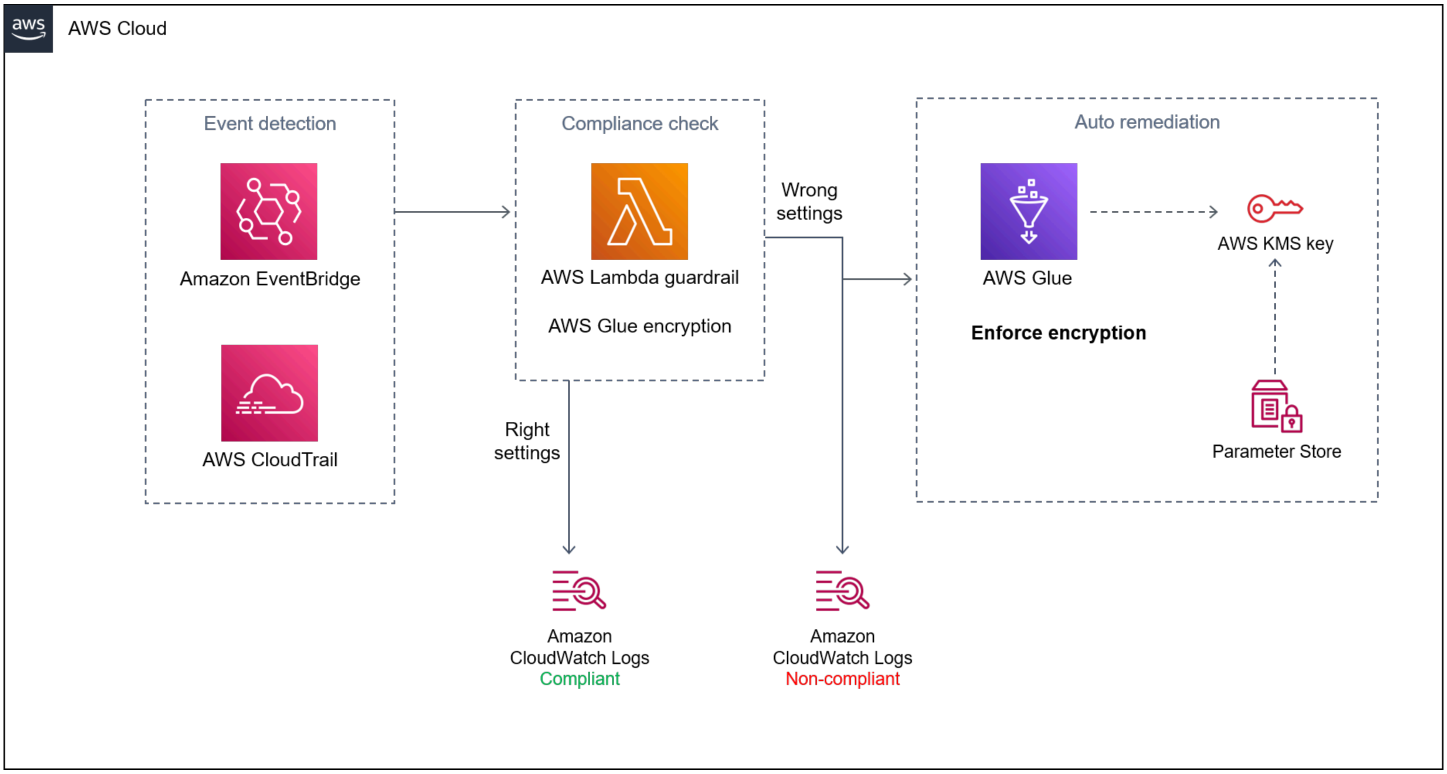
Target technology stack

- Amazon CloudWatch Logs (from AWS Lambda)
- Amazon EventBridge rule
- AWS CloudFormation stack
- AWS CloudTrail
- AWS Identity and Access Management (IAM) managed role and policy
- AWS Key Management Service (AWS KMS)
- AWS KMS alias

- AWS Lambda function
- AWS Systems Manager Parameter Store

Target architecture

The following diagram shows how to automate encryption enforcement in AWS Glue.



The diagram shows the following workflow:

1. A CloudFormation template creates all the resources, including the initial configuration and detective control for encryption enforcement in AWS Glue.
2. An EventBridge rule detects a state change in the encryption configuration.
3. A Lambda function is invoked for evaluation and logging through CloudWatch Logs. For non-compliant detection, the Parameter Store is recovered with an Amazon Resource Name (ARN) for an AWS KMS key. The service is remediated to compliant status with encryption enabled.

Automation and scale

If you're using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts where you want to enable encryption enforcement in AWS Glue.

Tools

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CloudTrail helps you enable operational and risk auditing, governance, and compliance of your AWS account.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.

Code

The code for this pattern is available in the GitHub aws-custom-guardrail-event-driven repository.

Best practices

AWS Glue supports data encryption at rest for authoring jobs in AWS Glue and developing scripts using development endpoints.

Consider the following best practices:

- Configure ETL jobs and development endpoints to use AWS KMS keys to write encrypted data at rest.

- Encrypt the metadata stored in the AWS Glue Data Catalog by using keys that you manage through AWS KMS.
- Use AWS KMS keys to encrypt job bookmarks and the logs generated by crawlers and ETL jobs.

Epics

Launch the CloudFormation template

Task	Description	Skills required
Deploy the CloudFormation template.	Download the `aws-custom-guardrail-event-driven.yaml` template from the GitHub repository, and then deploy the template. The `CREATE_COMPLETE` status indicates that your template was successfully deployed. **Note** The template requires no input parameters.	Cloud architect

Verify the encryption settings in AWS Glue

Task	Description	Skills required
Check the AWS KMS key configurations.	1. Sign in to the AWS Management Console and then open the AWS Glue console.	Cloud architect

Task	Description	Skills required
	2. In the navigation pane, under **Data Catalog**, choose **Catalog settings**. 3. Verify that the **Metadata encryption** and **Encrypt connection passwords** settings are flagged and configured to use `KMSKeyGlue` .	

Test the encryption enforcement

Task	Description	Skills required
Identify the encryption setting in CloudFormation.	1. Sign in to the AWS Management Console and then open the [CloudFormation console](https://console.aws.amazon.com/cloudformation/). 2. In the navigation pane, choose **Stacks**, and then choose your stack. 3. Choose the **Resources** tab. 4. In the **Resources** table, find the encryption setting by **Logical ID**.	Cloud architect
Switch the provisioned infrastructure to an uncompliant state.	1. Sign in to the AWS Management Console and then open the [AWS Glue console](https://console.aws.amazon.com/glue/). 2. In the navigation pane, under **Data Catalog**, choose **Catalog settings**.	Cloud architect

Task	Description	Skills required
	3. Clear the **Metadata encryption** check box. 4. Clear the **Encrypt connection passwords** check box. 5. Choose **Save**. 6. Refresh the AWS Glue console. The guardrail detects the uncompliant state in AWS Glue after you clear the check boxes, and then enforces compliance by automatically remediating the encryption misconfiguration. As a result, the encryption check boxes should again be selected after you refresh the page.	

Related resources

- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail (Amazon CloudWatch documentation)
- Setting up encryption in AWS Glue (AWS Glue documentation)

Build a data pipeline to ingest, transform, and analyze Google Analytics data using the AWS DataOps Development Kit

Anton Kukushkin and Rudy Puig, Amazon Web Services

Summary

This pattern describes how to build a data pipeline to ingest, transform, and analyze Google Analytics data by using the AWS DataOps Development Kit (AWS DDK) and other AWS services. The AWS DDK is an open-source development framework that helps you build data workflows and modern data architecture on AWS. One of the main objectives of the AWS DDK is to save you the time and effort that's typically devoted to labor-intensive data pipeline tasks, such as orchestrating pipelines, building infrastructure, and creating the DevOps behind that infrastructure. You can offload these labor-intensive tasks to AWS DDK so that you can focus on writing code and other high-value activities.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon AppFlow connector for Google Analytics, configured
- Python and pip (Python's package manager)
- Git, installed and configured
- AWS Command Line Interface (AWS CLI), installed and configured
- AWS Cloud Development Kit (AWS CDK), installed

Product versions

- Python 3.7 or later
- pip 9.0.3 or later

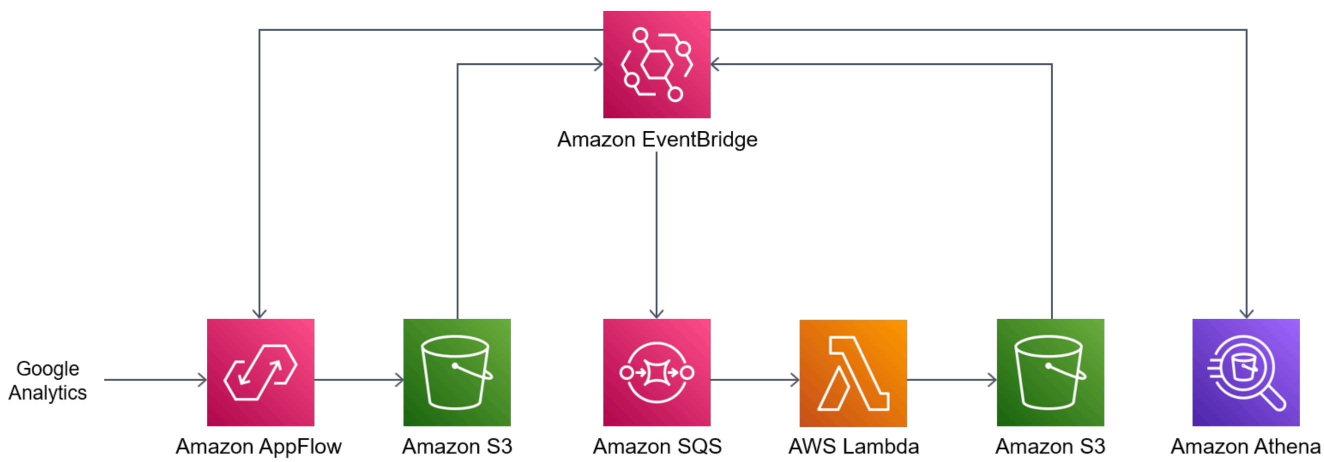
Architecture

Technology stack

- Amazon AppFlow
- Amazon Athena
- Amazon CloudWatch
- Amazon EventBridge
- Amazon Simple Storage Service (Amazon S3)
- Amazon Simple Queue Service (Amazon SQS)
- AWS DataOps Development Kit (AWS DDK)
- AWS Lambda

Target architecture

The following diagram shows the event-driven process that ingests, transforms, and analyzes Google Analytics data.



The diagram shows the following workflow:

1. An Amazon CloudWatch scheduled event rule invokes Amazon AppFlow.
2. Amazon AppFlow ingests Google Analytics data into an S3 bucket.
3. After the data is ingested by the S3 bucket, event notifications in EventBridge are generated, captured by a CloudWatch Events rule, and then put into an Amazon SQS queue.
4. A Lambda function consumes events from the Amazon SQS queue, reads the respective S3 objects, transforms the objects to Apache Parquet format, writes the transformed objects to the S3 bucket, and then creates or updates the AWS Glue Data Catalog table definition.
5. An Athena query runs against the table.

Tools

AWS tools

- Amazon AppFlow is a fully-managed integration service that enables you to securely exchange data between software as a service (SaaS) applications.
- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Cloud Development Kit (AWS CDK) is a framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- AWS DataOps Development Kit (AWS DDK) is an open-source development framework to help you build data workflows and modern data architecture on AWS.

Code

The code for this pattern is available in the GitHub AWS DataOps Development Kit (AWS DDK) and Analyzing Google Analytics data with Amazon AppFlow, Amazon Athena, and AWS DataOps Development Kit repositories.

Epics

Prepare the environment

Task	Description	Skills required
Clone the source code.	To clone the source code, run the following command: `git clone https://github.com/aws-samples/aws-ddk-examples.git`	DevOps engineer
Create a virtual environment.	Navigate to the source code directory, and then run the following command to create a virtual environment: `cd google-analytics-data-using-appflow/python && python3 -m venv .venv`	DevOps engineer
Install the dependencies.	To activate the virtual environment and install the dependencies, run the following command: `source .venv/bin/activate && pip install -r requirements.txt`	DevOps engineer

Deploy the application that uses your data pipeline

Task	Description	Skills required
Bootstrap the environment.	1. Confirm that the AWS CLI is set up with valid credentials for your AWS account. For more information, see [Using named profiles](#) in the AWS CLI documentation. 2. Run the `cdk bootstrap --profile [AWS_PROFILE]` command.	DevOps engineer
Deploy the data.	To deploy the data pipeline, run the `cdk deploy --profile [AWS_PROFILE]` command.	DevOps engineer

Test the deployment

Task	Description	Skills required
Validate stack status.	1. Open the [AWS CloudFormation console](#). 2. On the **Stacks** page, confirm that the status of the stack `DdkAppflowAthenaStack` is `CREATE_COMPLETE`.	DevOps engineer

Troubleshooting

Issue	Solution
Deployment fails during the creation of an `AWS::AppFlow::Flow` resource and you receive the following error: `Connector Profile with name ga-connection does not exist`	Confirm that you created an Amazon AppFlow connector for Google Analytics and named it `ga-connection`. For instructions, see Google Analytics in the Amazon AppFlow documentation.

Related resources

- AWS DataOps Development Kit (AWS DDK) (GitHub)
- AWS DDK Examples (GitHub)

Additional information

AWS DDK data pipelines are composed of one or many stages. In the following code examples, you use `AppFlowIngestionStage` to ingest data from Google Analytics, `SqsToLambdaStage` to handle data transformation, and `AthenaSQLStage` to run the Athena query.

First, the data transformation and ingestion stages are created, as the following code example shows:

```
        appflow_stage = AppFlowIngestionStage(
            self,
            id="appflow-stage",
            flow_name=flow.flow_name,
        )
        sqs_lambda_stage = SqsToLambdaStage(
            self,
            id="lambda-stage",
            lambda_function_props={
                "code": Code.from_asset("./ddk_app/lambda_handlers"),
                "handler": "handler.lambda_handler",
                "layers": [
                    LayerVersion.from_layer_version_arn(
```

```
                         self,
                         id="layer",
                         layer_version_arn=f"arn:aws:lambda:
{self.region}:336392948345:layer:AWSDataWrangler-Python39:1",
                     )
                 ],
                 "runtime": Runtime.PYTHON_3_9,
             },
         )
         # Grant lambda function S3 read & write permissions
         bucket.grant_read_write(sqs_lambda_stage.function)
         # Grant Glue database & table permissions
         sqs_lambda_stage.function.add_to_role_policy(
             self._get_glue_db_iam_policy(database_name=database.database_name)
         )
         athena_stage = AthenaSQLStage(
             self,
             id="athena-sql",
             query_string=[
                 (
                     "SELECT year, month, day, device, count(user_count) as cnt "
                     f"FROM {database.database_name}.ga_sample "
                     "GROUP BY year, month, day, device "
                     "ORDER BY cnt DESC "
                     "LIMIT 10; "
                 )
             ],
             output_location=Location(
                 bucket_name=bucket.bucket_name, object_key="query-results/"
             ),
             additional_role_policy_statements=[
                 self._get_glue_db_iam_policy(database_name=database.database_name)
             ],
         )
```

Next, the `DataPipeline` construct is used to "wire" the stages together by using EventBridge rules, as the following code example shows:

```
         (
             DataPipeline(self, id="ingestion-pipeline")
             .add_stage(
                 stage=appflow_stage,
                 override_rule=Rule(
```

```
                    self,
                    "schedule-rule",
                    schedule=Schedule.rate(Duration.hours(1)),
                    targets=appflow_stage.targets,
                ),
            )
            .add_stage(
                stage=sqs_lambda_stage,
                # By default, AppFlowIngestionStage stage emits an event after the flow
 run finishes successfully
                # Override rule below changes that behavior to call the the stage when
 data lands in the bucket instead
                override_rule=Rule(
                    self,
                    "s3-object-created-rule",
                    event_pattern=EventPattern(
                        source=["aws.s3"],
                        detail={
                            "bucket": {"name": [bucket.bucket_name]},
                            "object": {"key": [{"prefix": "ga-data"}]},
                        },
                        detail_type=["Object Created"],
                    ),
                    targets=sqs_lambda_stage.targets,
                ),
            )
            .add_stage(stage=athena_stage)
        )
```

For more code examples, see the GitHub Analyzing Google Analytics data with Amazon AppFlow, Amazon Athena, and AWS DataOps Development Kit repository.

Build a video processing pipeline by using Amazon Kinesis Video Streams and AWS Fargate

Piotr Chotkowski and Pushparaju Thangavel, Amazon Web Services

Summary

This pattern demonstrates how to use Amazon Kinesis Video Streams and AWS Fargate to extract frames from a video stream and store them as image files for further processing in Amazon Simple Storage Service (Amazon S3).

The pattern provides a sample application in the form of a Java Maven project. This application defines the AWS infrastructure by using the AWS Cloud Development Kit (AWS CDK). Both the frame processing logic and the infrastructure definitions are written in the Java programming language. You can use this sample application as a basis for developing your own real-time video processing pipeline or to build the video preprocessing step of a machine learning pipeline.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Java SE Development Kit (JDK) 11, installed
- Apache Maven, installed
- AWS Cloud Development Kit (AWS CDK), installed
- AWS Command Line Interface (AWS CLI) version 2, installed
- Docker (required for building Docker images to use in AWS Fargate task definitions), installed

Limitations

This pattern is intended as a proof of concept, or as a basis for further development. It should not be used in its current form in production deployments.

Product versions

- This pattern was tested with the AWS CDK version 1.77.0 (see AWS CDK versions)

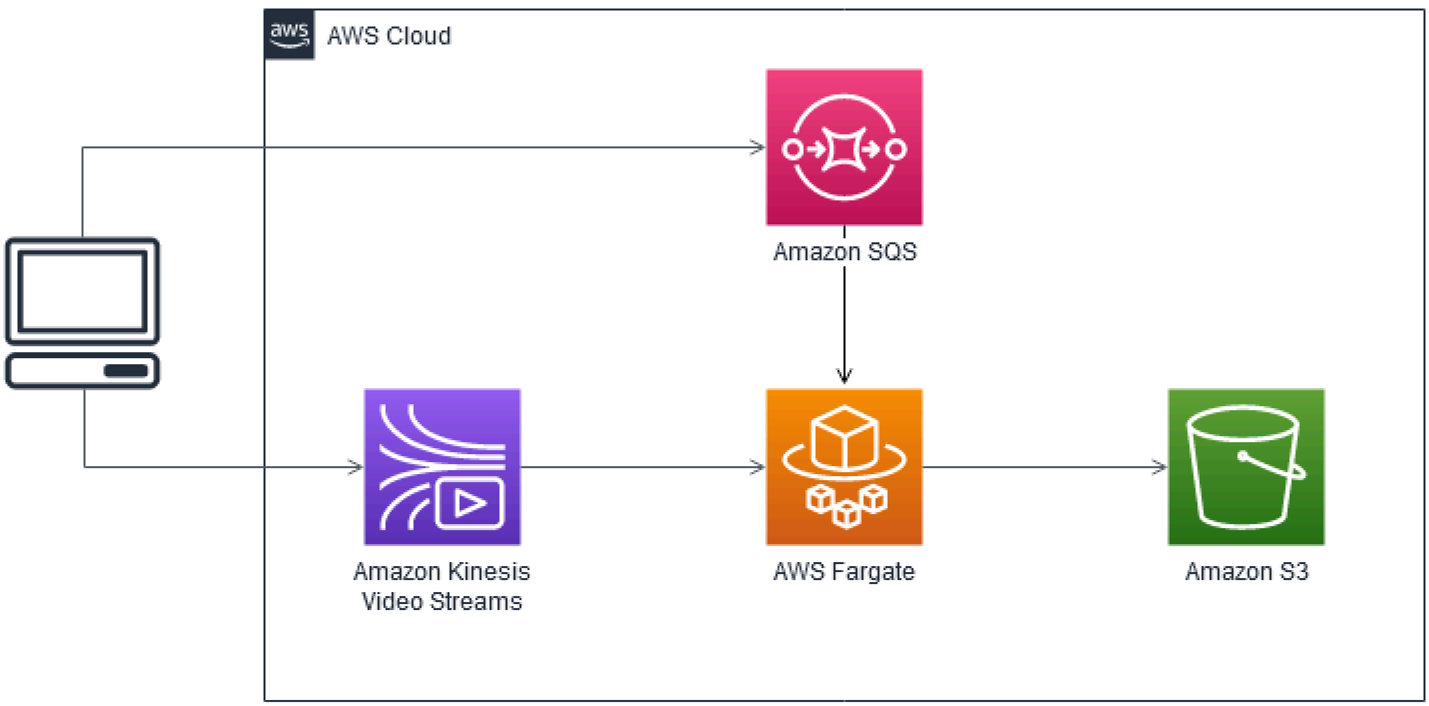
- JDK 11
- AWS CLI version 2

Architecture

Target technology stack

- Amazon Kinesis Video Streams
- AWS Fargate task
- Amazon Simple Queue Service (Amazon SQS) queue
- Amazon S3 bucket

Target architecture



The user creates a Kinesis video stream, uploads a video, and sends a JSON message that contains details about the input Kinesis video stream and the output S3 bucket to an SQS queue. AWS Fargate, which is running the main application in a container, pulls the message from the SQS queue and starts extracting frames. Each frame is saved in an image file and stored in the target S3 bucket.

Automation and scale

The sample application can scale both horizontally and vertically within a single AWS Region. Horizontal scaling can be achieved by increasing the number of deployed AWS Fargate tasks that read from the SQS queue. Vertical scaling can be achieved by increasing the number of frame-splitting and image-publishing threads in the application. These settings are passed as environment variables to the application in the definition of the QueueProcessingFargateService resource in the AWS CDK. Due to the nature of AWS CDK stack deployment, you can deploy this application in multiple AWS Regions and accounts with no additional effort.

Tools

Tools

- AWS CDK is a software development framework for defining your cloud infrastructure and resources by using programming languages such as TypeScript, JavaScript, Python, Java, and C#/.Net.
- Amazon Kinesis Video Streams is a fully managed AWS service that you can use to stream live video from devices to the AWS Cloud, or build applications for real-time video processing or batch-oriented video analytics.
- AWS Fargate is a serverless compute engine for containers. Fargate removes the need to provision and manage servers, and lets you focus on developing your applications.
- Amazon S3 is an object storage service that offers scalability, data availability, security, and performance.
- Amazon SQS is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications.

Code

- A .zip file of the sample application project (`frame-splitter-code.zip`) is attached.

Epics

Deploy the infrastructure

Task	Description	Skills required
Start the Docker daemon.	Start the Docker daemon on your local system. The AWS	Developer, DevOps engineer

Task	Description	Skills required
	CDK uses Docker to build the image that is used in the AWS Fargate task. You must run Docker before you proceed to the next step.	
Build the project.	Download the `frame-splitter-code` sample application (attached) and extract its contents into a folder on your local machine. Before you can deploy the infrastructure, you have to build the [Java Maven](https://maven.apache.org/) project. At a command prompt, navigate to the root directory of the project, and build the project by running the command: `mvn clean install`	Developer, DevOps engineer

Task	Description	Skills required
Bootstrap the AWS CDK.	(First-time AWS CDK users only) If this is the first time you're using the AWS CDK, you might have to bootstrap the environment by running the AWS CLI command: `cdk bootstrap --profile "$AWS_PROFILE_NAME"` where `$AWS_PROFILE_NAME` holds the name of the AWS profile from your AWS credentials. Or, you can remove this parameter to use the default profile. For more information, see the AWS CDK documentation.	Developer, DevOps engineer

Task	Description	Skills required
Deploy the AWS CDK stack.	In this step, you create the required infrastructure resources (SQS queue, S3 bucket, AWS Fargate task definition) in your AWS account, build the Docker image that is required for the AWS Fargate task, and deploy the application. At a command prompt, navigate to the root directory of the project, and run the command: `cdk deploy --profile "$AWS_PROFILE_NAME" --all` where `$AWS_PROFILE_NAME` holds the name of the AWS profile from your AWS credentials. Or, you can remove this parameter to use the default profile. Confirm the deployment. Note the **QueueUrl** and **Bucket** values from the CDK deployment output; you will need these in later steps. The AWS CDK creates the assets, uploads them to your AWS account, and creates all infrastructure resources. You can observe the resource creation process in the [AWS CloudFormation](https://console.aws.amazon.com/cloudformation)	Developer, DevOps engineer

Task	Description	Skills required
	console. For more information, see the AWS CloudFormation documentation and the AWS CDK documentation.	

Task	Description	Skills required
Create a video stream.	In this step, you create a Kinesis video stream that will serve as an input stream for video processing. Make sure that you have the AWS CLI installed and configured. In the AWS CLI, run: `aws kinesisvideo --profile "$AWS_PROFILE_NAME" create-stream --stream-name "$STREAM_NAME" --data-retention-in-hours "24"` where `$AWS_PROFILE_NAME` holds the name of the AWS profile from your AWS credentials (or remove this parameter to use the default profile) and `$STREAM_NAME` is any valid stream name. Alternatively, you can create a video stream by using the Kinesis console by following the steps in the [Kinesis Video Streams documentation](). Note the AWS Resource Name (ARN) of the created stream; you will need it later.	Developer, DevOps engineer

Run an example

Task	Description	Skills required
Upload the video to the stream.	In the project folder for the sample `frame-splitter-code` application, open the `ProcessingTaskTest.java` file in the `src/test/java/amazon/awscdk/examples/splitter` folder. Replace the `profileName` and `streamName` variables with the values you used in the previous steps. To upload the example video to the Kinesis video stream you created in the previous step, run: `amazon.awscdk.examples.splitter.ProcessingTaskTest#testExample test` Alternatively, you can upload your video by using one of the methods described in the [Kinesis Video Streams documentation].	Developer, DevOps engineer
Initiate video processing.	Now that you have uploaded a video to the Kinesis video stream, you can start processing it. To initiate the processing logic, you have to send a message with	Developer, DevOps engineer

Task	Description	Skills required

	details to the SQS queue that the AWS CDK created during deployment. To send a message by using the AWS CLI, run:
	`aws sqs --profile "$AWS_PROFILE_NAME" send-message --queue-url QUEUE_URL --message-body MESSAGE`
	where `$AWS_PROFILE_NAME` holds the name of the AWS profile from your AWS credentials (remove this parameter to use the default profile), `QUEUE_URL` is the **QueueUrl** value from the AWS CDK output, and `MESSAGE` is a JSON string in the following format:
	`{ "streamARN": "STREAM_ARN", "bucket": "BUCKET_NAME", "s3Directory": "test-output" }`
	where `STREAM_ARN` is the ARN of of the video stream you created in an earlier step and `BUCKET_NAME` is the **Bucket** value from the AWS CDK output.	

Task	Description	Skills required
	Sending this message initiates video processing. Alternatively, you can send a message by using the Amazon SQS console, as described in the Amazon SQS documentation.	
View images of the video frames.	You can see the resulting images in the S3 output bucket `s3://BUCKET_NAME/test-output` where `BUCKET_NAME` is the **Bucket** value from the AWS CDK output.	Developer, DevOps engineer

Related resources

- AWS CDK documentation
- AWS CDK API reference
- AWS CDK introductory workshop
- Amazon Kinesis Video Streams documentation
- Example: Identifying Objects in Video Streams Using SageMaker
- Example: Parsing and Rendering Kinesis Video Streams Fragments
- Analyze live video at scale in real time using Amazon Kinesis Video Streams and Amazon SageMaker (AWS Machine Learning blog post)
- AWS Fargate Getting Started

Additional information

Choosing an IDE

We recommend that you use your favorite Java IDE to build and explore this project.

Cleaning up

After you finish running this example, remove all deployed resources to avoid incurring additional AWS infrastructure costs.

To remove the infrastructure and the video stream, use these two commands in the AWS CLI:

```
cdk destroy --profile "$AWS_PROFILE_NAME" --all
```

```
aws kinesisvideo --profile "$AWS_PROFILE_NAME" delete-stream --stream-arn "$STREAM_ARN"
```

Alternatively, you can remove the resources manually by using the AWS CloudFormation console to remove the AWS CloudFormation stack, and the Kinesis console to remove the Kinesis video stream. Note that `cdk destroy` doesn't remove the output S3 bucket or the images in Amazon Elastic Container Registry (Amazon ECR) repositories (`aws-cdk/assets`). You have to remove them manually.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue

Rohan Jamadagni and Arunabha Datta, Amazon Web Services

Summary

This pattern provides guidance on how to configure Amazon Simple Storage Service (Amazon S3) for optimal data lake performance, and then load incremental data changes from Amazon S3 into Amazon Redshift by using AWS Glue, performing extract, transform, and load (ETL) operations.

The source files in Amazon S3 can have different formats, including comma-separated values (CSV), XML, and JSON files. This pattern describes how you can use AWS Glue to convert the source files into a cost-optimized and performance-optimized format like Apache Parquet. You can query Parquet files directly from Amazon Athena and Amazon Redshift Spectrum. You can also load Parquet files into Amazon Redshift, aggregate them, and share the aggregated data with consumers, or visualize the data by using Amazon QuickSight.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An S3 source bucket that has the right privileges and contains CSV, XML, or JSON files.

Assumptions

- The CSV, XML, or JSON source files are already loaded into Amazon S3 and are accessible from the account where AWS Glue and Amazon Redshift are configured.
- Best practices for loading the files, splitting the files, compression, and using a manifest are followed, as discussed in the [Amazon Redshift documentation](https://docs.aws.amazon.com/redshift/latest/dg/t_Loading-data-from-S3.html).
- The source file structure is unaltered.
- The source system is able to ingest data into Amazon S3 by following the folder structure defined in Amazon S3.
- The Amazon Redshift cluster spans a single Availability Zone. (This architecture is appropriate because AWS Lambda, AWS Glue, and Amazon Athena are serverless.) For high availability, cluster snapshots are taken at a regular frequency.

Limitations

- The file formats are limited to those that are currently supported by AWS Glue.
- Real-time downstream reporting isn't supported.

Architecture

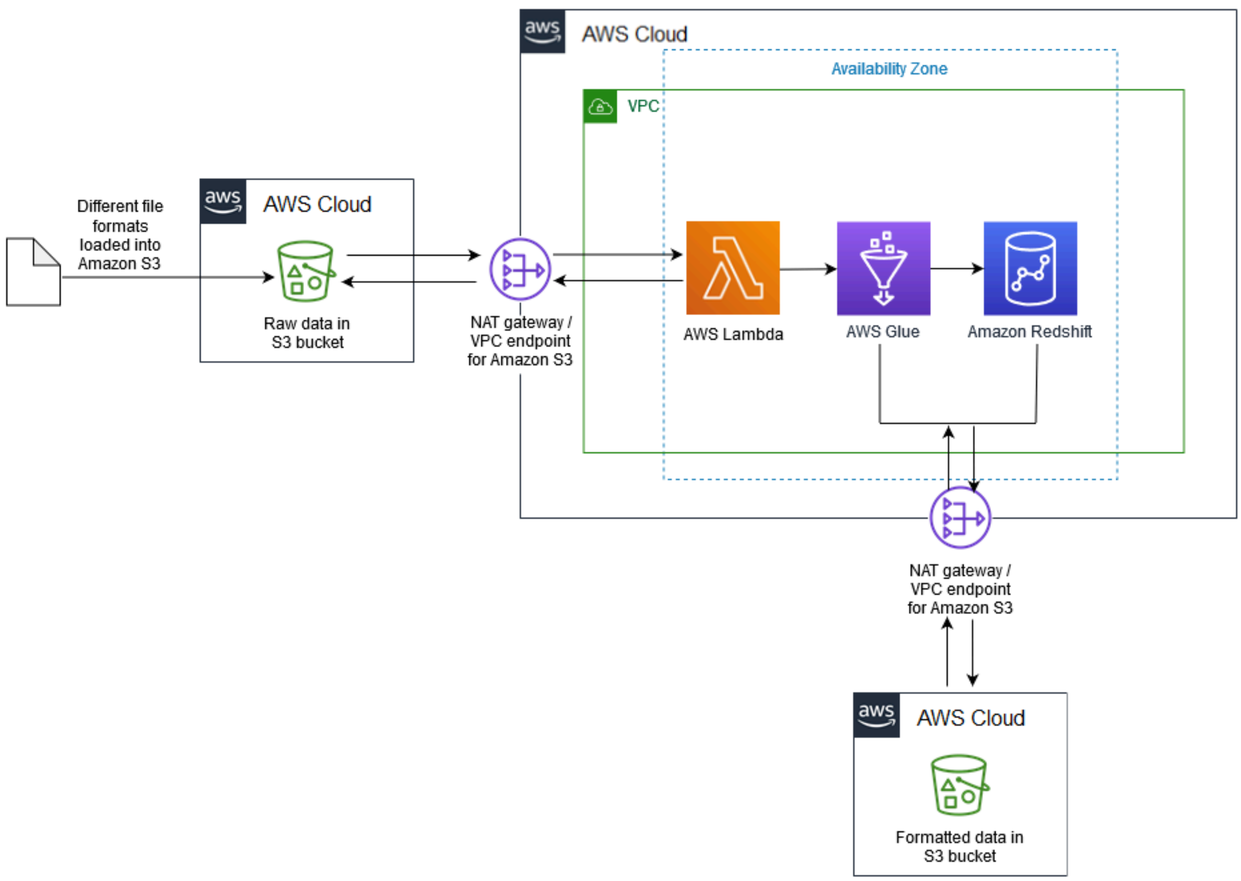
Source technology stack

- S3 bucket with CSV, XML, or JSON files

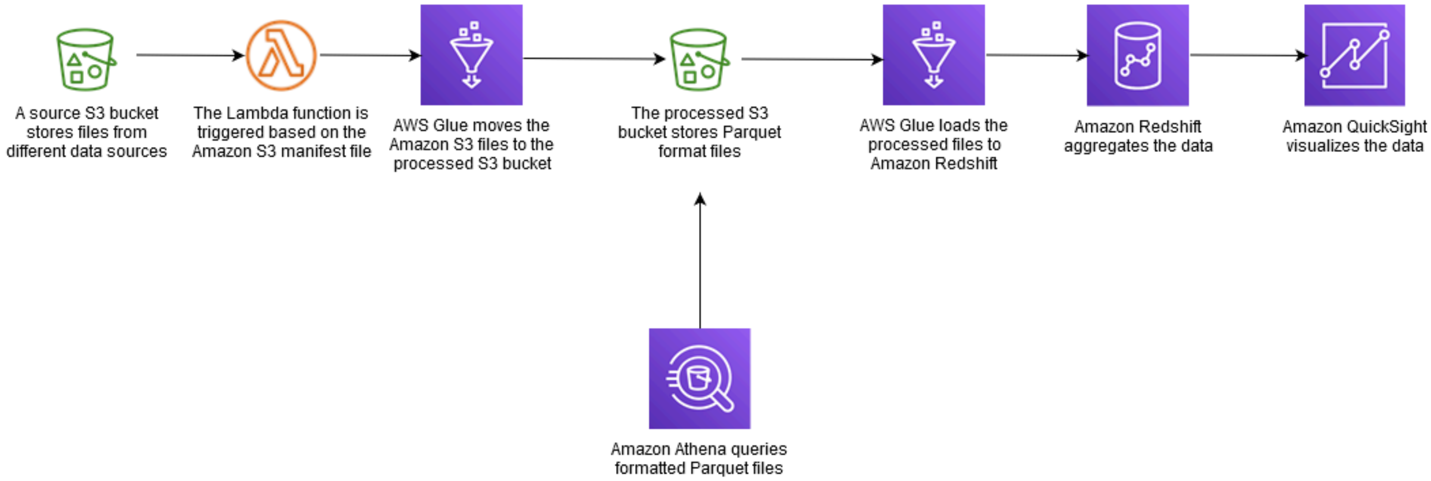
Target technology stack

- S3 data lake (with partitioned Parquet file storage)
- Amazon Redshift

Target architecture



Data flow



Tools

- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service. Amazon S3 can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- AWS Lambda – AWS Lambda lets you run code without provisioning or managing servers. AWS Lambda is an event-driven service; you can set up your code to automatically initiate from other AWS services.
- Amazon Redshift – Amazon Redshift is a fully managed, petabyte-scale data warehouse service. With Amazon Redshift, you can query petabytes of structured and semi-structured data across your data warehouse and your data lake using standard SQL.
- AWS Glue – AWS Glue is a fully managed ETL service that makes it easier to prepare and load data for analytics. AWS Glue discovers your data and stores the associated metadata (for example, table definitions and schema) in the AWS Glue Data Catalog. Your cataloged data is immediately searchable, can be queried, and is available for ETL.
- AWS Secrets Manager – AWS Secrets Manager facilitates protection and central management of secrets needed for application or service access. The service stores database credentials, API keys, and other secrets, and eliminates the need to hardcode sensitive information in plaintext format. Secrets Manager also offers key rotation to meet security and compliance needs. It has built-in integration for Amazon Redshift, Amazon Relational Database Service (Amazon RDS), and Amazon DocumentDB. You can store and centrally manage secrets by using the Secrets Manager console, the command-line interface (CLI), or Secrets Manager API and SDKs.

- [Amazon Athena](#) – Amazon Athena is an interactive query service that makes it easy to analyze data that's stored in Amazon S3. Athena is serverless and integrated with AWS Glue, so it can directly query the data that's cataloged using AWS Glue. Athena is elastically scaled to deliver interactive query performance.

Epics

Create the S3 buckets and folder structure

Task	Description	Skills required
Analyze source systems for data structure and attributes.	Perform this task for each data source that contributes to the Amazon S3 data lake.	Data engineer
Define the partition and access strategy.	This strategy should be based on the frequency of data captures, delta processin g, and consumption needs. Make sure that S3 buckets are not open to the public and that access is controlle d by specific service role-based policies only. For more information, see the [Amazon S3 documentation](#).	Data engineer
Create separate S3 buckets for each data source type and a separate S3 bucket per source for the processed (Parquet) data.	Create a separate bucket for each source, and then create a folder structure that's based on the source system's data ingestion frequency ; for example, `s3://source-system-name/date/hour` . For the processed (converted to Parquet format) files, create a similar structure	Data engineer

Task	Description	Skills required
	; for example, `s3://source-processed-bucket/date/hour` . For more information about creating S3 buckets, see the [Amazon S3 documentation]().	

Create a data warehouse in Amazon Redshift

Task	Description	Skills required
Launch the Amazon Redshift cluster with the appropriate parameter groups and maintenance and backup strategy.	Use the Secrets Manager database secret for admin user credentials while creating the Amazon Redshift cluster. For information about creating and sizing an Amazon Redshift cluster, see the [Amazon Redshift documentation]() and the [Sizing Cloud Data Warehouses]() whitepaper.	Data engineer
Create and attach the IAM service role to the Amazon Redshift cluster.	The AWS Identity and Access Management (IAM) service role ensures access to Secrets Manager and the source S3 buckets. For more information, see the AWS documentation on [authorization]() and [adding a role]().	Data engineer
Create the database schema.	Follow Amazon Redshift best practices for table design. Based on the use case, choose	Data engineer

Task	Description	Skills required
	the appropriate sort and distribution keys, and the best possible compression encoding. For best practices, see the AWS documentation.	
Configure workload management.	Configure workload management (WLM) queues, short query acceleration (SQA), or concurrency scaling, depending on your requirements. For more information, see Implementing workload management in the Amazon Redshift documentation.	Data engineer

Create a secret in Secrets Manager

Task	Description	Skills required
Create a new secret to store the Amazon Redshift sign-in credentials in Secrets Manager.	This secret stores the credentials for the admin user as well as individual database service users. For instructions, see the Secrets Manager documentation. Choose **Amazon Redshift Cluster** as the secret type. Additionally, on the **Secret rotation** page, turn on the rotation. This will create the appropriate user in the Amazon Redshift cluster and will rotate the key secrets at defined intervals.	Data engineer

Task	Description	Skills required
Create an IAM policy to restrict Secrets Manager access.	Restrict Secrets Manager access to only Amazon Redshift administrators and AWS Glue.	Data engineer

Configure AWS Glue

Task	Description	Skills required
In the AWS Glue Data Catalog, add a connection for Amazon Redshift.	For instructions, see the [AWS Glue documentation].	Data engineer
Create and attach an IAM service role for AWS Glue to access Secrets Manager, Amazon Redshift, and S3 buckets.	For more information, see the [AWS Glue documentation].	Data engineer
Define the AWS Glue Data Catalog for the source.	This step involves creating a database and required tables in the AWS Glue Data Catalog. You can either use a crawler to catalog the tables in the AWS Glue database, or define them as Amazon Athena external tables. You can also access the external tables defined in Athena through the AWS Glue Data Catalog. See the AWS documentation for more information about [defining the Data Catalog] and	Data engineer

Task	Description	Skills required
	creating an external table in Athena.	
Create an AWS Glue job to process source data.	The AWS Glue job can be a Python shell or PySpark to standardize, deduplicate, and cleanse the source data files. To optimize performance and avoid having to query the entire S3 source bucket, partition the S3 bucket by date, broken down by year, month, day, and hour as a pushdown predicate for the AWS Glue job. For more information, see the AWS Glue documentation. Load the processed and transformed data to the processed S3 bucket partitions in Parquet format. You can query the Parquet files from Athena.	Data engineer
Create an AWS Glue job to load data into Amazon Redshift.	The AWS Glue job can be a Python shell or PySpark to load the data by upserting the data, followed by a complete refresh. For details, see the AWS Glue documentation and the *Additional information* section.	Data engineer

Task	Description	Skills required
(Optional) Schedule AWS Glue jobs by using triggers as necessary.	The incremental data load is primarily driven by an Amazon S3 event that causes an AWS Lambda function to call the AWS Glue job. Use AWS Glue trigger-based scheduling for any data loads that demand time-based instead of event-based scheduling.	Data engineer

Create a Lambda function

Task	Description	Skills required
Create and attach an IAM service-linked role for AWS Lambda to access S3 buckets and the AWS Glue job.	Create an IAM service-linked role for AWS Lambda with a policy to read Amazon S3 objects and buckets, and a policy to access the AWS Glue API to start an AWS Glue job. For more information, see the [Knowledge Center]().	Data engineer
Create a Lambda function to run the AWS Glue job based on the defined Amazon S3 event.	The Lambda function should be initiated by the creation of the Amazon S3 manifest file. The Lambda function should pass the Amazon S3 folder location (for example, source_bucket/year/month/date/hour) to the AWS Glue job as a parameter. The AWS Glue job will use this	Data engineer

Task	Description	Skills required
	parameter as a pushdown predicate to optimize file access and job processin g performance. For more information, see the AWS Glue documentation.	
Create an Amazon S3 PUT object event to detect object creation, and call the respective Lambda function.	The Amazon S3 PUT object event should be initiated only by the creation of the manifest file. The manifest file controls the Lambda function and the AWS Glue job concurrency, and processes the load as a batch instead of processin g individual files that arrive in a specific partition of the S3 source bucket. For more information, see the Lambda documentation.	Data engineer

Related resources

- Amazon S3 documentation
- AWS Glue documentation
- Amazon Redshift documentation
- AWS Lambda
- Amazon Athena
- AWS Secrets Manager

Additional information

Detailed approach for upsert and complete refresh

Upsert: This is for datasets that require historical aggregation, depending on the business use case. Follow one of the approaches described in Updating and inserting new data (Amazon Redshift documentation) based on your business needs.

Complete refresh: This is for small datasets that don't need historical aggregations. Follow one of these approaches:

1. Truncate the Amazon Redshift table.
2. Load the current partition from the staging area

or:

1. Create a temporary table with current partition data.
2. Drop the target Amazon Redshift table.
3. Rename the temporary table to the target table.

Build an enterprise data mesh with Amazon DataZone, AWS CDK, and AWS CloudFormation

Dhrubajyoti Mukherjee, Ravi Kumar, Weizhou Sun, and Adjoa Taylor, Amazon Web Services

Summary

On Amazon Web Services (AWS), customers understand that data is the key to accelerating innovation and driving business value for their enterprise. To manage this massive data, you can adopt a decentralized architecture such as data mesh. A data mesh architecture facilitates product thinking, a mindset that takes customers, goals, and the market into account. Data mesh also helps to establish a federated governance model that provides fast, secure access to your data.

Strategies for building a data mesh-based enterprise solution on AWS discusses how you can use the Data Mesh Strategy Framework to formulate and implement a data mesh strategy for your organization. By using the Data Mesh Strategy Framework, you can optimize the organization of teams and their interactions to accelerate your data mesh journey.

This document provides guidance on how to build an enterprise data mesh with Amazon DataZone. Amazon DataZone is a data management service for cataloging, discovering, sharing, and governing data stored across AWS, on premises, and third-party sources. The pattern includes code artifacts that help you deploy the data mesh–based data solution infrastructure using AWS Cloud Development Kit (AWS CDK) and AWS CloudFormation. This pattern is intended for cloud architects and DevOps engineers.

For information about the objectives of this pattern and the solution scope, see the Additional information section.

Prerequisites and limitations

Prerequisites

- A minimum of two active AWS accounts: one for the central governance account and another for the member account
- AWS administrator credentials for the central governance account in your development environment
- AWS Command Line Interface (AWS CLI) installed to manage your AWS services from the command line

- Node.js and Node Package Manager (npm) installed to manage AWS CDK applications
- AWS CDK Toolkit installed globally in your development environment by using npm, to synthesize and deploy AWS CDK applications

```
npm install -g aws-cdk
```

- Python version 3.12 installed in your development environment
- TypeScript installed in your development environment or installed globally by using npm compiler:

```
npm install -g typescript
```

- Docker installed in your development environment
- A version control system such as Git to maintain the source code of the solution (recommended)
- An integrated development environment (IDE) or text editor with support for Python and TypeScript (strongly recommended)

Limitations

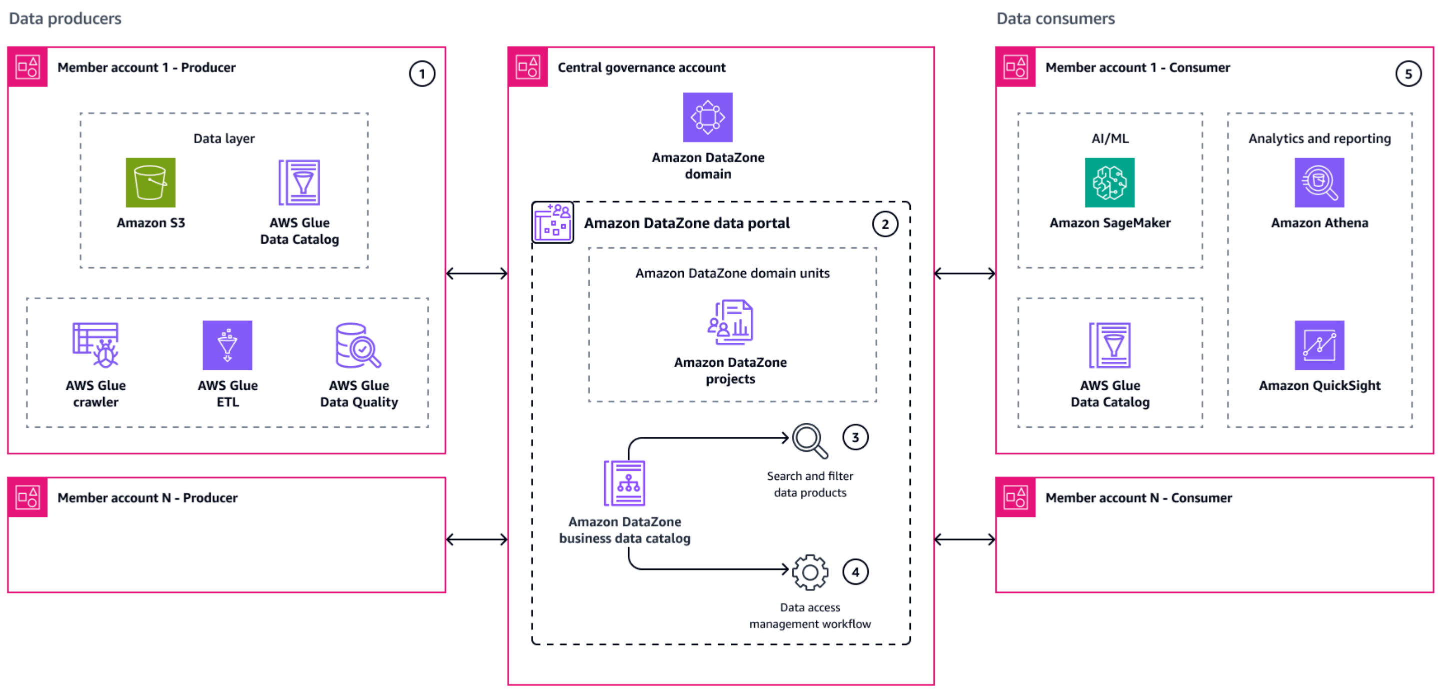
- The solution has been tested only on machines that are running Linux or macOS.
- In the current version, the solution doesn't support the integration of Amazon DataZone and AWS IAM Identity Center by default. However, you can configure it to support this integration.

Product versions

- Python version 3.12

Architecture

The following diagram shows a data mesh reference architecture. The architecture is based on Amazon DataZone and uses Amazon Simple Storage Service (Amazon S3) and AWS Glue Data Catalog as data sources. The AWS services that you use with Amazon DataZone in your data mesh implementation might differ, based on your organization's requirements.



1. In the producer accounts, raw data is either fit for consumption in its current form or it's transformed for consumption by using AWS Glue. The technical metadata for the data is stored in Amazon S3 and is evaluated using a AWS Glue data crawler. The data quality is measured by using AWS Glue Data Quality. The source database in the Data Catalog is registered as an asset in the Amazon DataZone catalog. The Amazon DataZone catalog is hosted in the central governance account using Amazon DataZone data source jobs.
2. The central governance account hosts the Amazon DataZone domain and the Amazon DataZone data portal. The AWS accounts of the data producers and consumers are associated with the Amazon DataZone domain. The Amazon DataZone projects of the data producers and consumers are organized under the corresponding Amazon DataZone domain units.
3. End users of the data assets log into the Amazon DataZone data portal by using their AWS Identity and Access Management (IAM) credentials or single sign-on (with integration through IAM Identity Center). They search, filter, and view asset information (for example, data quality information or business and technical metadata) in the Amazon DataZone data catalog.
4. After an end user finds the data asset that they want, they use the Amazon DataZone subscription feature to request access. The data owner on the producer team receives a notification and evaluates the subscription request in the Amazon DataZone data portal. The data owner approves or rejects the subscription request based on its validity.
5. After the subscription request is granted and fulfilled, the asset is accessed in the consumer account for the following activities:

- AI/ML model development by using Amazon SageMaker AI
- Analytics and reporting by using Amazon Athena and Amazon QuickSight

Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon Simple Storage Service (Amazon S3) by using standard SQL.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- Amazon DataZone is a data management service that helps you catalog, discover, share, and govern data stored across AWS, on premises, and in third-party sources.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data in a single dashboard.
- Amazon SageMaker AI is a managed machine learning (ML) service that helps you build and train ML models and then deploy them into a production-ready hosted environment.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code repository

The solution is available in the GitHub data-mesh-datazone-cdk-cloudformation repository.

Epics

Set up the environment

Task	Description	Skills required
Clone the repository.	To clone the repository, run the following command in your local development environment (Linux or macOS): `git clone https://github.com/aws-samples/data-mesh-datazone-cdk-cloudformation`	Cloud architect, DevOps engineer

Create the environment.	To create the Python virtual environment, run the following commands:
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	`pip install -r requirements.txt`
		Cloud architect, DevOps engineer

Bootstrap the account.	To bootstrap the central governance account by using AWS CDK, run the following command: `cdk bootstrap aws://<GOVERNANCE_ACCOUNT_ID>/<AWS_REGION>`	Cloud architect, DevOps engineer

Task	Description	Skills required
	Sign in to the AWS Management Console, open the central governance account console, and get the Amazon Resource Name (ARN) of the AWS CDK execution role.	
Construct the `DzDataMeshMemberStackSet.yaml` file.	To construct the `DzDataMeshMemberStackSet.yaml` file, from the root directory of the repository, initiate the following bash script: `./lib/scripts/create_dz_data_mesh_member_stack_set.sh`	Cloud architect, DevOps engineer
Confirm template creation.	Ensure that the AWS CloudFormation template file is created at the `lib/cfn-templates/DzDataMeshMemberStackSet.yaml` location.	Cloud architect, DevOps engineer

Deploy resources in the central governance account

Task	Description	Skills required
Modify the configuration.	In the `config/Config.ts` file, modify the following parameters:	Cloud architect, DevOps engineer

Task	Description	Skills required

	`DZ_APPLICATION_NAME - Name of the application.`
	`DZ_STAGE_NAME - Name of the stage.`
	`DZ_DOMAIN_NAME - Name of the Amazon DataZone domain`
	`DZ_DOMAIN_DESCRIPTION - Description of the Amazon DataZone domain`
	`DZ_DOMAIN_TAG - Tag of the Amazon DataZone domain`
	`DZ_ADMIN_PROJECT_NAME - Name of the Amazon DataZone project for administrators`
	`DZ_ADMIN_PROJECT_DESCRIPTION - Description of the Amazon DataZone project for administrators`
	`CDK_EXEC_ROLE_ARN - ARN of the cdk execution role`
	`DZ_ADMIN_ROLE_ARN - ARN of the administrator role`
	
	Keep the remaining parameters empty.	

Task	Description	Skills required
Update the Amazon DataZone glossary configuration.	To update the Amazon DataZone glossary configuration in the `lib/utils/glossary_config.json` file, use the following example configuration: `{ "GlossaryName": "PII Data", "GlossaryDescription": "If data source contains PII attributes", "GlossaryTerms": [{ "Name": "Yes", "ShortDescription": "Yes", "LongDescription": "Yes Glossary Term" }, { "Name": "No", "ShortDescription": "No", "LongDescription": "No Glossary Term" } ] }`	Cloud architect, DevOps engineer

```
{
    "GlossaryName":
 "PII Data",
    "GlossaryDescripti
on": "If data source
 contains PII attribute
s",
    "GlossaryTerms": [{
            "Name":
 "Yes",
            "ShortDes
cription": "Yes",
            "LongDesc
ription": "Yes Glossary
 Term"
        },
        {
            "Name":
 "No",
            "ShortDes
cription": "No",
            "LongDesc
ription": "No Glossary
 Term"
        }
    ]
}
```

Task	Description	Skills required
Update the Amazon DataZone metadata form configuration.	To update the Amazon DataZone metadata form configuration in the `lib/utils/metadata_form_config.json` file, use the following example configuration: `{ "FormName": "ScheduleDataRefresh", "FormDescription": "Form for data refresh schedule", "FormSmithyModel": "@amazon.datazone#displayname(defaultName: \"Data Refresh Schedule\")\nstructure ScheduleDataRefresh {\n @documentation(\"Schedule of Data Refresh\")\n @required\n @amazon.datazone#searchable\n @amazon.datazone#displayname(defaultName: \"Data Refresh Schedule\")\n data_refresh_schedule: String\n}" }`	Cloud architect, DevOps engineer

Task	Description	Skills required

Export the AWS credentials.	To export AWS credentials to your development environment for the IAM role with administrative permissions, use the following format:
	`export AWS_ACCESS_KEY_ID=`
	`export AWS_SECRET_ACCESS_KEY=`
	`export AWS_SESSION_TOKEN=`
		Cloud architect, DevOps engineer

Synthesize the template.	To synthesize the AWS CloudFormation template, run the following command: `npx cdk synth`	Cloud architect, DevOps engineer
Deploy the solution.	To deploy the solution, run the following command: `npx cdk deploy --all`	Cloud architect, DevOps engineer

Configure new member accounts

Task	Description	Skills required
Deploy the template.	Deploy the AWS CloudFormation template located at `lib/cfn-templates/DzDataMeshCfnStackSetExecutionRole.yaml` in the member account	Cloud architect, DevOps engineer

Task	Description	Skills required
	with the following input parameters: • `GovernanceAccountID` – Account ID of the governance account • `DataZoneKMSKeyID` – ID of the AWS Key Management Service (AWS KMS) key that encrypts the Amazon DataZone metadata • `NotificationQueueName` – Name of the Amazon SQS notification queue in the governance account	
Update the ARNs.	To update the list of AWS CloudFormation StackSet execution role ARNs for the member accounts, use the following code: `DZ_MEMBER_STACK_SET_EXEC_ROLE_LIST - List of Stack set execution role arns for the member accounts.`	Cloud architect, DevOps engineer

Task	Description	Skills required

Synthesize and deploy.	To synthesize the AWS CloudFormation template and deploy the solution, run the following commands:
	`npx cdk synth`
	`npx cdk deploy --all`
		Cloud architect, DevOps engineer

Task	Description	Skills required
Associate the member account.	To associate the member account with the central governance account, do the following: 1. Sign in to the console of the central governance account, and open the Amazon DataZone console at https://console.aws.amazon.com/datazone/. 2. Choose the domain that you created. 3. Scroll to the **Associated accounts** tab, and choose **Request Association**. 4. Provide the AWS account ID, and choose **AWSRAMPermissionDataZonePortalReadWrite** as **RAM policy**. 5. Choose **Request Association**. 6. Wait until you receive an email notification that your account is successfully bootstrapped.	Cloud architect, DevOps engineer

Task	Description	Skills required
Update the parameters.	To update the member account–specific parameters in the config file at `config/Config.ts` , use the following format: `export const DZ_MEMBER_ACCOUNT_CONFIG: memberAccountConfig = { '123456789012' : { PROJECT_NAME: 'TEST-PROJECT-123456789012', PROJECT_DESCRIPTION: 'TEST-PROJECT-123456789012', PROJECT_EMAIL: 'user@xyz.com' } }`	Cloud architect, DevOps engineer

Synthesize and deploy the template.	To synthesize the AWS CloudFormation template and deploy the solution, run the following commands:
	`npx cdk synth`
	`npx cdk deploy --all`
		Cloud architect, DevOps engineer

Task	Description	Skills required
Add member accounts.	To create and configure additional member accounts in the data solution, repeat the previous steps for each member account. This solution doesn't different iate between data producers and consumers.	Cloud architect, DevOps engineer

Clean up

Task	Description	Skills required
Disassociate the member accounts.	To disassociate the accounts, do the following: 1. Sign in to the console and open the Amazon DataZone console. 2. Choose **View Domains**. 3. Select the domain that you created. 4. Choose the **Account associations** tab. 5. Select the member account that you want to disassoci ate. 6. Choose **Disassociate**, and enter `disassociate` to confirm. 7. Repeat steps 3-6 for all member accounts.	Cloud architect, DevOps engineer

Task	Description	Skills required
Delete the stack instances.	To delete the AWS CloudFormation stack instances, do the following: 1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/. 2. In the navigation pane, choose **StackSets**. 3. Choose the stack set named **StackSet-DataZone-DataMesh-Member**, and choose the **Stack instances** tab. 4. Copy the AWS account ID of the member account that you want to remove from membership. 5. Choose **Actions**, choose **Delete stacks from StackSet**, and keep the default options. 6. In the **Account numbers** field, enter the account ID. 7. In the **Specify regions** dropdown list, choose the AWS Region. 8. Choose **Next**, and then choose **Submit**. 9. On the **Operations** tab, confirm that the operation has succeeded. The stack	Cloud architect, DevOps engineer

Task	Description	Skills required
	deletion might take some time. 10. Repeat steps 2-9 for all member accounts.	
Destroy all resources.	To destroy resources, implement the following steps in your local development environment (Linux or macOS): 1. Navigate to the root directory of your repository. 2. Export the AWS credentials for the IAM role that you used to create the AWS CDK stack. 3. To destroy the cloud resources, run the following command: `npx cdk destroy --all`	Cloud architect, DevOps engineer

Related resources

- Amazon DataZone quickstart with AWS Glue data
- Tutorial: Create your first AWS CDK app
- Getting started with AWS CloudFormation
- Strategies for building a data mesh-based enterprise solution on AWS

Additional information

Objectives

Implementing this pattern achieves the following:

- **Decentralized ownership of data** – Shift data ownership from a central team to teams that represent the source systems, business units, or use cases of your organization.
- **Product thinking** – Introduce a product-based mindset that includes customers, the market, and other factors when considering the data assets in your organization.
- **Federated governance** – Improve security guardrails, controls, and compliance across your organization's data products.
- **Multi-account and multiple-project support** – Support efficient, secure data sharing and collaboration across the business units or projects of your organization.
- **Centralized monitoring and notifications** – Monitor the cloud resources of your data mesh by using Amazon CloudWatch, and notify users when a new member account is associated.
- **Scalability and extensibility** – Add new use cases into the data mesh as your organization evolves.

Solution scope

When you use this solution, you can start small and scale as you progress in your data mesh journey. Often, when a member account adopts the data solution, it contains account configurations specific to the organization, project, or business unit. This solution accommodates these diverse AWS account configurations by supporting the following features:

- AWS Glue Data Catalog as the data source for Amazon DataZone
- Management of the Amazon DataZone data domain and the related data portal
- Management of adding member accounts in the data mesh–based data solution
- Management of Amazon DataZone projects and environments
- Management of Amazon DataZone glossaries and metadata forms
- Management of IAM roles that correspond to the data mesh–based data solution users
- Notification of data mesh–based data solution users
- Monitoring of the provisioned cloud infrastructure

This solution uses AWS CDK and AWS CloudFormation to deploy the cloud infrastructure. It uses AWS CloudFormation to do the following:

- Define and deploy cloud resources at a lower level of abstraction.
- Deploy cloud resources from the AWS Management Console. By using this approach, you can deploy infrastructure without a development environment.

The data mesh solution uses AWS CDK to define resources at higher abstraction level. As a result, the solution provides a decoupled, modular, and scalable approach by choosing the relevant tool to deploy the cloud resources.

Next steps

You can reach out to AWS experts for guidance on building data mesh with Amazon DataZone.

The modular nature of this solution supports building data management solutions with different architectures, such as data fabric and data lakes. In addition, based on the requirements of your organization, you can extend the solution to other Amazon DataZone data sources.

Calculate value at risk (VaR) by using AWS services

Sumon Samanta, Amazon Web Services

Summary

This pattern describes how to implement a value at risk (VaR) calculation system by using AWS services. In an on-premises environment, most VaR systems use a large, dedicated infrastructure and in-house or commercial grid scheduling software to run batch processes. This pattern presents a simple, reliable, and scalable architecture to handle VaR processing in the AWS Cloud. It builds a serverless architecture that uses Amazon Kinesis Data Streams as a streaming service, Amazon Simple Queue Service (Amazon SQS) as a managed queue service, Amazon ElastiCache as a caching service, and AWS Lambda to process orders and calculate risk.

VaR is a statistical measure that traders and risk managers use to estimate potential loss in their portfolio beyond a certain confidence level. Most VaR systems involve running a large number of mathematical and statistical calculations and storing the results. These calculations require significant compute resources, so VaR batch processes have to be broken into smaller sets of compute tasks. Splitting a large batch into smaller tasks is possible because these tasks are mostly independent (that is, calculations for one task don't depend other tasks).

Another important requirement for a VaR architecture is compute scalability. This pattern uses a serverless architecture that automatically scales in or out based on compute load. Because the batch or online compute demand is difficult to predict, dynamic scaling is required to complete the process within the timeline imposed by a service-level agreement (SLA). Also, a cost-optimized architecture should be able to scale down each compute resource as soon as the tasks on that resource are complete.

AWS services are well-suited for VaR calculations because they offer scalable compute and storage capacity, analytics services for processing in a cost-optimized way, and different types of schedulers to run your risk management workflows. Also, you pay only for the compute and storage resources that you use on AWS.

Prerequisites and limitations

Prerequisites

- An active AWS account.

- Input files, which depend on your business requirements. A typical use case involves the following input files:
  - Market data file (input to the VaR calculation engine)
  - Trade data file (unless trade data comes through a stream).
  - Configuration data file (model and other static configuration data)
  - Calculation engine model files (quantitative libraries)
  - Time series data file (for historical data such as the stock price for the last five years)
- If the market data or other input comes in through a stream, Amazon Kinesis Data Streams set up, and Amazon Identity and Access Management (IAM) permissions configured to write to the stream.

This pattern builds an architecture in which trade data is written from a trading system to a Kinesis data stream. Instead of using a streaming service, you can save your trade data in small batch files, store them in an Amazon Simple Storage Service (Amazon S3) bucket, and invoke an event to start processing the data.

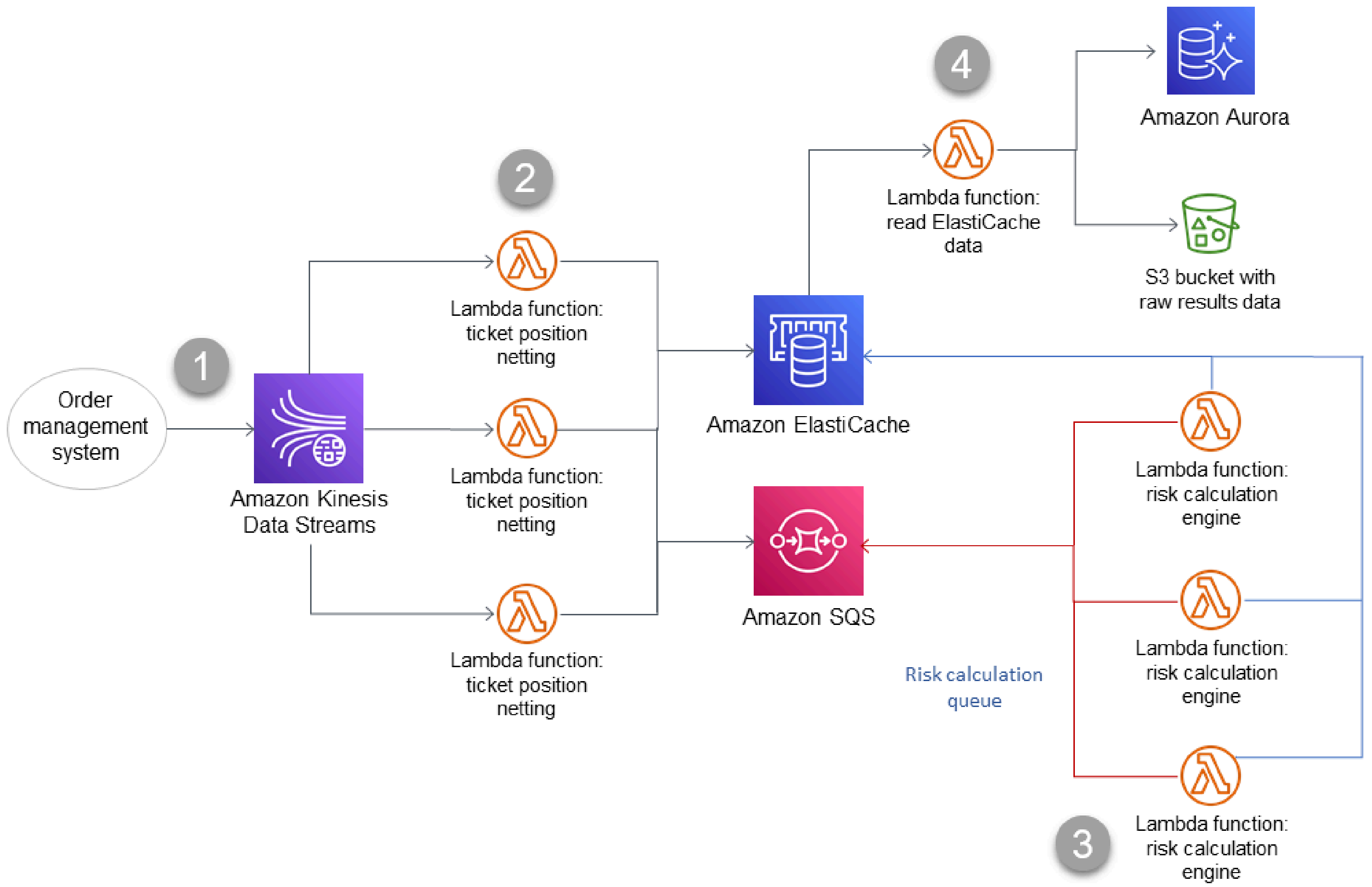
Limitations

- Kinesis data stream sequencing is guaranteed on each shard, so trade orders that are written to multiple shards are not guaranteed to be delivered in the same order as write operations.
- The AWS Lambda runtime limit is currently 15 minutes. (For more information, see the Lambda FAQ.)

Architecture

Target architecture

The following architecture diagram displays the AWS services and workflows for the risk assessment system.



The diagram illustrates the following:

1. Trades stream in from the order management system.
2. The *ticket position netting* Lambda function processes the orders and writes consolidated messages for each ticker to a risk queue in Amazon SQS.
3. The *risk calculation engine* Lambda function processes the messages from Amazon SQS, performs risk calculations, and updates the VaR profit and loss (PnL) information in the risk cache in Amazon ElastiCache.
4. The *read ElastiCache data* Lambda function retrieves the risk results from ElastiCache and stores them in a database and S3 bucket.

For more information about these services and steps, see the *Epics* section.

Automation and scale

You can deploy the entire architecture by using the AWS Cloud Development Kit (AWS CDK) or AWS CloudFormation templates. The architecture can support both batch processing and intraday (real-time) processing.

Scaling is built into the architecture. As more trades are written into the Kinesis data stream and are waiting to be processed, additional Lambda functions can be invoked to process those trades and can then scale down after processing is complete. Processing through multiple Amazon SQS risk calculation queues is also an option. If strict ordering or consolidation is required across queues, processing cannot be parallelized. However, for an end-of-the-day batch or a mini intraday batch, the Lambda functions can process in parallel and store the final results in ElastiCache.

Tools

AWS services

- Amazon Aurora MySQL-Compatible Edition is a fully managed, MySQL-compatible relational database engine that helps you set up, operate, and scale MySQL deployments. This pattern uses MySQL as an example, but you can use any RDBMS system to store data.
- Amazon ElastiCache helps you set up, manage, and scale distributed in-memory cache environments in the AWS Cloud.
- Amazon Kinesis Data Streams helps you collect and process large streams of data records in real time.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code

This pattern provides an example architecture for a VaR system in the AWS Cloud and describes how you can use Lambda functions for VaR calculations. To create your Lambda functions, see the code examples in the Lambda documentation. For assistance, contact AWS Professional Services.

Best practices

- Keep each VaR compute task as small and lightweight as possible. Experiment with different numbers of trades in each compute task to see which one is most optimized for compute time and cost.
- Store reusable objects in Amazon ElastiCache. Use a framework such as Apache Arrow to reduce serialization and deserialization.
- Consider Lambda's time limitation. If you think your compute tasks might exceed 15 minutes, try to break them down into smaller tasks to avoid the Lambda timeout. If this is not possible, you might consider a container orchestration solution with AWS Fargate, Amazon Elastic Container Service (Amazon ECS), and Amazon Elastic Kubernetes Service (Amazon EKS).

Epics

Trade flow to risk system

Task	Description	Skills required
Start writing trades.	New, settled, or partially settled trades are written from the order management system to a risk stream. This pattern uses Amazon Kinesis as the managed streaming service. The trade order ticker's hash is used to put trade orders across multiple shards.	Amazon Kinesis

Run Lambda functions for order processing

Task	Description	Skills required
Start risk processing with Lambda.	Run an AWS Lambda function for the new orders. Based	Amazon Kinesis, AWS Lambda, Amazon ElastiCache

Task	Description	Skills required
	on the number of pending trade orders, Lambda will automatically scale. Each Lambda instance has one or more orders and retrieves the latest position for each ticker from Amazon ElastiCache. (You can use a CUSIP ID, a Curve name, or an index name for other financial derivative products as a key to store and retrieve data from ElasticCache.) In ElastiCache, the total position (quantity) and the key-value pair <*ticker*, *net position*>, where *net position* is the scaling factor, are updated once for each ticker.	

Write messages for each ticker into queue

Task	Description	Skills required
Write consolidated messages to the risk queue.	Write the message to a queue. This pattern uses Amazon SQS as a managed queue service. A single Lambda instance might get a mini batch of trade orders at any given time, but will write only a single message for each ticker to Amazon SQS. A scaling factor is calculate	Amazon SQS, AWS Lambda

Task	Description	Skills required
	d: (*old net position* + *current position*) / *old net position*.	

Invoke risk engine

Task	Description	Skills required
Start risk calculations.	The Lambda function for the risk engine lambda is invoked. Each position is processed by a single Lambda function. However, for optimization purposes, each Lambda function can process multiple messages from Amazon SQS.	Amazon SQS, AWS Lambda

Retrieve risk results from cache

Task	Description	Skills required
Retrieve and update risk cache.	Lambda retrieves the current net position for each ticker from ElastiCache. It also retrieves a VaR profit and loss (PnL) array for each ticker from ElastiCache. If the PnL array already exists, the Lambda function updates the array and VaR with a scale, which comes from the Amazon SQS message written by the netting Lambda function. If the PnL array isn't	Amazon SQS, AWS Lambda, Amazon ElastiCache

Task	Description	Skills required
	in ElasticCache, a new PnL and VaR are calculated by using simulated ticker price series data.	

Update data in Elastic Cache and store in database

Task	Description	Skills required
Store risk results.	After the VaR and PnL numbers are updated in ElastiCache, a new Lambda function is invoked every five minutes. This function reads all stored data from ElastiCac he and stores it in an Aurora MySQL-Compatible database and in an S3 bucket.	AWS Lambda, Amazon ElastiCache

Related resources

- [Basel VaR Framework]

Configure cross-account access to a shared AWS Glue Data Catalog using Amazon Athena

Denis Avdonin, Amazon Web Services

Summary

This pattern provides step-by-step instructions, including AWS Identity and Access Management (IAM) policy samples, to configure cross-account sharing of a dataset stored in an Amazon Simple Storage Service (Amazon S3) bucket by using the AWS Glue Data Catalog. You can store the dataset in an S3 bucket. The metadata is collected by an AWS Glue crawler and put into the AWS Glue Data Catalog. The S3 bucket and the AWS Glue Data Catalog reside in an AWS account referred to as the *data account*. You can provide access to IAM principals in another AWS account referred to as the *consumer account*. Users can query the data in the consumer account by using the Amazon Athena serverless query engine.

Prerequisites and limitations

Prerequisites

- Two active AWS accounts
- An S3 bucket in one of the AWS accounts
- Athena engine version 2
- AWS Command Line Interface (AWS CLI), installed and configured (or AWS CloudShell for running AWS CLI commands)

Product versions

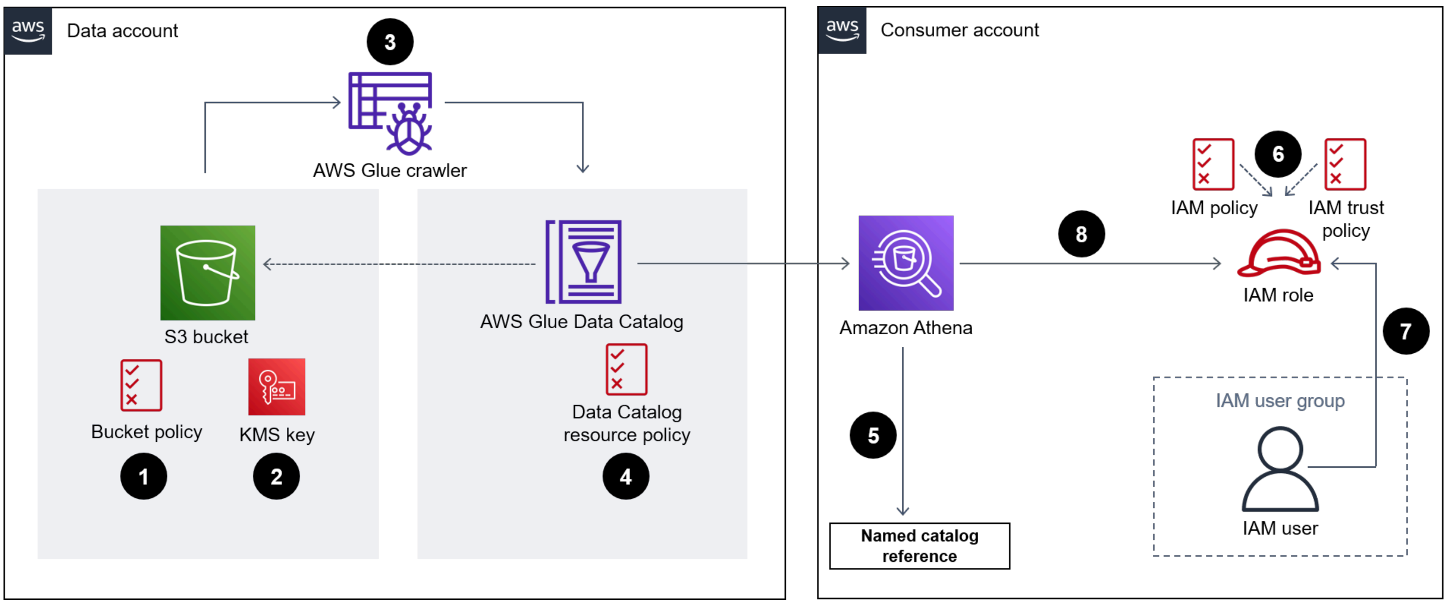
This pattern works with Athena engine version 2 and Athena engine version 3 only. We recommend that you upgrade to Athena engine version 3. If you can't upgrade from Athena engine version 1 to Athena engine version 3, follow the approach discussed in Cross-account AWS Glue Data Catalog access with Amazon Athena in the AWS Big Data Blog.

Architecture

Target technology stack

- Amazon Athena
- Amazon Simple Storage Service (Amazon S3)
- AWS Glue
- AWS Identity and Access Management (IAM)
- AWS Key Management Service (AWS KMS)

The following diagram shows an architecture that uses IAM permissions to share data in an S3 bucket in one AWS account (data account) with another AWS account (consumer account) through the AWS Glue Data Catalog.



The diagram shows the following workflow:

1. The S3 bucket policy in the data account grants permissions to an IAM role in the consumer account and to the AWS Glue crawler service role in the data account.
2. The AWS KMS key policy in the data account grants permissions to the IAM role in the consumer account and to the AWS Glue crawler service role in the data account.
3. The AWS Glue crawler in the data account discovers the schema of the data that's stored in the S3 bucket.
4. The resource policy of the AWS Glue Data Catalog in the data account grants access to the IAM role in the consumer account.
5. A user creates a named catalog reference in the consumer account by using an AWS CLI command.

6. An IAM policy grants an IAM role in the consumer account access to resources in the data account. The IAM role's trust policy allows users in the consumer account to assume the IAM role.
7. A user in the consumer account assumes the IAM role and accesses objects in the data catalog by using SQL queries.
8. The Athena serverless engine runs the SQL queries.

> **Note**
>
> IAM best practices recommend that you grant permissions to an IAM role and use identity federation.

Tools

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to protect your data.

Epics

Set up permissions in the data account

Task	Description	Skills required
Grant access to data in the S3 bucket.	Create an S3 bucket policy based on the following template and assign the	Cloud administrator

Task	Description	Skills required
	policy to the bucket where the data is stored.	

```
{
    "Version":
 "2012-10-17",
    "Statement": [
        {
            "Effect":
 "Allow",
            "Principa
l": {
                "AWS": [

 "arn:aws:iam::<con
sumer account id>:role/
<role name>",

 "arn:aws:iam::<dat
a account id>:role/
service-role/AWSGl
ueServiceRole-data-
bucket-crawler"
                ]
            },
            "Action":
 "s3:GetObject",
            "Resource":
 "arn:aws:s3:::data-
bucket/*"
        },
        {
            "Effect":
 "Allow",
            "Principa
l": {
                "AWS": [

  "arn:aws:iam::<con
sumer account id>:role/
<role name>",
```

Task	Description	Skills required

	``
	  "arn:aws:iam::<dat
	a account id>:role/
	service-role/AWSGl
	ueServiceRole-data-
	bucket-crawler"
	                ]
	            },
	            "Action":
	 "s3:ListBucket",
	            "Resource":
	 "arn:aws:s3:::data-
	bucket"
	        }
	    ]
	}
	``
	The bucket policy grants permissions to the IAM role in the consumer account and to the AWS Glue crawler service role in the data account.	

Task	Description	Skills required

(If required) Grant access to the data encryption key.	If the S3 bucket is encrypted by an AWS KMS key, grant `kms:Decrypt` permission on the key to the IAM role in the consumer account and to the AWS Glue crawler service role in the data account.
	
	Update the [key policy](https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html) with the following statement:
	
	`{`
	`    "Effect": "Allow",`
	`    "Principal": {`
	`        "AWS": [`
	`            "arn:aws:iam::<consumer account id>:role/<role name>",`
	`            "arn:aws:iam::<data account id>:role/service-role/AWSGlueServiceRole-data-bucket-crawler"`
	`        ]`
	`    },`
	`    "Action": "kms:Decrypt",`
	`    "Resource": "arn:aws:kms:<region>:<data account id>:key/<key id>"`
	`}`
		Cloud administrator

Task	Description	Skills required
Grant the crawler access to the data.	Attach the following IAM policy to the crawler's service role: ```{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::data-bucket/*" }, { "Effect": "Allow", "Action": "s3:ListBucket", "Resource": "arn:aws:s3:::data-bucket" }] }```	Cloud administrator

```
{
    "Version":
 "2012-10-17",
    "Statement": [
        {
            "Effect":
 "Allow",
            "Action":
 "s3:GetObject",
            "Resource":
 "arn:aws:s3:::data-
bucket/*"
        },
        {
            "Effect":
 "Allow",
            "Action":
 "s3:ListBucket",
            "Resource":
 "arn:aws:s3:::data-
bucket"
        }
    ]
}
```

Task	Description	Skills required
(If required) Grant the crawler access to the data encryption key.	If the S3 bucket is encrypted by an AWS KMS key, grant `kms:Decrypt` permission on the key to the crawler's service role by attaching the following policy to it:	Cloud administrator

```
{
    "Effect": "Allow",
    "Action": "kms:Decrypt",
    "Resource":
 "arn:aws:kms:<region>:<data account
 id>:key/<key id>"
}
```

Task	Description	Skills required

Grant the IAM role in the consumer account and the crawler access to the data catalog.	1. Sign in to the AWS Management Console and open the [AWS Glue console](#).
	2. In the navigation pane, under **Data Catalog**, choose **Settings**.
	3. In the **Permissions** section, add the following statement, and then choose **Save**.
	
	`{`
	`    "Version" : "2012-10-17",`
	`    "Statement" : [`
	`        {`
	`            "Effect" : "Allow",`
	`            "Principal" : {`
	`                "AWS" : [`
	`                    "arn:aws:iam::<consumer account id>:role/<role name>",`
	`                    "arn:aws:iam::<data account id>:role/service-role/AWSGlueServiceRole-data-bucket-crawler"`
	`                ]`
	`            },`
	`            "Action" : "glue:*",`
		Cloud administrator

Task	Description	Skills required

	`"Resource`
	`" : [`
	
	`"arn:aws:glue:<reg`
	`ion>:<data account`
	`id>:catalog",`
	
	`"arn:aws:glue:<reg`
	`ion>:<data account`
	`id>:database/*",`
	
	`"arn:aws:glue:<reg`
	`ion>:<data account`
	`id>:table/*"`
	`]`
	`}`
	`]`
	`}`
	
	This policy allows all AWS Glue actions on all databases and tables in the data account. You can customize the policy to grant only required permissions to the consumer principals. For example, you can provide read-only access to specific tables or views in a database.	

Access data from the consumer account

Task	Description	Skills required
Create a named reference for the data catalog.	To create a named data catalog reference, use CloudShell or a locally	Cloud administrator

Task	Description	Skills required
	installed AWS CLI to run the following command: `aws athena create-data-catalog --name <shared catalog name> --type GLUE --parameters catalog-id=<data account id>`	

Task	Description	Skills required
Grant the IAM role in the consumer account access to the data.	Attach the following policy to the IAM role in the consumer account to grant the role cross-account access to the data: `{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::data-bucket/*" }, { "Effect": "Allow", "Action": "s3:ListBucket", "Resource": "arn:aws:s3:::data-bucket" }, { "Effect": "Allow", "Action": "glue:*", "Resource": ["arn:aws:glue:<region>:<data account id>:catalog",`	Cloud administrator

Task	Description	Skills required

	``` "arn:aws:glue:<reg ion>:<data account id>:database/*", "arn:aws:glue:<reg ion>:<data account id>:table/*"] }] } ```
	Next, use the following template to specify what users can accept the IAM role in its trust policy:
	``` { "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "AWS": "arn:aws:iam::<consumer account id>:user/<IAM user>" }, "Action": "sts:AssumeRole" }] } ```
	Finally, grant user permissions to assume the IAM role by	

Task	Description	Skills required
	attaching the same policy to the user group they belong to.	

(If required) Grant the IAM role in the consumer account access to the data encryption key.	If the S3 bucket is encrypted by an AWS KMS key, grant `kms:Decrypt` permission on the key to the IAM role in the consumer account by attaching the following policy to it:
	`{`
	`    "Effect": "Allow",`
	`    "Action": "kms:Decrypt",`
	`    "Resource": "arn:aws:kms:<region>:<data account id>:key/<key id>"`
	`}`
		Cloud administrator

Switch to the IAM role in the consumer account to access data.	As a data consumer, [switch to the IAM role](#) to access data in the data account.	Data consumer

Task	Description	Skills required

Access the data.	Query data using Athena. For example, open the Athena query editor and run the following query:
	`SELECT * FROM <shared catalog name>.<database name>.<table name>`
	Instead of using a named catalog reference, you can also refer to the catalog by its Amazon Resource Name (ARN).
	Note
	If you use a dynamic catalog reference in a query or view, surround the reference with escaped double quotation marks (\"). For example:
	`SELECT * FROM \"glue:arn:aws:glue:<region>:<data account id>:catalog\".<database name>.<table name>`
		Data consumer

Task	Description	Skills required
	For more information, see Cross-account access to AWS Glue data catalogs in the Amazon Athena User Guide.	

Related resources

- Cross-account access to AWS Glue data catalogs (Athena documentation)
- (AWS CLI) create-data-catalog (AWS CLI Command Reference)
- Cross-account AWS Glue Data Catalog access with Amazon Athena (AWS Big Data Blog)
- Security best practices in IAM (IAM documentation)

Additional information

Using Lake Formation as an alternative for cross-account sharing

You can also use AWS Lake Formation to share access to AWS Glue catalog objects across accounts. Lake Formation provides fine-grained access control at the column and row level, tag-based access control, governed tables for ACID transactions, and other functionality. Although Lake Formation is well-integrated with Athena, it does require additional configuration compared to this pattern's IAM-only approach. We recommend that you consider the decision to use Lake Formation or IAM-only access controls within the wider context of your overall solution architecture. Considerations include what other services are involved and how they integrate with both approaches.

Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL

Po Hong, Amazon Web Services

Summary

NORMALIZE is a Teradata extension to the ANSI SQL standard. When a SQL table includes a column that has a **PERIOD** data type, **NORMALIZE** combines values that meet or overlap in that column, to form a single period that consolidates multiple, individual period values. To use **NORMALIZE**, at least one column in the SQL **SELECT** list must be of Teradata's temporal **PERIOD** data type. For more information about **NORMALIZE**, see the Teradata documentation.

Amazon Redshift doesn't support **NORMALIZE**, but you can implement this functionality by using native SQL syntax and the **LAG** window function in Amazon Redshift. This pattern focuses on using the Teradata **NORMALIZE** extension with the **ON MEETS OR OVERLAPS** condition, which is the most popular format. It explains how this feature works in Teradata and how it can be converted into Amazon Redshift native SQL syntax.

Prerequisites and limitations

Prerequisites

- Basic Teradata SQL knowledge and experience
- Amazon Redshift knowledge and experience

Architecture

Source technology stack

- Teradata data warehouse

Target technology stack

- Amazon Redshift

Target architecture

For a high-level architecture for migrating a Teradata database to Amazon Redshift, see the pattern Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents. The migration doesn't automatically convert the Teradata **NORMALIZE** phrase to Amazon Redshift SQL. You can convert this Teradata extension by following the guidelines in this pattern.

Tools

Code

To illustrate the concept and functionality of **NORMALIZE**, consider the following table definition in Teradata:

```
CREATE TABLE systest.project
     (    emp_id        INTEGER,
          project_name  VARCHAR(20),
          dept_id       INTEGER,
          duration      PERIOD(DATE)
     );
```

Run the following SQL code to insert sample data into the table:

```
BEGIN TRANSACTION;

INSERT INTO systest.project VALUES (10, 'First Phase', 1000,  PERIOD(DATE '2010-01-10',
 DATE '2010-03-20') );
INSERT INTO systest.project VALUES (10, 'First Phase', 2000,  PERIOD(DATE '2010-03-20',
 DATE '2010-07-15') );

INSERT INTO systest.project VALUES (10, 'Second Phase', 2000,  PERIOD(DATE
 '2010-06-15', DATE '2010-08-18') );
INSERT INTO systest.project VALUES (20, 'First Phase', 2000,  PERIOD(DATE '2010-03-10',
 DATE '2010-07-20') );

INSERT INTO systest.project VALUES (20, 'Second Phase', 1000,  PERIOD(DATE
 '2020-05-10', DATE '2020-09-20') );

END TRANSACTION;
```

Results:

```
select * from systest.project order by 1,2,3;
```

```
 *** Query completed. 4 rows found. 4 columns returned.
 *** Total elapsed time was 1 second.

     emp_id  project_name              dept_id  duration
-----------  --------------------  -----------  ------------------------
         10  First Phase                  1000  ('10/01/10', '10/03/20')
         10  First Phase                  2000  ('10/03/20', '10/07/15')
         10  Second Phase                 2000  ('10/06/15', '10/08/18')
         20  First Phase                  2000  ('10/03/10', '10/07/20')
         20  Second Phase                 1000  ('20/05/10', '20/09/20')
```

Teradata NORMALIZE use case

Now add the Teradata **NORMALIZE** SQL clause to the **SELECT** statement:

```
SELECT NORMALIZE ON MEETS OR OVERLAPS emp_id, duration
FROM systest.project
ORDER BY 1,2;
```

This **NORMALIZE** operation is performed on a single column (emp_id). For emp_id=10, the three overlapping period values in duration coalesce into a single period value, as follows:

```
     emp_id  duration
-----------  ------------------------
         10  ('10/01/10', '10/08/18')
         20  ('10/03/10', '10/07/20')
         20  ('20/05/10', '20/09/20')
```

The following **SELECT** statement performs a **NORMALIZE** operation on project_name and dept_id. Note that the **SELECT** list contains only one **PERIOD** column, duration.

```
SELECT NORMALIZE project_name, dept_id, duration
FROM systest.project;
```

Output:

```
project_name              dept_id  duration
--------------------  -----------  ------------------------
First Phase                  1000  ('10/01/10', '10/03/20')
Second Phase                 1000  ('20/05/10', '20/09/20')
First Phase                  2000  ('10/03/10', '10/07/20')
```

```
Second Phase                2000  ('10/06/15', '10/08/18')
```

Amazon Redshift equivalent SQL

Amazon Redshift currently doesn't support the **PERIOD** data type in a table. Instead, you need to divide a Teradata **PERIOD** data field into two parts: start_date, end_date, as follows:

```
CREATE TABLE systest.project
     (    emp_id        INTEGER,
          project_name  VARCHAR(20),
          dept_id       INTEGER,
          start_date  DATE,
          end_date    DATE
     );
```

Insert sample data into the table:

```
BEGIN TRANSACTION;

INSERT INTO systest.project VALUES (10, 'First Phase', 1000,  DATE '2010-01-10', DATE
 '2010-03-20' );
INSERT INTO systest.project VALUES (10, 'First Phase', 2000,  DATE '2010-03-20', DATE
 '2010-07-15');

INSERT INTO systest.project VALUES (10, 'Second Phase', 2000,  DATE '2010-06-15', DATE
 '2010-08-18' );
INSERT INTO systest.project VALUES (20, 'First Phase', 2000,  DATE '2010-03-10', DATE
 '2010-07-20' );

INSERT INTO systest.project VALUES (20, 'Second Phase', 1000,  DATE '2020-05-10', DATE
 '2020-09-20' );

END TRANSACTION;
```

Output:

```
 emp_id | project_name | dept_id | start_date |  end_date
--------+--------------+---------+------------+------------
     10 | First Phase  |    1000 | 2010-01-10 | 2010-03-20
     10 | First Phase  |    2000 | 2010-03-20 | 2010-07-15
     10 | Second Phase |    2000 | 2010-06-15 | 2010-08-18
     20 | First Phase  |    2000 | 2010-03-10 | 2010-07-20
```

```
     20 | Second Phase |    1000 | 2020-05-10 | 2020-09-20
(5 rows)
```

To rewrite Teradata's **NORMALIZE** clause, you can use the [LAG window function] in Amazon Redshift. This function returns the values for a row at a given offset above (before) the current row in the partition.

You can use the **LAG** function to identify each row that begins a new period by determining if a period meets or overlaps with the previous period (0 if yes and 1 if no). When this flag is cumulatively summed up, it provides a group identifier that can be used in the outer **Group By** clause to arrive at the desired result in Amazon Redshift.

Here's a sample Amazon Redshift SQL statement that uses **LAG()**:

```
SELECT emp_id, start_date, end_date,
            (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY emp_id ORDER BY
 start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
FROM systest.project
ORDER BY 1,2;
```

Output:

```
 emp_id | start_date |  end_date  | groupstartflag
--------+------------+------------+----------------
     10 | 2010-01-10 | 2010-03-20 |              1
     10 | 2010-03-20 | 2010-07-15 |              0
     10 | 2010-06-15 | 2010-08-18 |              0
     20 | 2010-03-10 | 2010-07-20 |              1
     20 | 2020-05-10 | 2020-09-20 |              1
(5 rows)
```

The following Amazon Redshift SQL statement normalizes only on the emp_id column:

```
SELECT T2.emp_id, MIN(T2.start_date) as new_start_date, MAX(T2.end_date) as
 new_end_date
FROM
( SELECT T1.*, SUM(GroupStartFlag) OVER (PARTITION BY emp_id ORDER BY start_date ROWS
 UNBOUNDED PRECEDING) As GroupID
FROM ( SELECT emp_id, start_date, end_date,
            (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY emp_id ORDER BY
 start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
```

```
FROM systest.project ) T1
) T2
GROUP BY T2.emp_id, T2.GroupID
ORDER BY 1,2;
```

Output:

```
 emp_id | new_start_date | new_end_date
--------+----------------+------------------------------------
     10 | 2010-01-10     | 2010-08-18
     20 | 2010-03-10     | 2010-07-20
     20 | 2020-05-10     | 2020-09-20
(3 rows)
```

The following Amazon Redshift SQL statement normalizes on both the project_name and dept_id columns:

```
SELECT T2.project_name, T2.dept_id, MIN(T2.start_date) as new_start_date,
 MAX(T2.end_date) as new_end_date
FROM
( SELECT T1.*, SUM(GroupStartFlag) OVER (PARTITION BY project_name, dept_id ORDER BY
 start_date ROWS UNBOUNDED PRECEDING) As GroupID
FROM ( SELECT project_name, dept_id, start_date, end_date,
            (CASE WHEN start_date <= LAG(end_date) OVER (PARTITION BY project_name,
 dept_id ORDER BY start_date, end_date) THEN 0 ELSE 1 END) AS GroupStartFlag
FROM systest.project ) T1
) T2
GROUP BY T2.project_name, T2.dept_id, T2.GroupID
ORDER BY 1,2,3;
```

Output:

```
 project_name | dept_id | new_start_date | new_end_date
--------------+---------+----------------+--------------
 First Phase  |    1000 | 2010-01-10     | 2010-03-20
 First Phase  |    2000 | 2010-03-10     | 2010-07-20
 Second Phase |    1000 | 2020-05-10     | 2020-09-20
 Second Phase |    2000 | 2010-06-15     | 2010-08-18
(4 rows)
```

Epics

Convert NORMALIZE to Amazon Redshift SQL

Task	Description	Skills required
Create your Teradata SQL code.	Use the NORMALIZE phrase according to your needs.	SQL developer
Convert the code to Amazon Redshift SQL.	To convert your code, follow the guidelines in the "Tools" section of this pattern.	SQL developer
Run the code in Amazon Redshift.	Create your table, load data into the table, and run your code in Amazon Redshift.	SQL developer

Related resources

References

- Teradata NORMALIZE temporal feature (Teradata documentation)
- LAG window function (Amazon Redshift documentation)
- Migrate to Amazon Redshift (AWS website)
- Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents (AWS Prescriptive Guidance)
- Convert the Teradata RESET WHEN feature to Amazon Redshift SQL (AWS Prescriptive Guidance)

Tools

- AWS Schema Conversion Tool (AWS SCT)

Partners

- AWS Migration Competency Partners

Convert the Teradata RESET WHEN feature to Amazon Redshift SQL

Po Hong, Amazon Web Services

Summary

RESET WHEN is a Teradata feature used in SQL analytical window functions. It is an extension to the ANSI SQL standard. **RESET WHEN** determines the partition over which an SQL window function operates based on some specified condition. If the condition evaluates to **TRUE**, a new, dynamic sub-partition is created inside the existing window partition. For more information about **RESET WHEN**, see the Teradata documentation.

Amazon Redshift doesn't support **RESET WHEN** in SQL window functions. To implement this functionality, you have to convert **RESET WHEN** to the native SQL syntax in Amazon Redshift, and use multiple, nested functions. This pattern demonstrates how you can use the Teradata **RESET WHEN** feature and how you can convert it to Amazon Redshift SQL syntax.

Prerequisites and limitations

Prerequisites

- Basic knowledge of the Teradata data warehouse and its SQL syntax
- Good understanding of Amazon Redshift and its SQL syntax

Architecture

Source technology stack

- Teradata data warehouse

Target technology stack

- Amazon Redshift

Architecture

For a high-level architecture for migrating a Teradata database to Amazon Redshift, see the pattern Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents. The migration doesn't automatically convert the Teradata **RESET WHEN** phrase to Amazon Redshift SQL. You can convert this Teradata extension by following the guidelines in the next section.

Tools

Code

To illustrate the concept of **RESET WHEN**, consider the following table definition in Teradata:

```
create table systest.f_account_balance
( account_id integer NOT NULL,
  month_id integer,
  balance integer )
unique primary index (account_id, month_id);
```

Run the following SQL code to insert sample data into the table:

```
BEGIN TRANSACTION;
Insert Into systest.f_account_balance values (1,1,60);
Insert Into systest.f_account_balance values (1,2,99);
Insert Into systest.f_account_balance values (1,3,94);
Insert Into systest.f_account_balance values (1,4,90);
Insert Into systest.f_account_balance values (1,5,80);
Insert Into systest.f_account_balance values (1,6,88);
Insert Into systest.f_account_balance values (1,7,90);
Insert Into systest.f_account_balance values (1,8,92);
Insert Into systest.f_account_balance values (1,9,10);
Insert Into systest.f_account_balance values (1,10,60);
Insert Into systest.f_account_balance values (1,11,80);
Insert Into systest.f_account_balance values (1,12,10);
END TRANSACTION;
```

The sample table has the following data:

account_id	month_id	balance
1	1	60

1	2	99
1	3	94
1	4	90
1	5	80
1	6	88
1	7	90
1	8	92
1	9	10
1	10	60
1	11	80
1	12	10

For each account, let's say that you want to analyze the sequence of consecutive monthly balance increases. When one month's balance is less than, or equal to, the previous month's balance, the requirement is to reset the counter to zero and restart.

Teradata RESET WHEN use case

To analyze this data, Teradata SQL uses a window function with a nested aggregate and a **RESET WHEN** phrase, as follows:

```
SELECT account_id, month_id, balance,
 ( ROW_NUMBER() OVER (PARTITION BY account_id ORDER BY month_id
RESET WHEN balance <= SUM(balance) over (PARTITION BY account_id ORDER BY month_id ROWS
 BETWEEN 1 PRECEDING AND 1 PRECEDING) ) -1 ) as balance_increase
FROM systest.f_account_balance
ORDER BY 1,2;
```

Output:

account_id	month_id	balance	balance_increase
1	1	60	0
1	2	99	1
1	3	94	0
1	4	90	0
1	5	80	0
1	6	88	1
1	7	90	2
1	8	92	3
1	9	10	0
1	10	60	1
1	11	80	2
1	12	10	0

The query is processed as follows in Teradata:

1. The **SUM(balance)** aggregate function calculates the sum of all balances for a given account in a given month.
2. We check to see if a balance in a given month (for a given account) is greater than the balance of the previous month.
3. If the balance increased, we track a cumulative count value. If the **RESET WHEN** condition evaluates to **false**, which means that the balance has increased over successive months, we continue to increase the count.
4. The **ROW_NUMBER()** ordered analytical function calculates the count value. When we reach a month whose balance is less than, or equal to, the balance of the previous month, the **RESET WHEN** condition evaluates to **true**. If so, we start a new partition and **ROW_NUMBER()** restarts

the count from 1. We use **ROWS BETWEEN 1 PRECEDING AND 1 PRECEDING** to access the value of the previous row.

5. We subtract 1 to ensure that the count value starts with 0.

Amazon Redshift equivalent SQL

Amazon Redshift doesn't support the **RESET WHEN** phrase in an SQL analytical window function. To produce the same result, you must rewrite the Teradata SQL using Amazon Redshift native SQL syntax and nested sub-queries, as follows:

```
SELECT account_id, month_id, balance,
   (ROW_NUMBER() OVER(PARTITION BY account_id, new_dynamic_part ORDER BY month_id) -1)
 as balance_increase
FROM
( SELECT account_id, month_id, balance, prev_balance,
SUM(dynamic_part) OVER (PARTITION BY account_id ORDER BY month_id ROWS BETWEEN
 UNBOUNDED PRECEDING AND CURRENT ROW) As new_dynamic_part
FROM ( SELECT account_id, month_id, balance,
SUM(balance) over (PARTITION BY account_id ORDER BY month_id ROWS BETWEEN 1 PRECEDING
 AND 1 PRECEDING) as prev_balance,
(CASE When balance <= prev_balance Then 1 Else 0 END) as dynamic_part
FROM systest.f_account_balance ) A
) B
ORDER BY 1,2;
```

Because Amazon Redshift doesn't support nested window functions in the **SELECT** clause of a single SQL statement, you must use two nested sub-queries.

- In the inner sub-query (alias A), a dynamic partition indicator (**dynamic_part**) is created and populated. **dynamic_part** is set to 1 if one month's balance is less than or equal to the preceding month's balance; otherwise, it's set to 0.
- In the next layer (alias B), a **new_dynamic_part** attribute is generated as the result of a **SUM** window function.
- Finally, you add **new_dynamic_part** as a new partition attribute (**dynamic partition**) to the existing partition attribute (**account_id**) and apply the same **ROW_NUMBER()** window function as in Teradata (and minus one).

After these changes, Amazon Redshift SQL generates the same output as Teradata.

Epics

Convert RESET WHEN to Amazon Redshift SQL

Task	Description	Skills required
Create your Teradata window function.	Use nested aggregates and the RESET WHEN phrase according to your needs.	SQL developer
Convert the code to Amazon Redshift SQL.	To convert your code, follow the guidelines in the "Tools" section of this pattern.	SQL developer
Run the code in Amazon Redshift.	Create your table, load data into the table, and run your code in Amazon Redshift.	SQL developer

Related resources

References

- RESET WHEN Phrase (Teradata documentation)
- RESET WHEN explanation (Stack Overflow)
- Migrate to Amazon Redshift (AWS website)
- Migrate a Teradata database to Amazon Redshift using AWS SCT data extraction agents (AWS Prescriptive Guidance)
- Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL (AWS Prescriptive Guidance)

Tools

- AWS Schema Conversion Tool (AWS SCT)

Partners

- AWS Migration Competency Partners

Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code

Kirankumar Chandrashekar and Abdel Jaidi, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern describes how to use serverless computing and infrastructure as code (IaC) to implement and administer a data lake on the Amazon Web Services (AWS) Cloud. This pattern is based on the serverless data lake framework (SDLF) workshop developed by AWS.

SDLF is a collection of reusable resources that accelerate the delivery of enterprise data lakes on the AWS Cloud and helps with faster deployment to production. It is used to implement the foundational structure of a data lake by following best practices.

SDLF implements a continuous integration / continuous deployment (CI/CD) process throughout the code and infrastructure deployment by using AWS services such as AWS CodePipeline, AWS CodeBuild, and AWS CodeCommit.

This pattern uses multiple AWS serverless services to simplify data lake management. These include Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB for storage, AWS Lambda and AWS Glue for computing, and Amazon CloudWatch Events, Amazon Simple Queue Service (Amazon SQS), and AWS Step Functions for orchestration.

AWS CloudFormation and AWS code services act as the IaC layer to provide reproducible and fast deployments with easy operations and administration.

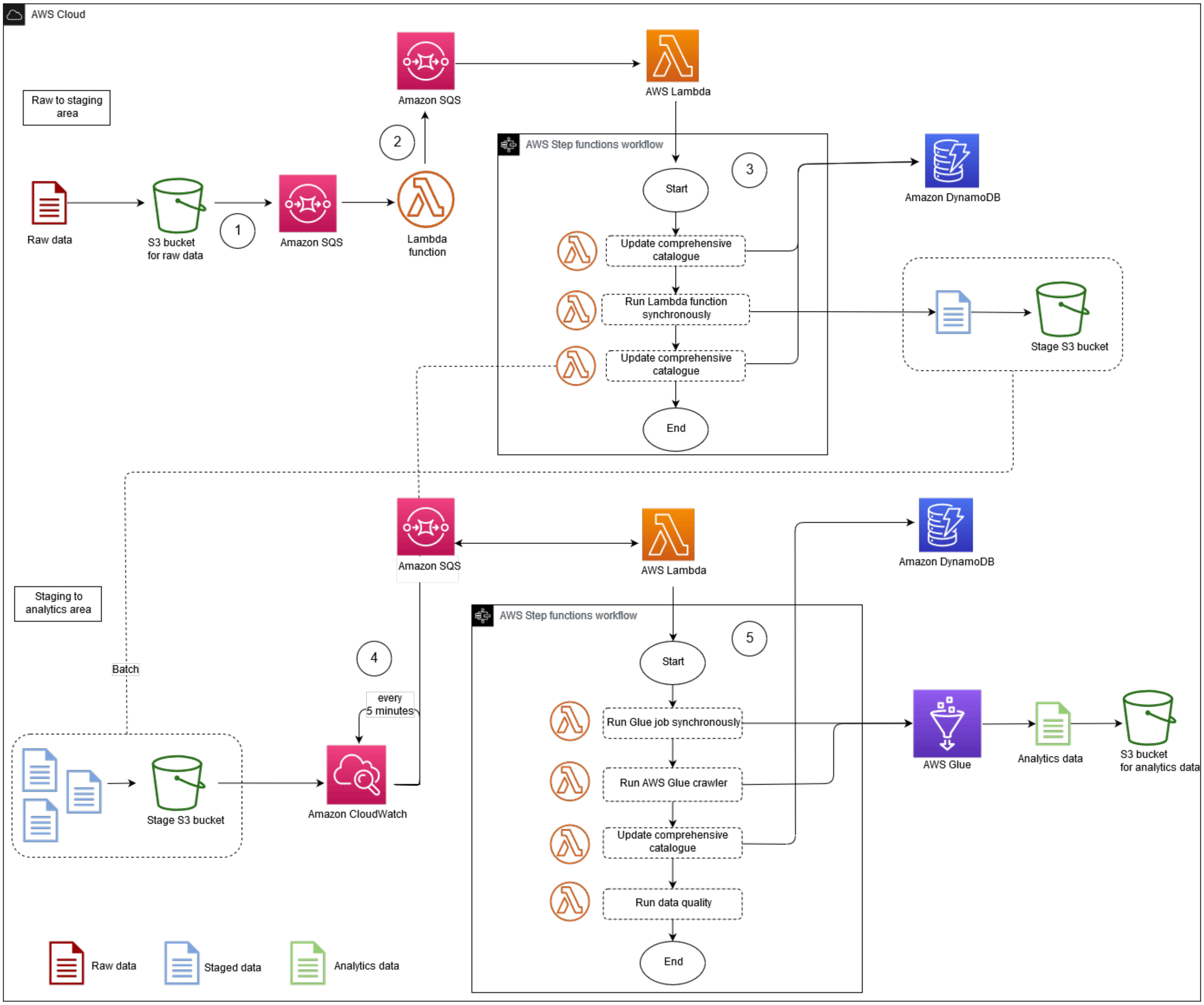
Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured.
- A Git client, installed and configured.

- The SDLF workshop, open in a web browser window and ready to use.

Architecture



The architecture diagram illustrates an event-driven process with the following steps.

1. After a file is added to the raw data S3 bucket, an Amazon S3 event notification is placed in an SQS queue. Each notification is delivered as a JSON file, which contains metadata such as the S3 bucket name, object key, or timestamp.
2. This notification is consumed by a Lambda function that routes the event to the correct extraction, transformation, and loading (ETL) process based on the metadata. The Lambda

function can also use contextual configurations stored in an Amazon DynamoDB table. This step enables decoupling and scaling to multiple applications in the data lake.

3. The event is routed to the first Lambda function in the ETL process, which transforms and moves data from the raw data area to the staging area for the data lake. The first step is to update the comprehensive catalog. This is a DynamoDB table that contains all the file metadata of the data lake. Each row in this table holds operational metadata about a single object stored in Amazon S3. A synchronous call is made to a Lambda function that performs a light transformation, which is a computationally inexpensive operation (such as converting a file from one format to another), on the S3 object. Because a new object has been added to the staging S3 bucket, the comprehensive catalog is updated and a message is sent to the SQS queue waiting for the next phase in the ETL.
4. A CloudWatch Events rule triggers a Lambda function every 5 minutes. This function checks if messages were delivered to the SQS queue from the previous ETL phase. If a message was delivered, the Lambda function begins the second function from AWS Step Functions in the ETL process.
5. A heavy transformation is then applied on a batch of files. This heavy transformation is a computationally expensive operation, such as a synchronous call to an AWS Glue job, AWS Fargate task, Amazon EMR step, or Amazon SageMaker notebook. Table metadata is extracted from the output files by using an AWS Glue crawler, which updates the AWS Glue catalog. File metadata is also added to the comprehensive catalog table in DynamoDB. Finally, a data quality step leveraging Deequ is also run.

Technology stack

- Amazon CloudWatch Events
- AWS CloudFormation
- AWS CodePipeline
- AWS CodeBuild
- AWS CodeCommit
- Amazon DynamoDB
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon SQS

- AWS Step Functions

Tools

- Amazon CloudWatch Events – CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS CloudFormation – CloudFormation helps create and provision AWS infrastructure deployments predictably and repeatedly.
- AWS CodeBuild – CodeBuild is a fully managed build service that compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- AWS CodeCommit – CodeCommit is a version control service hosted by AWS that you can use to privately store and manage assets (such as source code and binary files).
- AWS CodePipeline – CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your software changes continuously.
- Amazon DynamoDB – DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with scalability.
- AWS Glue – AWS Glue is a fully managed ETL service that makes it easier to prepare and load data for analytics.
- AWS Lambda – Lambda supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service. Amazon S3 can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- AWS Step Functions - AWS Step Functions is a serverless function orchestrator that makes it easy to sequence AWS Lambda functions and multiple AWS services into business-critical applications.
- Amazon SQS – Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service that helps you decouple and scale microservices, distributed systems, and serverless applications.
- Deequ – Deequ is a tool that helps you compute data quality metrics for large datasets, define and verify data quality constraints, and stay informed about changes in the data distribution.

Code repository

The source code and resources for the SDLF are available in the AWS Labs GitHub repository.

Epics

Set up the CI/CD pipeline to provision IaC

Task	Description	Skills required
Set up the CI/CD pipeline to manage IaC for the data lake.	Sign in to the AWS Management Console and follow the steps from the Initial setup section of the SDLF workshop. This creates the initial CI/CD resources , such as CodeCommit repositories, CodeBuild environments, and CodePipeline pipelines that provision and manage IaC for the data lake.	DevOps engineer

Version-control the IaC

Task	Description	Skills required
Clone the CodeCommit repository on your local machine.	Follow the steps from the Deploying the foundations section of the SDLF workshop. This helps you clone the Git repository that hosts IaC into your local environment. For more information, see Connecting to CodeCommit repositories from the CodeCommit documentation.	DevOps engineer

Task	Description	Skills required
Modify the CloudFormation templates.	Use your local workstation and a code editor to modify the CloudFormation templates according to your use cases or requirements. Commit them to the locally cloned Git repository. For more information, see [Working with AWS CloudFormation templates]() from the AWS CloudFormation documentation.	DevOps engineer

Task	Description	Skills required
Push the changes to the CodeCommit repository.	Your infrastructure code is now under version control and modifications to your code base are tracked. When you push a change to the CodeCommit repository, CodePipeline automatically applies it to your infrastru cture and delivers it to CodeBuild. **Important** If you use the AWS SAM CLI in CodeBuild, run the `sam package` and `sam deploy` commands. If you use AWS CLI, run the `aws cloudformation package` and `aws cloudformation deploy` commands.	DevOps engineer

Related resources

Set up the CI/CD pipeline to provision IaC

- SDLF workshop – Initial setup

Version-control the IaC

- SDLF workshop – Deploying the foundations
- Connecting to CodeCommit repositories

- Working with AWS CloudFormation templates

Other resources

- AWS serverless data analytics pipeline reference architecture
- SDLF documentation

Enforce tagging of Amazon EMR clusters at launch

Priyanka Chaudhary, Amazon Web Services

Summary

This pattern provides a security control that ensures that Amazon EMR clusters are tagged when they are created.

Amazon EMR is an Amazon Web Services (AWS) service for processing and analyzing vast amounts of data. Amazon EMR offers an expandable, low-configuration service as an easier alternative to running in-house cluster computing. You can use tagging to categorize AWS resources in different ways, such as by purpose, owner, or environment . For example, you can tag your Amazon EMR clusters by assigning custom metadata to each cluster. A tag consists of a key and value that you define. We recommend that you create a consistent set of tags to meet your organization's requirements. When you add a tag to an Amazon EMR cluster, the tag is also propagated to each active Amazon Elastic Compute Cloud (Amazon EC2) instance that is associated with the cluster. Similarly, when you remove a tag from an Amazon EMR cluster, that tag is removed from each associated, active EC2 instance as well.

The detective control monitors API calls and initiates an Amazon CloudWatch Events event for the RunJobFlow, AddTags, RemoveTags, and CreateTags APIs. The event calls AWS Lambda, which runs a Python script. The Python function gets the Amazon EMR cluster ID from the JSON input from the event and performs the following checks:

- Check if the Amazon EMR cluster is configured with tag names that you specify.
- If not, send an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information: the Amazon EMR cluster name, violation details, AWS Region, AWS account, and Amazon Resource Name (ARN) for Lambda that this notification is sourced from.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Simple Storage Service (Amazon S3) bucket to upload the provided Lambda code. Or, you can create an S3 bucket for this purpose, as described in the *Epics* section.

- An active email address where you would like to receive violation notifications.
- A list of mandatory tags you want to check for.

Limitations

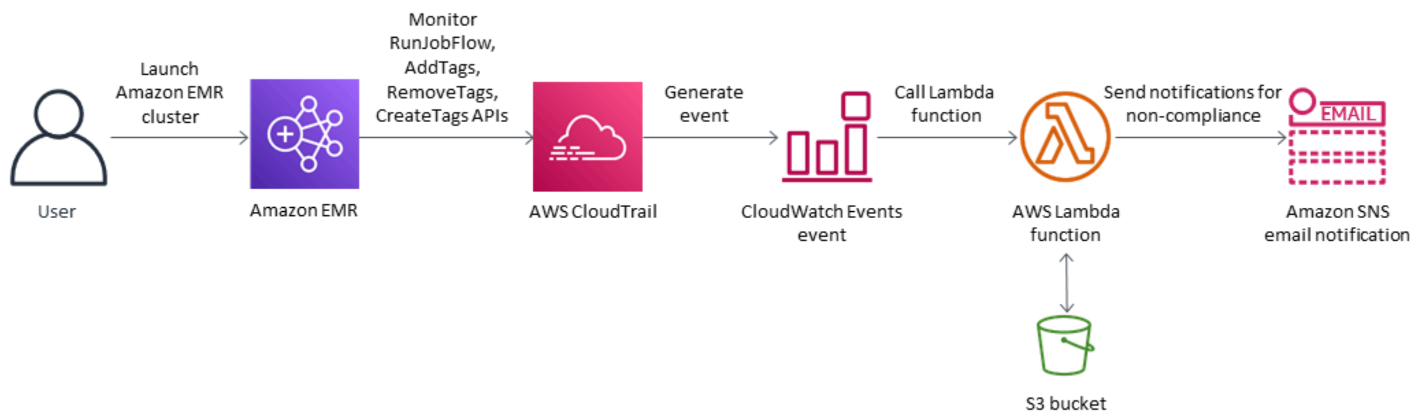
- This security control is regional. You must deploy it in each AWS Region that you want to monitor.

Product versions

- Amazon EMR release 4.8.0 and later.

Architecture

Workflow architecture



Automation and scale

- If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- Amazon CloudWatch Events - Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- Amazon EMR - Amazon EMR is web service that simplifies running big data frameworks and processing vast amounts of data efficiently.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes the following attachments:

- `EMRTagValidation.zip` – The Lambda code for the security control.
- `EMRTagValidation.yml` – The CloudFormation template that sets up the event and Lambda function.

Epics

Set up the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	On the Amazon S3 console, choose or create an S3 bucket to host the Lambda code .zip file. This S3 bucket must be	Cloud architect

Task	Description	Skills required
	in the same AWS Region as the Amazon EMR cluster you want to monitor. An Amazon S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.	
Upload the Lambda code.	Upload the Lambda code .zip file provided in the *Attachments* section to the S3 bucket.	Cloud architect

Deploy the AWS CloudFormation template

Task	Description	Skills required
Launch the AWS CloudFormation template.	Open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/) in the same AWS Region as your S3 bucket and deploy the template. For more information about deploying AWS CloudFormation templates, see [Creating a stack on the AWS CloudFormation console](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html) in the CloudFormation documentation.	Cloud architect
Complete the parameters in the template.	When you launch the template, you'll be prompted for the following information:	Cloud architect

Task	Description	Skills required
	• **S3 bucket**: Specify the bucket that you created or selected in the first epic. This is where you uploaded the attached Lambda code (.zip file). • **S3 key**: Specify the location of the Lambda .zip file in your S3 bucket (for example, *filename*.zip or controls/*filename*.zip). Do not include leading slashes. • **Notification email**: Provide an active email address where you want to receive Amazon SNS notifications. • **Tagging key names**: Provide the tags you want to check for, in a comma-separated list (for example, `ApplicationID` , `Environment` , `Owner`). The CloudWatch Events event monitors the cluster for these tags and sends a notification if they aren't found. • **Lambda logging level**: Specify the logging level and frequency for the Lambda function. Use **Info** to log detailed informational messages on progress, **Error** for error events	

Task	Description	Skills required
	that would still allow the deployment to continue, and **Warning** for potentially harmful situations.	

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the CloudFormation template deploys successfully, it sends a subscription email to the email address you provided. You must confirm this email subscription to start receiving violation notifications.	Cloud architect

Related resources

- AWS Lambda developer guide
- Tagging clusters in Amazon EMR

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Ensure Amazon EMR logging to Amazon S3 is enabled at launch

Priyanka Chaudhary, Amazon Web Services

Summary

This pattern provides a security control that monitors logging configuration for Amazon EMR clusters running on Amazon Web Services (AWS).

Amazon EMR is an AWS tool for big data processing and analysis. Amazon EMR offers the expandable low-configuration service as an alternative to running in-house cluster computing. Amazon EMR provides two types of EMR clusters.

- **Transient Amazon EMR clusters**: Transient Amazon EMR clusters automatically shut down and stop incurring costs when processing is finished.
- **Persistent Amazon EMR clusters**: Persistent Amazon EMR clusters continue to run after the data processing job is complete.

Amazon EMR and Hadoop both produce log files that report status on the cluster. By default, these are written to the master node in the **/mnt/var/log/** directory. Depending on how you configure the cluster when you launch it, you can also save these logs to Amazon Simple Storage Service (Amazon S3) and view them through the graphical debugging tool. Note that Amazon S3 logging can be specified only when the cluster is launched. With this configuration, logs are sent from the primary node to the Amazon S3 location every 5 minutes. For transient clusters, Amazon S3 logging is important because the clusters disappear when processing is complete, and these log files can be use to debug any failed jobs.

The pattern uses an AWS CloudFormation template to deploy a security control that monitors for API calls and starts Amazon CloudWatch Events on "RunJobFlow." The trigger invokes AWS Lambda, which runs a Python script. The Lambda function retrieves the EMR cluster ID from the event JSON input and also checks for an Amazon S3 log URI. If an Amazon S3 URI is not found, the Lambda function sends an Amazon Simple Notification Service (Amazon SNS) notification detailing the EMR cluster name, violation details, AWS Region, AWS account, and the Lambda Amazon Resource Name (ARN) that the notification is sourced from.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An S3 bucket for the Lambda code .zip file
- An email address where you want to receive the violation notification

Limitations

- This detective control is regional and must be deployed in the AWS Regions you intend to monitor.

Product versions

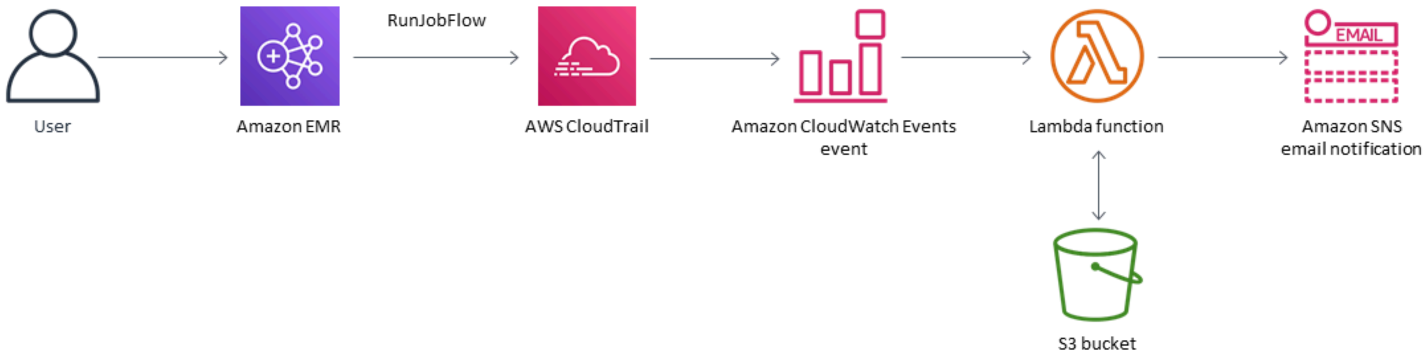
- Amazon EMR release 4.8.0 and later

Architecture

Target technology stack

- Amazon CloudWatch Events event
- Amazon EMR
- Lambda function
- S3 bucket
- Amazon SNS

Target architecture



Automation and scale

- If you are using AWS Organizations, you can use [AWS CloudFormation StackSets](#) to deploy this template in multiple accounts that you want to monitor.

Tools

Tools

- AWS CloudFormation – AWS CloudFormation helps you model and set up AWS resources using infrastructure as code.
- AWS Cloudwatch Events – AWS CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- Amazon EMR – Amazon EMR is a managed cluster platform that simplifies running big data frameworks.
- AWS Lambda – AWS Lambda supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon S3 is a web services interface that you can use to store and retrieve any amount of data from anywhere on the web.
- Amazon SNS – Amazon SNS is a web service that coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses.

Code

- A .zip file of the project is available as an attachment.

Epics

Define the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	To host the Lambda code .zip file, choose or create an S3 bucket with a unique name that does not contain leading	Cloud Architect

Task	Description	Skills required
	slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket needs to be in the same AWS Region as the Amazon EMR cluster that is being evaluated .	

Upload the Lambda code to the S3 bucket

Task	Description	Skills required
Upload the Lambda code to the S3 bucket.	Upload the Lambda code .zip file that's provided in the "Attachments" section to the S3 bucket. The S3 bucket must be in the same Region as the Amazon EMR cluster that is being evaluated.	Cloud Architect

Deploy the AWS CloudFormation template

Task	Description	Skills required
Deploy the AWS CloudFormation template.	On the AWS CloudFormation console, in the same Region as your S3 bucket, deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters. For more information about	Cloud Architect

Task	Description	Skills required
	deploying AWS CloudFormation templates, see the "Related resources" section.	

Complete the parameters in the AWS CloudFormation template

Task	Description	Skills required
Name the S3 bucket.	Enter the name of the S3 bucket that you created in the first epic.	Cloud Architect
Provide the Amazon S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, <directory>/<file-name>.zip).	Cloud Architect
Provide an email address.	Provide an active email address to receive Amazon SNS notifications.	Cloud Architect
Define the logging level.	Define the logging level and frequency for your Lambda function. "Info" designates detailed informational messages on the application’s progress. "Error" designates error events that could still allow the application to continue running. "Warning" designates potentially harmful situations.	Cloud Architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the template successfully deploys, it sends a subscription email message to the email address provided. You must confirm this email subscription to receive violation notifications.	Cloud Architect

Related resources

- AWS Lambda
- Amazon EMR logging
- Deploying AWS CloudFormation templates

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Generate test data using an AWS Glue job and Python

Moinul Al-Mamun, Amazon Web Services

Summary

This pattern shows you how to quickly and easily generate millions of sample files concurrently by creating an AWS Glue job written in Python. The sample files are stored in an Amazon Simple Storage Service (Amazon S3) bucket. The ability to quickly generate a large number of sample files is important for testing or evaluating services in the AWS Cloud. For example, you can test the performance of AWS Glue Studio or AWS Glue DataBrew jobs by performing data analysis on millions of small files in an Amazon S3 prefix.

Although you can use other AWS services to generate sample datasets, we recommend that you use AWS Glue. You don't need to manage any infrastructure because AWS Glue is a serverless data processing service. You can just bring your code and run it in an AWS Glue cluster. Additionally, AWS Glue provisions, configures, and scales the resources required to run your jobs. You pay only for the resources that your jobs use while running.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI), installed and configured to work with the AWS account

Product versions

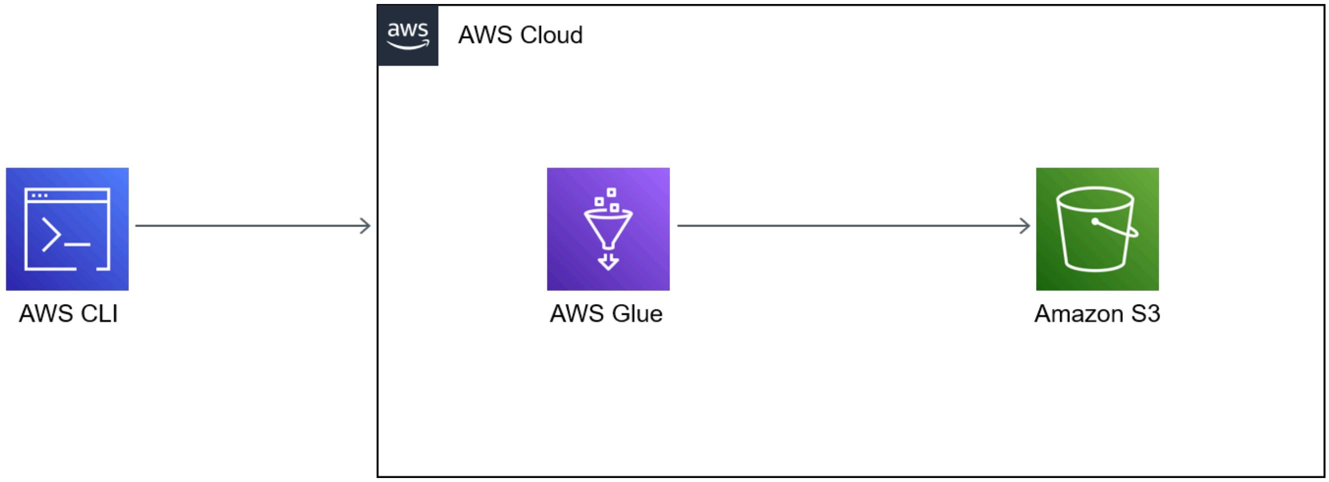
- Python 3.9
- AWS CLI version 2

Limitations

The maximum number of AWS Glue jobs per trigger is 50. For more information, see AWS Glue endpoints and quotas.

Architecture

The following diagram depicts an example architecture centered around an AWS Glue job that writes its output (that is, sample files) to an S3 bucket.



The diagram includes the following workflow:

1. You use the AWS CLI, AWS Management Console, or an API to initiate the AWS Glue job. The AWS CLI or API enables you to automate the parallelization of the invoked job and reduce the runtime for generating sample files.
2. The AWS Glue job generates file content randomly, converts the content into CSV format, and then stores the content as an Amazon S3 object under a common prefix. Each file is less than a kilobyte. The AWS Glue job accepts two user-defined job parameters: `START_RANGE` and `END_RANGE`. You can use these parameters to set file names and the number of files generated in Amazon S3 by each job run. You can run multiple instances of this job in parallel (for example, 100 instances).

Tools

- [Amazon Simple Storage Service (Amazon S3)] is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- [AWS Command Line Interface (AWS CLI)] is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- [AWS Glue] is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.

- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Best practices

Consider the following AWS Glue best practices as you implement this pattern:

- **Use the right AWS Glue worker type to reduce cost.** We recommend that you understand the different properties of worker types, and then choose the right worker type for your workload based on CPU and memory requirements. For this pattern, we recommend that you use a Python shell job as your job type to minimize DPU and reduce cost. For more information, see Adding jobs in AWS Glue in the AWS Glue Developer Guide.
- **Use the right concurrency limit to scale your job.** We recommend that you base the maximum concurrency of your AWS Glue job on your time requirement and required number of files.
- **Start generating a small number of files at first.** To reduce cost and save time when you build your AWS Glue jobs, start with a small number of files (such as 1,000). This can make troubleshooting easier. If generating a small number of files is successful, then you can scale to a larger number of files.
- **Run locally first.** To reduce cost and save time when you build your AWS Glue jobs, start the development locally and test your code. For instructions on setting up a Docker container that can help you write AWS Glue extract, transform, and load (ETL) jobs both in a shell and in an integrated development environment (IDE), see the Developing AWS Glue ETL jobs locally using a container post on the AWS Big Data Blog.

For more AWS Glue best practices, see Best practices in the AWS Glue documentation.

Epics

Create a destination S3 bucket and IAM role

Task	Description	Skills required
Create an S3 bucket for storing the files.	Create an S3 bucket and a prefix within it.	App developer

Task	Description	Skills required
	Note This pattern uses the `s3://{your-s3-bucket-name}/small-files/` location for demonstration purposes.	

Task	Description	Skills required

Create and configure an IAM role.	You must create an IAM role that your AWS Glue job can use to write to your S3 bucket.
	
	1. Create an IAM role (for example, called `"AWSGlueServiceRole-smallfiles"`).
	2. Choose AWS Glue as the policy's trusted entity.
	3. Attach an AWS managed policy called `"AWSGlueServiceRole"` to the role.
	4. Create an inline policy or customer managed policy called `"s3-small-file-access"` based on the following configuration. Replace `"{bucket}"` with your bucket name.
	
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Effect": "Allow",`
	`            "Action": [`
	`                "s3:GetObject",`
	`                "s3:PutObject"`
		App developer

Task	Description	Skills required

	``
	            ],
	            "Resource
	": [
	
	 "arn:aws:s3:::{buc
	ket}/small-files/i
	nput/*"
	            ]
	        }
	    ]
	}
	``
	5. Attach the `"s3-small-file-access"` policy to your role.	

Create and configure an AWS Glue job to handle concurrent runs

Task	Description	Skills required
Create an AWS Glue job.	You must create an AWS Glue job that generates your content and stores it in an S3 bucket. Create an AWS Glue job, and then configure your job by completing the following steps: 1. Sign in to the AWS Management Console and open the AWS Glue console. 2. In the navigation pane, under **Data Integration and ETL**, choose **Jobs**.	App developer

Task	Description	Skills required
	3. In the **Create job** section, choose **Python Shell script editor**. 4. In the **Options** section, select **Create a new script with boilerplate code**, and then choose **Create**. 5. Choose **Job details**. 6. For **Name**, enter **create_small_files**. 7. For **IAM Role**, select the IAM role that you created earlier. 8. In the **This job runs** section, choose **A new script to be authored by you**. 9. Expand **Advanced properties**. 10. For **Maximum concurrency**, enter **100** for demonstration purposes. **Note**: Maximum concurrency defines how many instances of the job that you can run in parallel. 11. Choose **Save**.	

Task	Description	Skills required

Update the job code.	1. Open the [AWS Glue console](). 2. In the navigation pane, choose **Jobs**. 3. In the **Your jobs** section, choose the job that you created earlier. 4. Choose the **Script** tab, and then update the script based on the following code. Update the `BUCKET_NAME` , `PREFIX`, and `text_str` variables with your values.
	 `from awsglue.utils import getResolvedOptions`
	`import sys`
	`import boto3`
	`from random import randrange`
	
	`# Two arguments`
	`args = getResolvedOptions(sys.argv, ['START_RANGE', 'END_RANGE'])`
	
	`START_RANGE = int(args['START_RANGE'])`
	`END_RANGE = int(args['END_RANGE'])`
	
	`BUCKET_NAME = '{BUCKET_NAME}'`
		App developer

Task	Description	Skills required

	``
	PREFIX = 'small-files/input/'
	s3 = boto3.resource('s3')
	for x in range(START_RANGE, END_RANGE):
	    # generate file name
	    file_name = f"input_{x}.txt"
	    # generate text
	    text_str = str(randrange(100000))+","+str(randrange(100000))+", " + str(randrange(10000000)) + "," + str(randrange(10000))
	    # write in s3
	    s3.Object(BUCKET_NAME, PREFIX + file_name).put(Body=text_str)
	``
	5. Choose **Save**.	

Run the AWS Glue job from the command line or console

Task	Description	Skills required
Run the AWS Glue job from the command line.	To run your AWS Glue job from the AWS CLI , run the following command using your values: ``` cmd:~$ aws glue start-job-run --job-name ```	App developer

Task	Description	Skills required
	``` create_small_files --arguments '{"--STAR T_RANGE":"0","--EN D_RANGE":"1000000"}' cmd:~$ aws glue start-job-run --job-name create_small_files --arguments '{"--STAR T_RANGE":"1000000" ,"--END_RANGE":"20 00000"}' ``` **Note** For instructions on running the AWS Glue job from the AWS Management Console, see the *Run the AWS Glue job in the AWS Management Console* story in this pattern. **Tip** We recommend using the AWS CLI to run AWS Glue jobs if you want to run multiple executions at a time with different parameters, as shown in the example above.	

Task	Description	Skills required

	To generate all AWS CLI commands that are required to generate a defined number of files using a certain parallelization factor, run the following bash code (using your values):
	
	``
	# define parameters
	NUMBER_OF_FILES=
	10000000;
	PARALLELIZATION=50;
	
	# initialize
	_SB=0;
	
	# generate commands
	for i in $(seq 1
	 $PARALLELIZATION);
	do
	      echo aws glue
	 start-job-run --
	job-name create_sm
	all_files --argumen
	ts "'"'{"--START_RANG
	E":"'$(((NUMBER_OF
	_FILES/PARALLELIZA
	TION) * (i-1) +
	 _SB))'","--END_RAN
	GE":"'$(((NUMBER_O
	F_FILES/PARALLELIZ
	ATION) * (i)))'"}'"'";
	      _SB=1;
	done
	``
	
	If you use the script above, consider the following:	

Task	Description	Skills required
	• The script simplifies the invocation and generation of small files at scale. • Update `NUMBER_OF_FILES` and `PARALLELIZATION` with your values. • The script above prints a list of commands that you must run. Copy those output commands, and then run them in your terminal. • If you want to run the commands directly from within the script, remove the `echo` statement in line 11. **Note** To see an example of output from the above script, see *Shell script output* in the *Additional information* section of this pattern.	

Task	Description	Skills required
Run the AWS Glue job in the AWS Management Console.	1. Sign in to the AWS Management Console and open the [AWS Glue console](https://console.aws.amazon.com/glue/). 2. In the navigation pane, under **Data Integration and ETL**, choose **Jobs**. 3. In the **Your jobs** section, choose your job. 4. In the **Parameters (optional)** section, update your parameters. 5. Choose **Action**, and then choose **Run job**. 6. Repeat steps 3-5 as many times as you require. For example, to create 10 million files, repeat this process 10 times.	App developer

Task	Description	Skills required
Check the status of your AWS Glue job.	1. Open the AWS Glue console. 2. In the navigation pane, choose **Jobs**. 3. In the **Your jobs** section, choose the job that you created earlier (that is, `create_small_files`). 4. For insight into the progress and generation of your files, review the **Run ID**, **Run Status**, and other columns.	App developer

Related resources

References

- Registry of Open Data on AWS
- Data sets for analytics
- Open Data on AWS
- Adding jobs in AWS Glue
- Getting started with AWS Glue

Guides and patterns

- AWS Glue best practices
- Load testing applications

Additional information

Benchmarking test

This pattern was used to generate 10 million files using different parallelization parameters as part of a benchmarking test. The following table shows the output of the test:

Parallelization	Number of files generated by a job run	Job duration	Speed
10	1,000,000	6 hours, 40 minutes	Very slow
50	200,000	80 minutes	Moderate
100	100,000	40 minutes	Fast

If you want to make the process faster, you can configure more concurrent runs in your job configuration. You can easily adjust the job configuration based on your requirements, but keep in mind that there is an AWS Glue service quota limit. For more information, see [AWS Glue endpoints and quotas](#).

Shell script output

The following example shows the output of the shell script from the *Run the AWS Glue job from the command line* story in this pattern.

```
user@MUC-1234567890 MINGW64 ~
  $ # define parameters
  NUMBER_OF_FILES=10000000;
  PARALLELIZATION=50;
  # initialize
  _SB=0;

  # generate commands
  for i in $(seq 1 $PARALLELIZATION);
   do
         echo aws glue start-job-run --job-name create_small_files --arguments
 "'"'{"--START_RANGE":"'$(((NUMBER_OF_FILES/PARALLELIZATION)  (i-1) + SB))'","--
ENDRANGE":"'$(((NUMBER_OF_FILES/PARALLELIZATION)  (i)))'"}'"'";
         _SB=1;
```

```
    done

  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"0","--END_RANGE":"200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"200001","--END_RANGE":"400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"400001","--END_RANGE":"600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"600001","--END_RANGE":"800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"800001","--END_RANGE":"1000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1000001","--END_RANGE":"1200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1200001","--END_RANGE":"1400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1400001","--END_RANGE":"1600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1600001","--END_RANGE":"1800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"1800001","--END_RANGE":"2000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2000001","--END_RANGE":"2200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2200001","--END_RANGE":"2400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2400001","--END_RANGE":"2600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2600001","--END_RANGE":"2800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"2800001","--END_RANGE":"3000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3000001","--END_RANGE":"3200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3200001","--END_RANGE":"3400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3400001","--END_RANGE":"3600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3600001","--END_RANGE":"3800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"3800001","--END_RANGE":"4000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4000001","--END_RANGE":"4200000"}'
```

```
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4200001","--END_RANGE":"4400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4400001","--END_RANGE":"4600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4600001","--END_RANGE":"4800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"4800001","--END_RANGE":"5000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5000001","--END_RANGE":"5200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5200001","--END_RANGE":"5400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5400001","--END_RANGE":"5600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5600001","--END_RANGE":"5800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"5800001","--END_RANGE":"6000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6000001","--END_RANGE":"6200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6200001","--END_RANGE":"6400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6400001","--END_RANGE":"6600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6600001","--END_RANGE":"6800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"6800001","--END_RANGE":"7000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7000001","--END_RANGE":"7200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7200001","--END_RANGE":"7400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7400001","--END_RANGE":"7600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7600001","--END_RANGE":"7800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"7800001","--END_RANGE":"8000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8000001","--END_RANGE":"8200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8200001","--END_RANGE":"8400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8400001","--END_RANGE":"8600000"}'
```

```
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8600001","--END_RANGE":"8800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"8800001","--END_RANGE":"9000000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9000001","--END_RANGE":"9200000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9200001","--END_RANGE":"9400000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9400001","--END_RANGE":"9600000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9600001","--END_RANGE":"9800000"}'
  aws glue start-job-run --job-name create_small_files --arguments '{"--
START_RANGE":"9800001","--END_RANGE":"10000000"}'

  user@MUC-1234567890 MINGW64 ~
```

FAQ

How many concurrent runs or parallel jobs should I use?

The number of concurrent runs and parallel jobs depend on your time requirement and desired number of test files. We recommend that you check the size of the files that you're creating. First, check how much time an AWS Glue job takes to generate your desired number of files. Then, use the right number of concurrent runs to meet your goals. For example, if you assume that 100,000 files takes 40 minutes to complete the run but your target time is 30 minutes, then you must increase the concurrency setting for your AWS Glue job.

What type of content can I create using this pattern?

You can create any type of content, such as text files with different delimiters (for example, PIPE, JSON, or CSV). This pattern uses Boto3 to write to a file and then saves the file in an S3 bucket.

What level of IAM permission do I need in the S3 bucket?

You must have an identity-based policy that allows `Write` access to objects in your S3 bucket. For more information, see [Amazon S3: Allows read and write access to objects in an S3 bucket]() in the Amazon S3 documentation.

Cost-effectively ingest IoT data directly into Amazon S3 using AWS IoT Greengrass

Sebastian Viviani and Rizwan Syed, Amazon Web Services

Summary

This pattern shows you how to cost-effectively ingest Internet of Things (IoT) data directly into an Amazon Simple Storage Service (Amazon S3) bucket by using an AWS IoT Greengrass Version 2 device. The device runs a custom component that reads the IoT data and saves the data in persistent storage (that is, a local disk or volume). Then, the device compresses the IoT data into an Apache Parquet file and uploads the data periodically to an S3 bucket.

The amount and speed of IoT data that you ingest is limited only by your edge hardware capabilities and network bandwidth. You can use Amazon Athena to cost-effectively analyze your ingested data. Athena supports compressed Apache Parquet files and data visualization by using Amazon Managed Grafana.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An edge gateway that runs on AWS IoT Greengrass Version 2 and collects data from sensors (The data sources and the data collection process are beyond the scope of this pattern, but you can use nearly any type of sensor data. This pattern uses a local MQTT broker with sensors or gateways that publish data locally.)
- AWS IoT Greengrass component, roles, and SDK dependencies
- A stream manager component to upload the data to the S3 bucket
- AWS SDK for Java, AWS SDK for JavaScript, or AWS SDK for Python (Boto3) to run the APIs

Limitations

- The data in this pattern isn't uploaded in real time to the S3 bucket. There is a delay period, and you can configure the delay period. Data is buffered temporarily in the edge device and then uploaded once the period expires.
- The SDK is available only in Java, Node.js, and Python.

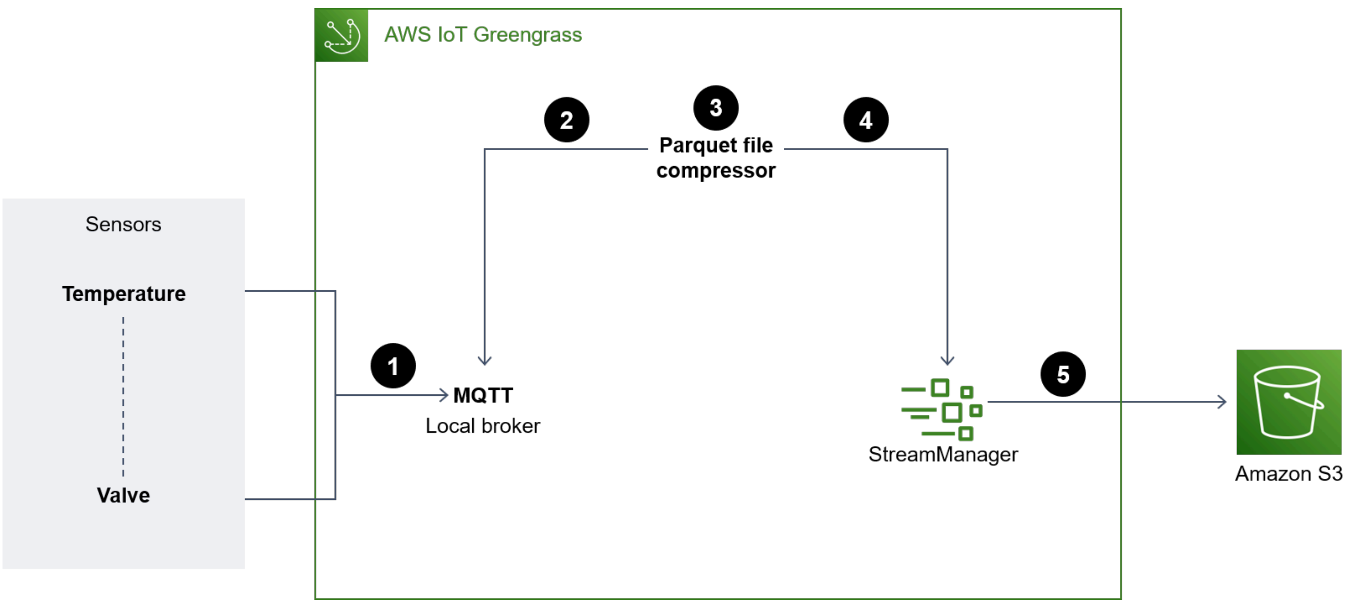
Architecture

Target technology stack

- Amazon S3
- AWS IoT Greengrass
- MQTT broker
- Stream manager component

Target architecture

The following diagram shows an architecture designed to ingest IoT sensor data and store that data in an S3 bucket.



The diagram shows the following workflow:

1. Multiple sensors (for example, temperature and valve) updates are published to a local MQTT broker.
2. The Parquet file compressor that's subscribed to these sensors updates topics and receives these updates.
3. The Parquet file compressor stores the updates locally.
4. After the period lapses, the stored files are compressed into Parquet files and passed on to the stream manager to get uploaded to the specified S3 bucket.

5. The stream manager uploads the Parquet files to the S3 bucket.

> **Note**
>
> The stream manager (`StreamManager`) is a managed component. For examples of how to export data to Amazon S3, see Stream manager in the AWS IoT Greengrass documentation. You can use a local MQTT broker as a component or another broker like Eclipse Mosquitto.

Tools

AWS tools

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS IoT Greengrass is an open source IoT edge runtime and cloud service that helps you build, deploy, and manage IoT applications on your devices.

Other tools

- Apache Parquet is an open-source column-oriented data file format designed for storage and retrieval.
- MQTT (Message Queuing Telemetry Transport) is a lightweight messaging protocol that's designed for constrained devices.

Best practices

Use the right partition format for uploaded data

There are no specific requirements for the root prefix names in the S3 bucket (for example, `"myAwesomeDataSet/"` or `"dataFromSource"`), but we recommend that you use a meaningful partition and prefix so that it's easy to understand the purpose of the dataset.

We also recommend that you use the right partitioning in Amazon S3 so that the queries run optimally on the dataset. In the following example, the data is partitioned in HIVE format so that the amount of data scanned by each Athena query is optimized. This improves performance and reduces cost.

```
s3://<ingestionBucket>/<rootPrefix>/year=YY/month=MM/day=DD/
HHMM_<suffix>.parquet
```

Epics

Set up your environment

Task	Description	Skills required
Create an S3 bucket.	1. Create an S3 bucket or use an existing bucket. 2. Create a meaningful prefix for the S3 bucket where you want to ingest the IoT data (for example, `s3:\\<bucket>\<prefix>`). 3. Record your prefix for later use.	App developer

Add IAM permissions to the S3 bucket.	To grant users write access to the S3 bucket and prefix that you created earlier, add the following IAM policy to your AWS IoT Greengrass role:
	`{`
	`"Version": "2012-10-17",`
	`"Statement": [`
	`{`
	`"Sid": "S3DataUpload",`
	`"Effect": "Allow",`
	`"Action": [`
		App developer

Task	Description	Skills required

	``
	 "s3:List*",
	
	 "s3:Put*"
	            ],
	            "Resource":
	 [
	
	 "arn:aws:s3:::<ing
	estionBucket>",
	
	 "arn:aws:s3:::<ing
	estionBucket>/<pre
	fix>/*"
	            ]
	        }
	    ]
	}
	``
	
	For more information, see [Creating an IAM policy to access Amazon S3 resources](#) in the Aurora documentation.
	
	Next, update the resource policy (if needed) for the S3 bucket to allow write access with the correct AWS [principals](#).	

Build and deploy the AWS IoT Greengrass component

Task	Description	Skills required
Update the recipe of the component.	[Update the component configuration](#) when you	App developer

Task	Description	Skills required

	create a deployment based on the following example:
	
	``
	{
	  "region": "<region>",
	  "parquet_period": <period>,
	  "s3_bucket": "<s3Bucket>",
	  "s3_key_prefix": "<s3prefix>"
	}
	``
	
	Replace `<region>` with your AWS Region, `<period>` with your periodic interval, `<s3Bucket>` with your S3 bucket, and `<s3prefix>` with your prefix.	

Task	Description	Skills required
Create the component.	Do one of the following: • Create the component. • Add the component to the CI/CD pipeline (if one exists). Be sure to copy the artifact from the artifact repository to the AWS IoT Greengrass artifact bucket. Then, create or update your AWS IoT Greengrass component. • **Note** Add the MQTT broker as a component or add it manually later. : This decision affects the authentication scheme that you can use with the broker. Manually adding a broker decouples the broker from AWS IoT Greengras s and enables any supported authentication scheme of the broker. The AWS provided broker components	App developer

Task	Description	Skills required
	have predefine d authentication schemes. For more information, see [MQTT 3.1.1 broker (Moquette)]() and [MQTT 5 broker (EMQX)]().	
Update the MQTT client.	The sample code doesn't use authentication because the component connects locally to the broker. If your scenario differs, update the MQTT client section as needed. Additionally, do the following: 1. Update the MQTT topics in the subscription. 2. Update the MQTT message parser as needed as messages from each source may differ.	App developer

Add the component to the AWS IoT Greengrass Version 2 core device

Task	Description	Skills required
Update the deployment of the core device.	If the deployment of the AWS IoT Greengrass Version 2 core device already exists, [revise the deployment](). If the deployment doesn't exist, [create a new deployment]().	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

To give the component the correct name, update the log manager configuration for the new component (if needed) based on the following:

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
    ...
    },
    "componentLogsConfigurationMap": {
      "<com.iot.ingest.parquet>": {
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
      ...
    }
  },
  "periodicUploadIntervalSec": "300"
}
```

Finally, complete the revision of the deployment for your AWS IoT Greengrass core device.

</td><td></td></tr>
</table>

Verify data ingestion into the S3 bucket

Task	Description	Skills required
Check the logs for the AWS IoT Greengrass volume.	Check for the following: • The MQTT client is successfully connected to the local MQTT broker. • The MQTT client is subscribed to the correct topics. • Sensor update messages are coming to the broker on the MQTT topics. • Parquet compression happens at every periodic interval.	App developer
Check the S3 bucket.	Verify if the data is being uploaded to the S3 bucket. You can see the files being uploaded at every period. You can also verify if the data is uploaded to the S3 bucket by querying the data in the next section.	App developer

Set up querying from Athena

Task	Description	Skills required
Create a database and table.	1. Create an AWS Glue database (if needed).	App developer

Task	Description	Skills required
	2. Create a table in AWS Glue manually or by running a crawler in AWS Glue.	
Grant Athena access to the data.	1. Update permissions to allow Athena to access the S3 bucket. For more information, see Fine-grained access to databases and tables in the AWS Glue Data Catalog in the Athena documentation. 2. Query the table in your database.	App developer

Troubleshooting

Issue	Solution
MQTT client fails to connect	• Validate the permissions on the MQTT broker. If you have an MQTT broker from AWS, see MQTT 3.1.1 broker (Moquette) and MQTT 5 broker (EMQX). • Validate the credentials on the MQTT client. If you have an MQTT broker from AWS, see MQTT 3.1.1 broker (Moquette) and MQTT 5 broker (EMQX).
MQTT client fails to subscribe	Validate the permissions on the MQTT broker. If you have an MQTT broker from AWS, see MQTT 3.1.1 broker (Moquette) and MQTT 5 broker (EMQX).
Parquet files don't get created	• Verify that the MQTT topics are correct.

Issue	Solution
	• Verify that the MQTT messages from the sensors are in the correct format.
Objects are not uploaded to the S3 bucket	• Verify that you have internet connectivity and endpoint connectivity. • Verify that the resource policy for your S3 bucket is correct. • Verify the permissions for the AWS IoT Greengrass Version 2 core device role.

Related resources

- DataFrame (Pandas documentation)
- Apache Parquet Documentation (Parquet documentation)
- Develop AWS IoT Greengrass components (AWS IoT Greengrass Developer Guide, Version 2)
- Deploy AWS IoT Greengrass components to devices (AWS IoT Greengrass Developer Guide, Version 2)
- Interact with local IoT devices (AWS IoT Greengrass Developer Guide, Version 2)
- MQTT 3.1.1 broker (Moquette) (AWS IoT Greengrass Developer Guide, Version 2)
- MQTT 5 broker (EMQX) (AWS IoT Greengrass Developer Guide, Version 2)

Additional information

Cost analysis

The following cost analysis scenario demonstrates how the data ingestion approach covered in this pattern can impact data ingestion costs in the AWS Cloud. The pricing examples in this scenario are based on prices at the time of publication. Prices are subject to change. Additionally, your costs may vary depending on your AWS Region, AWS service quotas, and other factors related to your cloud environment.

Input signal set

This analysis uses the following set of input signals as the basis for comparing IoT ingestion costs with other available alternatives.

Number of signals	Frequency	Data per signal
125	25 Hz	8 bytes

In this scenario, the system receives 125 signals. Each signal is 8 bytes and occurs every 40 milliseconds (25 Hz). These signals could come individually or grouped in a common payload. You have the option to split and pack these signals based on your needs. You can also determine the latency. Latency consists of the time period for receiving, accumulating, and ingesting the data.

For comparison purposes, the ingestion operation for this scenario is based in the `us-east-1` AWS Region. The cost comparison applies to AWS services only. Other costs, like hardware or connectivity, are not factored into the analysis.

Cost comparisons

The following table shows the monthly cost in US dollars (USD) for each ingestion method.

Method	Monthly cost
AWS IoT SiteWise*	331.77 USD
AWS IoT SiteWise Edge with data processing pack (keeping all data at the edge)	200 USD
AWS IoT Core and Amazon S3 rules for accessing raw data	84.54 USD
Parquet file compression at the edge and uploading to Amazon S3	0.5 USD

*Data must be downsampled to comply with service quotas. This means there is some data loss with this method.

Alternative methods

This section shows the equivalent costs for the following alternative methods:

- **AWS IoT SiteWise –** Each signal must be uploaded in an individual message. Therefore, the total number of messages per month is 125×25×3600×24×30, or 8.1 billion messages per month. However, AWS IoT SiteWise can handle only 10 data points per second per property. Assuming the data is downsampled to 10 Hz, the number of messages per month is reduced to 125×10×3600×24×30, or 3.24 billion. If you use the publisher component that packs measurements in groups of 10 (at 1 USD per million messages), then you get a monthly cost of 324 USD per month. Assuming that each message is 8 bytes (1 Kb/125), that's 25.92 Gb of data storage. This adds a monthly cost of 7.77 USD per month. The total cost for the first month is 331.77 USD and increases by 7.77 USD every month.
- **AWS IoT SiteWise Edge with data processing pack, including all models and signals fully processed at the edge (that is, no cloud ingestion) –** You can use the data processing pack as an alternative to reduce costs and to configure all the models that get calculated at the edge. This can work just for storage and visualization, even if no real calculation is performed. In this case, it's necessary to use powerful hardware for the edge gateway. There is a fixed cost of 200 USD per month.
- **Direct ingestion to AWS IoT Core by MQTT and an IoT rule to store the raw data in Amazon S3 –** Assuming all the signals are published in a common payload, the total number of messages published to AWS IoT Core is 25×3600×24×30, or 64.8 million per month. At 1 USD per million messages, that's a monthly cost of 64.8 USD per month. At 0.15 USD per million rule activations and with one rule per message, that adds a monthly cost of 19.44 USD per month. At a cost of 0.023 USD per Gb of storage in Amazon S3, that adds another 1.5 USD per month (increasing every month to reflect the new data). The total cost for the first month is 84.54 USD and increases by 1.5 USD every month.
- **Compressing data at the edge in a Parquet file and uploading to Amazon S3 (proposed method) –** The compression ratio depends on the type of data. With the same industrial data tested for MQTT, the total output data for a full month is 1.2 Gb. This costs 0.03 USD per month. Compression ratios (using random data) described in other benchmarks are on the order of 66 percent (closer to a worst-case scenario). The total data is 21 Gb and costs 0.5 USD per month.

Parquet file generator

The following code example shows the structure of a Parquet file generator that's written in Python. The code example is for illustration purposes only and won't work if pasted into your environment.

```
import queue
```

```
import paho.mqtt.client as mqtt
import pandas as pd

#queue for decoupling the MQTT thread
messageQueue = queue.Queue()
client = mqtt.Client()
streammanager = StreamManagerClient()

def feederListener(topic, message):
    payload = {
        "topic" : topic,
        "payload" : message,
    }
    messageQueue.put_nowait(payload)

def on_connect(client_instance, userdata, flags, rc):
    client.subscribe("#",qos=0)

def on_message(client, userdata, message):
    feederListener(topic=str(message.topic),
 message=str(message.payload.decode("utf-8")))


filename = "tempfile.parquet"
streamname = "mystream"
destination_bucket= "amzn-s3-demo-bucket"
keyname="mykey"
period= 60

client.on_connect = on_connect
client.on_message = on_message
streammanager.create_message_stream(
            MessageStreamDefinition(name=streamname,
 strategy_on_full=StrategyOnFull.OverwriteOldestData)
        )


while True:
   try:
       message = messageQueue.get(timeout=myArgs.mqtt_timeout)
   except (queue.Empty):
       logger.warning("MQTT message reception timed out")

   currentTimestamp = getCurrentTime()
```

```
  if currentTimestamp >= nextUploadTimestamp:
      df = pd.DataFrame.from_dict(accumulator)
      df.to_parquet(filename)
      s3_export_task_definition = S3ExportTaskDefinition(input_url=filename,
bucket=destination_bucket, key=key_name)
      streammanager.append_message(streamname,
Util.validate_and_serialize_to_json_bytes(s3_export_task_definition))
      accumulator = {}
      nextUploadTimestamp += period
  else:
       accumulator.append(message)
```

Launch a Spark job in a transient EMR cluster using a Lambda function

Dhrubajyoti Mukherjee, Amazon Web Services

Summary

This pattern uses the Amazon EMR RunJobFlow API action to launch a transient cluster to run a Spark job from a Lambda function. A transient EMR cluster is designed to terminate as soon as the job is complete or if any error occurs. A transient cluster provides cost savings because it runs only during the computation time, and it provides scalability and flexibility in a cloud environment.

The transient EMR cluster is launched using the Boto3 API and the Python programming language in a Lambda function. The Lambda function, which is written in Python, provides the added flexibility of initiating the cluster when it is needed.

To demonstrate a sample batch computation and output, this pattern will launch a Spark job in an EMR cluster from a Lambda function and run a batch computation against the example sales data of a fictional company. The output of the Spark job will be a comma-separated values (CSV) file in Amazon Simple Storage Service (Amazon S3). The input data file, Spark .jar file, a code snippet, and an AWS CloudFormation template for a virtual private cloud (VPC) and AWS Identity and Access Management (IAM) roles to run the computation are provided as an attachment.

Prerequisites and limitations

Prerequisites

- An active AWS account

Limitations

- Only one Spark job can be initiated from the code at a time.

Product versions

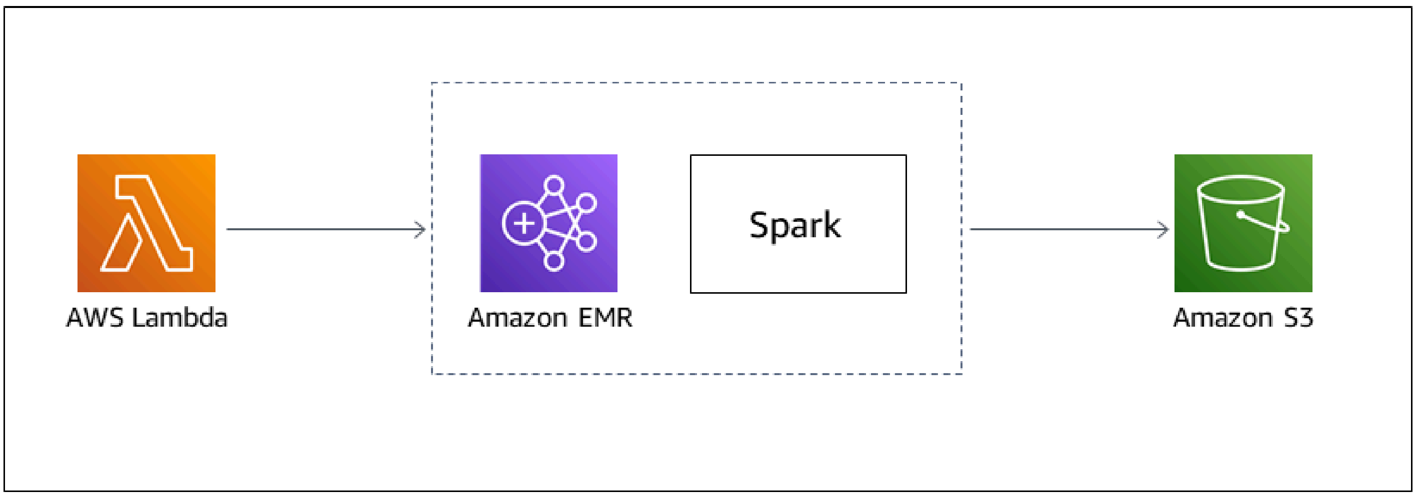
- Tested on Amazon EMR 6.0.0

Architecture

Target technology stack

- Amazon EMR
- AWS Lambda
- Amazon S3
- Apache Spark

Target architecture



Automation and scale

To automate the Spark-EMR batch computation, you can use either of the following options.

- Implement an Amazon EventBridge rule that can initiate the Lambda function in a cron schedule. For more information, see [Tutorial: Schedule AWS Lambda functions using EventBridge].
- Configure [Amazon S3 event notifications] to initiate the Lambda function on file arrival.
- Pass the input parameters to the AWS Lambda function through the event body and Lambda environment variables.

Tools

AWS services

- Amazon EMR is a managed cluster platform that simplifies running big data frameworks on AWS to process and analyze large amounts of data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Apache Spark is a multiple-language analytics engine for large-scale data processing.

Epics

Create the Amazon EMR and Lambda IAM roles and the VPC

Task	Description	Skills required
Create the IAM roles and the VPC.	If you already have the AWS Lambda and Amazon EMR IAM roles and a VPC, you can skip this step. To run the code, both the EMR cluster and the Lambda function require IAM roles. The EMR cluster also requires a VPC with a public subnet or a private subnet with a NAT gateway. To automatically create all the IAM roles and a VPC, deploy the attached AWS CloudForm ation template as is, or you can create the roles and the VPC manually as specified in	Cloud architect

Task	Description	Skills required
	the *Additional information* section.	
Note the AWS CloudFormation template output keys.	After the CloudFormation template has successfully deployed, navigate to the **Outputs** tab in the AWS CloudFormation console. Note the five output keys: • `S3Bucket` • `LambdaExecutionRole` • `ServiceRole` • `JobFlowRole` • `Ec2SubnetId` You will use the values from these keys when you create the Lambda function.	Cloud architect

Upload the Spark .jar file

Task	Description	Skills required
Upload the Spark .jar file.	Upload the Spark .jar file to the S3 bucket that the AWS CloudFormation stack created. The bucket name is the same as the output key `S3Bucket`.	General AWS

Create the Lambda function to launch the EMR cluster

Task	Description	Skills required
Create a Lambda function.	On the Lambda console, create a Python 3.9+ Lambda function with an execution role. The execution role policy must allow Lambda to launch an EMR cluster. (See the attached AWS CloudForm ation template.)	Data engineer, Cloud engineer
Copy and paste the code.	Replace the code in the `lambda_function.py` file with the code from the *Additional information* section of this pattern.	Data engineer, Cloud engineer
Change the parameters in the code.	Follow the comments in the code to change the parameter values to match your AWS account.	Data engineer, Cloud engineer
Launch the function to initiate the cluster.	Launch the function to initiate the creation of a transient EMR cluster with the Spark .jar file provided. It will run the Spark job and terminate automatically when the job is complete.	Data engineer, Cloud engineer
Check the EMR cluster status.	After the EMR cluster is initiated, it appears in the Amazon EMR console under the **Clusters** tab. Any errors while launching the cluster	Data engineer, Cloud engineer

Task	Description	Skills required
	or running the job can be be checked accordingly.	

Set up and run the sample demo

Task	Description	Skills required
Upload the Spark .jar file.	Download the Spark .jar file from the *Attachments* section and upload it to the S3 bucket.	Data engineer, Cloud engineer
Upload the input dataset.	Upload the attached `fake_sales_data.csv` file to the S3 bucket.	Data engineer, Cloud engineer
Paste the Lambda code and change the parameters.	Copy the code from the **Tools** section, and paste the code in a Lambda function, replacing the code `lambda_function.py` file. Change the parameter values to match your account.	Data engineer, Cloud engineer
Launch the function and verify the output.	After the Lambda function initiates the cluster with the provided Spark job, it generates a .csv file in the S3 bucket.	Data engineer, Cloud engineer

Related resources

- [Building Spark]
- [Apache Spark and Amazon EMR]

- [Boto3 Docs run_job_flow documentation]
- [Apache Spark information and documentation]

Additional information

Code

```
"""
Copy paste the following code in your Lambda function. Make sure to change the
 following key parameters for the API as per your account

-Name (Name of Spark cluster)
-LogUri (S3 bucket to store EMR logs)
-Ec2SubnetId (The subnet to launch the cluster into)
-JobFlowRole (Service role for EC2)
-ServiceRole (Service role for Amazon EMR)

The following parameters are additional parameters for the Spark job itself. Change the
 bucket name and prefix for the Spark job (located at the bottom).

-s3://your-bucket-name/prefix/lambda-emr/SparkProfitCalc.jar (Spark jar file)
-s3://your-bucket-name/prefix/fake_sales_data.csv (Input data file in S3)
-s3://your-bucket-name/prefix/outputs/report_1/ (Output location in S3)
"""
import boto3

client = boto3.client('emr')


def lambda_handler(event, context):
    response = client.run_job_flow(
        Name='spark_job_cluster',
        LogUri='s3://your-bucket-name/prefix/logs',
        ReleaseLabel='emr-6.0.0',
        Instances={
            'MasterInstanceType': 'm5.xlarge',
            'SlaveInstanceType': 'm5.large',
            'InstanceCount': 1,
            'KeepJobFlowAliveWhenNoSteps': False,
            'TerminationProtected': False,
            'Ec2SubnetId': 'subnet-XXXXXXXXXXXXXX'
        },
```

```
        Applications=[{'Name': 'Spark'}],
        Configurations=[
            {'Classification': 'spark-hive-site',
             'Properties': {
                 'hive.metastore.client.factory.class':
 'com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory'}
             }
        ],
        VisibleToAllUsers=True,
        JobFlowRole='EMRLambda-EMREC2InstanceProfile-XXXXXXXXX',
        ServiceRole='EMRLambda-EMRRole-XXXXXXXXX',
        Steps=[
            {
                'Name': 'flow-log-analysis',
                'ActionOnFailure': 'TERMINATE_CLUSTER',
                'HadoopJarStep': {
                    'Jar': 'command-runner.jar',
                    'Args': [
                        'spark-submit',
                        '--deploy-mode', 'cluster',
                        '--executor-memory', '6G',
                        '--num-executors', '1',
                        '--executor-cores', '2',
                        '--class', 'com.aws.emr.ProfitCalc',
                        's3://your-bucket-name/prefix/lambda-emr/SparkProfitCalc.jar',
                        's3://your-bucket-name/prefix/fake_sales_data.csv',
                        's3://your-bucket-name/prefix/outputs/report_1/'
                    ]
                }
            }
        ]
    )
```

IAM roles and VPC creation

To launch the EMR cluster in a Lambda function, a VPC and IAM roles are needed. You can set up the VPC and IAM roles by using the AWS CloudFormation template in the Attachments section of this pattern, or you can manually create them by using the following links.

The following IAM roles are required to run Lambda and Amazon EMR.

Lambda execution role

A Lambda function's execution role grants it permission to access AWS services and resources.

Service role for Amazon EMR

The Amazon EMR role defines the allowable actions for Amazon EMR when provisioning resources and performing service-level tasks that are not performed in the context of an Amazon Elastic Compute Cloud (Amazon EC2) instance running within a cluster. For example, the service role is used to provision EC2 instances when a cluster launches.

Service role for EC2 instances

The service role for cluster EC2 instances (also called the EC2 instance profile for Amazon EMR) is a special type of service role that is assigned to every EC2 instance in an Amazon EMR cluster when the instance launches. Application processes that run on top of Apache Hadoop assume this role for permissions to interact with other AWS services.

VPC and subnet creation

You can create a VPC from the VPC console.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Migrate Apache Cassandra workloads to Amazon Keyspaces by using AWS Glue

Nikolai Kolesnikov, Karthiga Priya Chandran, and Samir Patel, Amazon Web Services

Summary

This pattern shows you how to migrate your existing Apache Cassandra workloads to Amazon Keyspaces (for Apache Cassandra) by using CQLReplicator on AWS Glue. You can use CQLReplicator on AWS Glue to minimize the replication lag of migrating your workloads down to a matter of minutes. You also learn how to use an Amazon Simple Storage Service (Amazon S3) bucket to store data required for the migration, including Apache Parquet files, configuration files, and scripts. This pattern assumes that your Cassandra workloads are hosted on Amazon Elastic Compute Cloud (Amazon EC2) instances in a virtual private cloud (VPC).

Prerequisites and limitations

Prerequisites

- Cassandra cluster with a source table
- Target table in Amazon Keyspaces to replicate the workload
- S3 bucket to store intermediate Parquet files that contain incremental data changes
- S3 bucket to store job configuration files and scripts

Limitations

- CQLReplicator on AWS Glue requires some time to provision Data Processing Units (DPUs) for the Cassandra workloads. The replication lag between the Cassandra cluster and the target keyspace and table in Amazon Keyspaces is likely to last for only a matter of minutes.

Architecture

Source technology stack

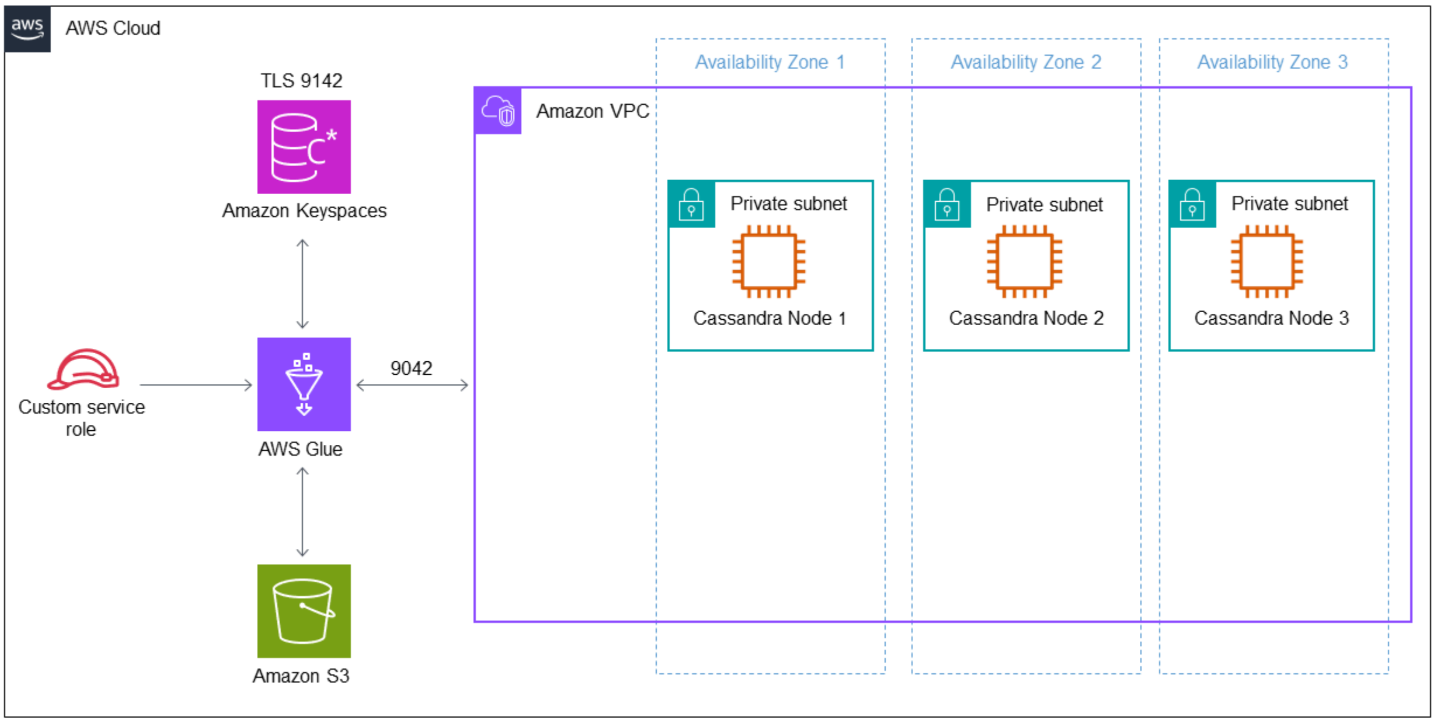
- Apache Cassandra
- DataStax Server
- ScyllaDB

Target technology stack

- Amazon Keyspaces

Migration architecture

The following diagram shows an example architecture where a Cassandra cluster is hosted on EC2 instances and spread across three Availability Zones. The Cassandra nodes are hosted in private subnets.



The diagram shows the following workflow:

1. A custom service role provides access to Amazon Keyspaces and the S3 bucket.
2. An AWS Glue job reads the job configuration and scripts in the S3 bucket.
3. The AWS Glue job connects through port 9042 to read data from the Cassandra cluster.
4. The AWS Glue job connects through port 9142 to write data to Amazon Keyspaces.

Tools

AWS services and tools

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS CloudShell is a browser-based shell that you can use to manage AWS services by using the AWS Command Line Interface (AWS CLI) and a range of preinstalled development tools.
- AWS Glue is a fully managed ETL service that helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- Amazon Keyspaces (for Apache Cassandra) is a managed database service that helps you migrate, run, and scale your Cassandra workloads in the AWS Cloud.

Code

The code for this pattern is available in the GitHub CQLReplicator repository.

Best practices

- To determine the necessary AWS Glue resources for the migration, estimate the number of rows in the source Cassandra table. For example, 250 K rows per 0.25 DPU (2 vCPUs, 4 GB of memory) with 84 GB disk.
- Pre-warm Amazon Keyspaces tables before running CQLReplicator. For example, eight CQLReplicator tiles (AWS Glue jobs) can write up to 22 K WCUs per second, so the target should be pre-warmed up to 25-30 K WCUs per second.
- To enable communication between AWS Glue components, use a self-referencing inbound rule for all TCP ports in your security group.
- Use the incremental traffic strategy to distribute the migration workload over time.

Epics

Deploy CQLReplicator

Task	Description	Skills required
Create a target keyspace and table.	1. Create a keyspace and table in Amazon Keyspaces. For more information on write capacity, see *Write*	App owner, AWS administr ator, DBA, App developer

Task	Description	Skills required
	unit calculations in the Additional information section of this pattern. You can also create a keyspace by using the Cassandra Query Language (CQL). For more informati on, see *Create a keyspace by using CQL* in the Additional information section of this pattern. **Note** After you create the table, consider switching the table to on-demand capacity mode to avoid unnecessary charges. 2. To update to throughput mode, run the following script: `ALTER TABLE target_keyspace.target_table WITH CUSTOM_PROPERTIES = { 'capacity_mode': { 'throughput_mode': 'PAY_PER_REQUEST'} }`	

Task	Description	Skills required

Configure the Cassandra driver to connect to Cassandra.	Use the following configuration script:
	<pre>Datastax-java-driver {
	  basic.request.consistency = "LOCAL_QUORUM"
	  basic.contact-points = ["127.0.0.1:9042"]
	   advanced.reconnect-on-init = true
	   basic.load-balancing-policy {
	        local-datacenter = "datacenter1"
	}
	advanced.auth-provider = {
	       class = PlainTextAuthProvider
	       username = "user-at-sample"
	       password = "S@MPLE=PASSWORD="
	}
	}</pre>
	Note
	The preceding script uses the Spark Cassandra Connector. For more information, see the reference configuration for Cassandra.
		DBA

Task	Description	Skills required

Configure the Cassandra driver to connect to Amazon Keyspaces.	Use the following configura tion script:
	
	`datastax-java-driver {`
	`basic {`
	`  load-balancing-pol`
	`icy {`
	`    local-datacenter =`
	` us-west-2`
	`        }`
	`  contact-points = [`
	`            "cassandr`
	`a.us-west-2.amazon`
	`aws.com:9142"`
	`        ]`
	`  request {`
	`  page-size = 2500`
	`  timeout = 360 seconds`
	`  consistency =`
	` LOCAL_QUORUM`
	`        }`
	`    }`
	`advanced {`
	` control-connection {`
	`  timeout = 360 seconds`
	`        }`
	` session-leak.thres`
	`hold = 6`
	` connection {`
	` connect-timeout = 360`
	` seconds`
	` init-query-timeout =`
	` 360 seconds`
	` warn-on-init-error =`
	` false`
	`        }`
	` auth-provider = {`
	`  class = software.`
	`aws.mcs.auth.SigV4`
	`AuthProvider`
		DBA

Task	Description	Skills required

	``
	  aws-region = us-
	west-2
	 }
	
	 ssl-engine-factory {
	  class = DefaultSs
	lEngineFactory
	        }
	    }
	}
	``
	
	Note
	
	The preceding script uses the Spark Cassandra Connector. For more information, see the reference configuration for [Cassandra]().	

Task	Description	Skills required
Create an IAM role for the AWS Glue job.	Create a new AWS service role named `glue-cassandra-migration` with AWS Glue as a trusted entity. **Note** The `glue-cassandra-migration` should provide read and write access to the S3 bucket and Amazon Keyspaces. The S3 bucket contains the .jar files, configuration files for Amazon Keyspaces and Cassandra, and the intermediate Parquet files. For example, it contains the `AWSGlueServiceRole`, `AmazonS3FullAccess`, and `AmazonKeyspacesFullAccess` managed policies.	AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Download CQLReplicator in AWS CloudShell.</td><td>Download the project to your home folder by running the following command:<pre><code>git clone https://github.com/aws-samples/cql-replicator.git
cd cql-replicator/glue
# Only for AWS CloudShell, the bc package includes bc and dc. Bc is an arbitrary precision numeric processing arithmetic language
sudo yum install bc -y</code></pre></td><td></td></tr>
<tr><td>Modify the reference configuration files.</td><td>Copy <code>CassandraConnector.conf</code> and <code>KeyspacesConnector.conf</code> to the <code>../glue/conf</code> directory in the project folder.</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required

Initiate the migration process.	The following command initializes the CQLReplicator environment. Initializaition involves copying .jar artifacts, and creating an AWS Glue connector, an S3 bucket, an AWS Glue job, the `migration` keyspace, and the `ledger` table:
	`cd cql-replicator/glue/bin`
	`./cqlreplicator --state init --sg '"sg-1","sg-2"' \ --subnet "subnet-XXXXXXXXXXX" \ --az us-west-2a --region us-west-2 \ --glue-iam-role glue-cassandra-migration \ --landing-zone s3://cql-replicator-1234567890-us-west-2`
	The script includes the following parameters:
	• `--sg` – The security groups that allow access to the Cassandra cluster from AWS Glue and include the self-referencing inbound rule for all traffic
		AWS DevOps

Task	Description	Skills required
	• `--subnet` – The subnet to which the Cassandra cluster belongs • `--az` – The Availability Zone of the subnet • `--region` – The AWS Region where the Cassandra cluster is deployed • `--glue-iam-role` – The IAM role permissions that AWS Glue can assume when calling Amazon Keyspaces and Amazon S3 on your behalf • `--landing zone` – An optional parameter for reusing an S3 bucket (If you don't supply a value for the `--landing zone` parameter, the `init` process will try to create a new bucket to store the configuration files, .jar artifacts, and intermediate files.)	

Task	Description	Skills required
Validate the deployment.	After you run the previous command, the AWS account should contain the following: • The CQLReplicator AWS Glue job and the AWS Glue connector in AWS Glue • The S3 bucket that stores the artifacts • The target keyspace `migration` and the `ledger` table in Amazon Keyspaces	AWS DevOps

Run CQLReplicator

Task	Description	Skills required
Start the migration process.	To operate CQLReplicator on AWS Glue, you need to use the `--state run` command, followed by a series of parameters. The precise configuration of these parameters is primarily determined by your unique migration requirements. For example, these settings might vary if you choose to replicate time to live (TTL) values and updates, or you offload objects exceeding 1 MB to Amazon S3.	AWS DevOps

Task	Description	Skills required
	To replicate the workload from the Cassandra cluster to Amazon Keyspaces, run the following command: `./cqlreplicator --state run --tiles 8 \ --landing-zone s3://cql-replicator-1234567890-us-west-2 \ --region us-west-2 \ --src-keyspace source_keyspace \ --src-table source_table \ --trg-keyspace taget_keyspace \ --writetime-column column_name \ --trg-table target_table --inc-traffic` Your source keyspace and table are `source_keyspace.source_table` in the Cassandra cluster. Your target keyspace and table are `target_keyspace.target_table` in Amazon Keyspaces. The parameter `--inc-traffic`	

Task	Description	Skills required
	helps prevent incremental traffic from overloading the Cassandra cluster and Amazon Keyspaces with a high number of requests. To replicate updates, add `--writetime-column regular_column_name` to your command line. The regular column is going to be used as the source of the write timestamp.	

Monitor the migration process

Task	Description	Skills required
Validate migrated Cassandra rows during the historical migration phase.	To obtain the number of rows replicated during the backfilling phase, run the following command: `./cqlreplicator --state stats \ --landing-zone s3://cql-replicator-1234567890-us-west-2 \ --src-keyspace source_keyspace --src-table source_table --region us-west-2`	AWS DevOps

Stop the migration process

Task	Description	Skills required
Use the `cqlreplicator` command or the AWS Glue console.	To stop the migration process gracefully, run the following command: `./cqlreplicator --state request-stop --tiles 8 \ --landing-zone s3://cql-replicator-1234567890-us-west-2 \ --region us-west-2 \ --src-keyspace source_keyspace --src-table source_table` To stop the migration process immediately, use the AWS Glue console.	AWS DevOps

Clean up

Task	Description	Skills required
Delete the deployed resources .	The following command will delete the AWS Glue job, connector, S3 bucket, and Keyspaces table `ledger`: `./cqlreplicator --state cleanup --landing-zone`	AWS DevOps

Task	Description	Skills required
	`s3://cql-replicator-1234567890-us-west-2`	

Troubleshooting

Issue	Solution
AWS Glue jobs failed and returned an Out of Memory (OOM) error.	1. Change the worker type (scale up). For example, change `G0.25X` to `G.1X` or `G.1X` to `G.2X`. Alternatively, increase the number of DPUs per AWS Glue job (scale out) in CQLReplicator. 2. Start the migration process from the point where it was interrupted. To restart failed CQLReplicator jobs, rerun the `--state run` command with the same parameters.

Related resources

- CQLReplicator with AWS Glue README.MD
- AWS Glue documentation
- Amazon Keyspaces documentation
- Apache Cassandra

Additional information

Migration considerations

You can use AWS Glue to migrate your Cassandra workload to Amazon Keyspaces, while keeping your Cassandra source databases completely functional during the migration process. After the replication is complete, you can choose to cut over your applications to Amazon Keyspaces with minimal replication lag (less than minutes) between the Cassandra cluster and Amazon Keyspaces.

To maintain data consistency, you can also use a similar pipeline to replicate the data back to the Cassandra cluster from Amazon Keyspaces.

Write unit calculations

As an example, consider that you intend to write 500,000,000 with the row size 1 KiB during one hour. The total number of Amazon Keyspaces write units (WCUs) that you require is based on this calculation:

`(number of rows/60 mins 60s) 1 WCU per row = (500,000,000/(60*60s) * 1 WCU) = 69,444 WCUs required`

69,444 WCUs per second is the rate for 1 hour, but you could add some cushion for overhead. For example, `69,444 * 1.10 = 76,388` WCUs has 10 percent overhead.

Create a keyspace by using CQL

To create a keyspace by using CQL, run the following commands:

```
CREATE KEYSPACE target_keyspace WITH replication = {'class': 'SingleRegionStrategy'}
CREATE TABLE target_keyspace.target_table ( userid uuid, level text, gameid int,
 description text, nickname text, zip text, email text, updatetime text, PRIMARY KEY
 (userid, level, gameid) ) WITH default_time_to_live = 0 AND CUSTOM_PROPERTIES =
 {'capacity_mode':{ 'throughput_mode':'PROVISIONED', 'write_capacity_units':76388,
 'read_capacity_units':3612 }} AND CLUSTERING ORDER BY (level ASC, gameid ASC)
```

Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator

Tony Velcich, Amazon Web Services

Summary

This pattern describes the process for migrating Apache Hadoop data from a Hadoop Distributed File System (HDFS) to Amazon Simple Storage Service (Amazon S3). It uses WANdisco LiveData Migrator to automate the data migration process.

Prerequisites and limitations

Prerequisites

- Hadoop cluster edge node where LiveData Migrator will be installed. The node should meet the following requirements:
  - Minimum specification: 4 CPUs, 16 GB RAM, 100 GB storage.
  - 2 Gbps minimum network.
  - Port 8081 accessible on your edge node to access the WANdisco UI.
  - Java 1.8 64-bit.
  - Hadoop client libraries installed on the edge node.
  - Ability to authenticate as the HDFS superuser (for example, "hdfs").
  - If Kerberos is enabled on your Hadoop cluster, a valid keytab that contains a suitable principal for the HDFS superuser must be available on the edge node.
- An active AWS account with access to an S3 bucket.
- An AWS Direct Connect link established between your on-premises Hadoop cluster (specifically the edge node) and AWS.

Product versions

- LiveData Migrator 1.8.6
- WANdisco UI (OneUI) 5.8.0

Architecture

Source technology stack

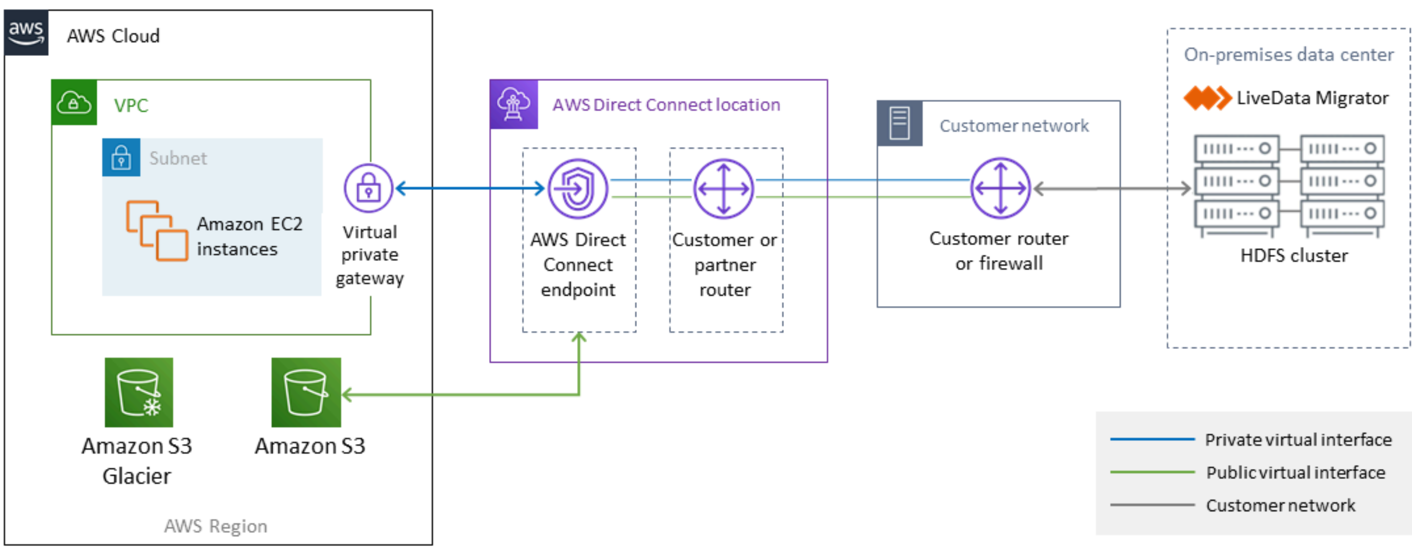
- On-premises Hadoop cluster

Target technology stack

- Amazon S3

Architecture

The following diagram shows the LiveData Migrator solution architecture.



The workflow consists of four primary components for data migration from on-premises HDFS to Amazon S3.

- LiveData Migrator – Automates the migration of data from HDFS to Amazon S3, and resides on an edge node of the Hadoop cluster.
- HDFS – A distributed file system that provides high-throughput access to application data.
- Amazon S3 – An object storage service that offers scalability, data availability, security, and performance.
- AWS Direct Connect – A service that establishes a dedicated network connection from your on-premises data centers to AWS.

Automation and scale

You will typically create multiple migrations so that you can select specific content from your source file system by path or directory. You can also migrate data to multiple, independent file systems at the same time by defining multiple migration resources.

Epics

Configure Amazon S3 storage in your AWS account

Task	Description	Skills required
Sign in to your AWS account.	Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.	AWS experience
Create an S3 bucket.	If you don't already have an existing S3 bucket to use as the target storage, choose the "Create bucket" option on the Amazon S3 console, and specify a bucket name, AWS Region, and bucket settings for block public access. AWS and WANdisco recommend that you enable the block public access options for the S3 bucket, and set up the bucket access and user permission policies to meet your organization's requirements. An AWS example is provided at https://docs.aws.amazon.com/AmazonS3/latest/dev/	AWS experience

Task	Description	Skills required
	example-walkthroughs-managing-access-example1.html.	

Install LiveData Migrator

Task	Description	Skills required
Download the LiveData Migrator installer.	Download the LiveData Migrator installer and upload it to the Hadoop edge node. You can download a free trial of LiveData Migrator at https://www2.wandisco.com/ldm-trial. You can also obtain access to LiveData Migrator from AWS Marketplace, at https://aws.amazon.com/marketplace/pp/B07B8SZND9.	Hadoop administrator, Application owner
Install LiveData Migrator.	Use the downloaded installer and install LiveData Migrator as the HDFS superuser on an edge node in your Hadoop cluster. See the "Additional information" section for the installation commands.	Hadoop administrator, Application owner
Check the status of LiveData Migrator and other services.	Check the status of LiveData Migrator, Hive migrator, and WANdisco UI by using the commands provided in the "Additional information" section.	Hadoop administrator, Application owner

Configure storage through the WANdisco UI

Task	Description	Skills required
Register your LiveData Migrator account.	Log in to the WANdisco UI through a web browser on port 8081 (on the Hadoop edge node) and provide your details for registration. For example, if you are running LiveData Migrator on a host named myldmhost.example.com, the URL would be: http://myldmhost.example.com:8081	Application owner
Configure your source HDFS storage.	Provide the configuration details needed for your source HDFS storage. This will include the "fs.defaultFS" value and a user-defined storage name. If Kerberos is enabled, provide the principal and keytab location for LiveData Migrator to use. If NameNode HA is enabled on the cluster, provide a path to the core-site.xml and hdfs-site.xml files on the edge node.	Hadoop administrator, Application owner
Configure your target Amazon S3 storage.	Add your target storage as the S3a type. Provide the user-defined storage name and the S3 bucket name. Enter "org.apache.hadoop.fs.s3a.SimpleAWSCredential	AWS, Application owner

Task	Description	Skills required
	sProvider" for the Credentials Provider option, and provide the AWS access and secret keys for the S3 bucket. Additional S3a propertie s will also be needed. For details, see the "S3a Propertie s" section in the LiveData Migrator documentation at https://docs.wandisco.com/ live-data-migrator/docs/ command-reference/#fil esystem-add-s3a.	

Prepare for the migration

Task	Description	Skills required
Add exclusions (if needed).	If you want to exclude specific datasets from migration , add exclusions for the source HDFS storage. These exclusions can be based on file size, file names (based on regex patterns), and modificat ion date.	Hadoop administrator, Application owner

Create and start the migration

Task	Description	Skills required
Create and configure the migration.	Create a migration in the dashboard of the WANdisco UI. Choose your source (HDFS)	Hadoop administrator, Application owner

Task	Description	Skills required
	and target (the S3 bucket). Add new exclusions that you have defined in the previous step. Select either the "Overwrite" or the "Skip if Size Match" option. Create the migration when all fields are complete.	
Start the migration.	On the dashboard, select the migration you created. Click to start the migration. You can also start a migration automatically by choosing the auto-start option when you create the migration.	Application owner

Manage bandwidth (optional)

Task	Description	Skills required
Set a network bandwidth limit between the source and target.	In the Storages list on the dashboard, select your source storage and select "Bandwidth Management" in the Grouping list. Clear the unlimited option, and define the maximum bandwidth limit and unit. Choose "Apply."	Application owner, Networking

Monitor and manage migrations

Task	Description	Skills required
View migration information using the WANdisco UI.	Use the WANdisco UI to view license, bandwidth, storage and migration informati on. The UI also provides a notification system so you can receive notifications about errors, warnings, or important milestones in your usage.	Hadoop administrator, Application owner
Stop, resume, and delete migrations.	You can stop a migration from transferring content to its target by placing it in the STOPPED state. Stopped migrations can be resumed. Migrations in the STOPPED state can also be deleted.	Hadoop administrator, Application owner

Related resources

- LiveData Migrator documentation
- LiveData Migrator in AWS Marketplace
- WANdisco LiveData Migrator demonstration (video)

Additional information

Installing LiveData Migrator

You can use the following commands to install LiveData Migrator, assuming that the installer is inside your working directory:

```
su – hdfs
chmod +x livedata-migrator.sh && sudo ./livedata-migrator.sh
```

Checking the status of LiveData Migrator and other services after installation

Use the following commands to check the status of LiveData Migrator, Hive migrator, and WANdisco UI:

```
service livedata-migrator status
service hivemigrator status
service livedata-ui status
```

Migrate Oracle Business Intelligence 12c to the AWS Cloud from on-premises servers

Lanre (Lan-Ray) showunmi and Patrick Huang, Amazon Web Services

Summary

This pattern shows how to migrate Oracle Business Intelligence Enterprise Edition 12c from on-premises servers to the AWS Cloud by using AWS CloudFormation. It also describes how you can use other AWS services to implement Oracle BI 12c components that deliver high availability, security, flexibility, and the ability to dynamically scale.

For a list of best practices related to migrating Oracle BI 12c to the AWS Cloud, see the **Additional information** section of this pattern.

Note

It's a best practice to run multiple test migrations before transferring your existing Oracle BI 12c data to the cloud. These tests help you fine tune your migration approach, identify and fix potential issues, and estimate downtime requirements more accurately.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Secure network connectivity between your on-premises servers and AWS through either AWS Virtual Private Network (AWS VPN) services or AWS Direct Connect
- Software licenses for your Oracle operating system, Oracle BI 12c, Oracle Database, Oracle WebLogic Server, and Oracle HTTP Server

Limitations

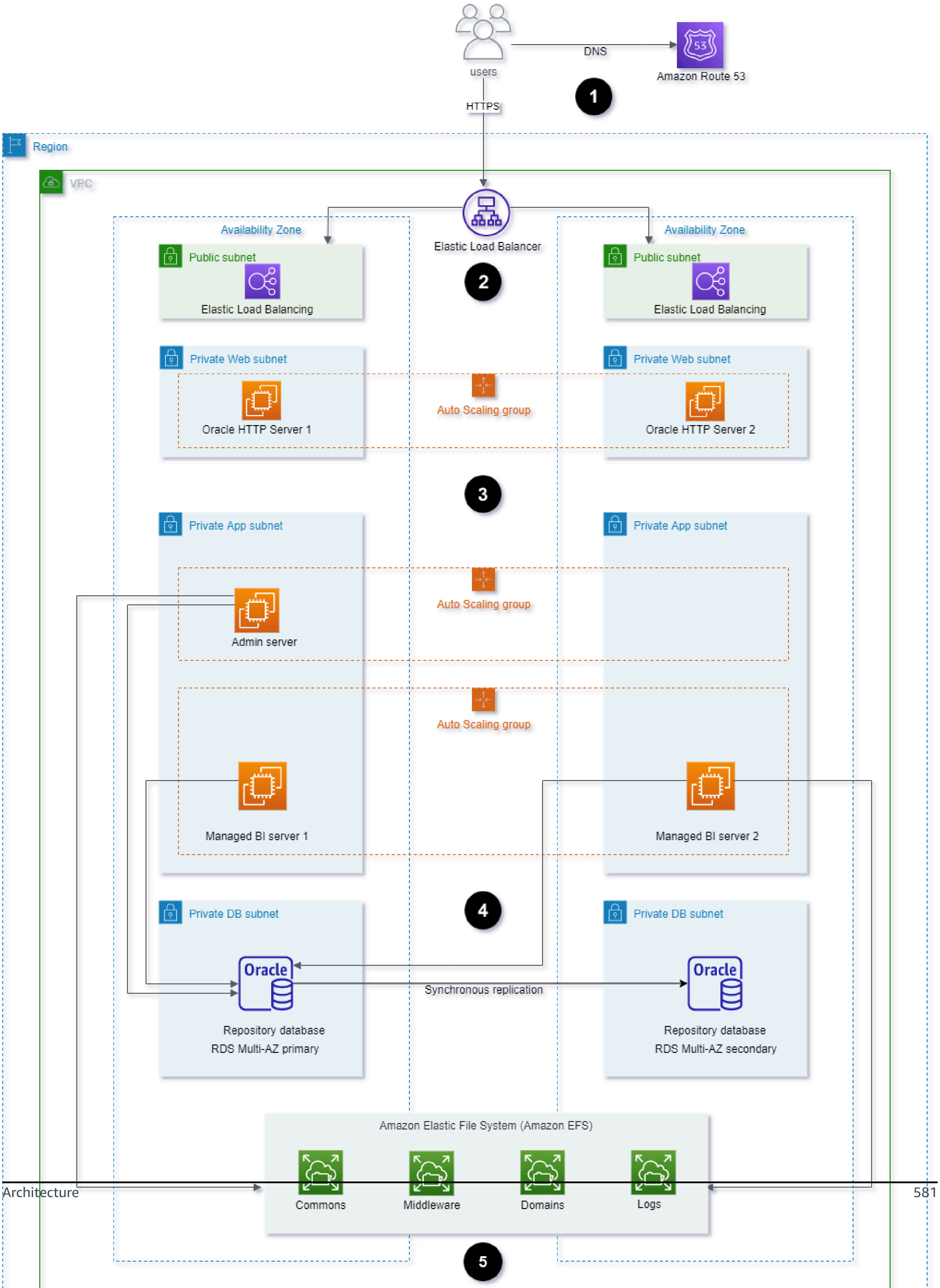
For information about storage size limits, see the Amazon Relational Database Service (Amazon RDS) for Oracle documentation.

Product versions

- Oracle Business Intelligence Enterprise Edition 12c
- Oracle WebLogic Server 12c
- Oracle HTTP Server 12c
- Oracle Database 12c (or newer)
- Oracle Java SE 8

Architecture

The following diagram shows an example architecture for running Oracle BI 12c components in the AWS Cloud:


users
DNS
Amazon Route 53
1
HTTPS
Region
VPC
Availability Zone
Elastic Load Balancer
Availability Zone
Public subnet
Elastic Load Balancing
2
Public subnet
Elastic Load Balancing
Private Web subnet
Oracle HTTP Server 1
Auto Scaling group
Private Web subnet
Oracle HTTP Server 2
3
Private App subnet
Private App subnet
Auto Scaling group
Admin server
Auto Scaling group
Managed BI server 1
Managed BI server 2
4
Private DB subnet
Private DB subnet
Oracle
Synchronous replication
Oracle
Repository database
RDS Multi-AZ primary
Repository database
RDS Multi-AZ secondary
Amazon Elastic File System (Amazon EFS)
Commons
Middleware
Domains
Logs
5

This diagram shows the following architecture:

1. Amazon Route 53 provides domain name service (DNS) configuration.
2. Elastic Load Balancing (ELB) distributes network traffic to improve the scalability and availability of the Oracle BI 12c components across multiple Availability Zones.
3. Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling groups host the Oracle HTTP Servers, Weblogic Admin server, and managed BI servers across multiple Availability Zones.
4. Amazon Relational Database Service (Amazon RDS) for Oracle database store BI Server metadata across multiple Availability Zones.
5. Amazon Elastic File System (Amazon EFS) is mounted to every Oracle BI 12c component for shared file storage.

Technology stack

- Amazon Elastic Block Store (Amazon EBS)
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic File System (Amazon EFS)
- Amazon RDS for Oracle
- AWS Certificate Manager (ACM)
- Elastic Load Balancing (ELB)
- Oracle BI 12c
- Oracle WebLogic Server 12c
- Oracle HTTP Server (OHS)

Tools

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Certificate Manager (ACM) helps you create, store, and renew public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications.
- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need, and quickly scale them up or down.
- Amazon EC2 Auto Scaling helps you maintain application availability and allows you to automatically add or remove Amazon EC2 instances according to conditions you define.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.
- Elastic Load Balancing distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.
- Oracle Data Pump helps you move data and metadata from one database to another at high speeds.
- Oracle Fusion Middleware is a suite of application development tools and integration solutions to identity management, collaboration, and business intelligence reporting.
- Oracle GoldenGate helps you design, run, orchestrate, and monitor your data replication and stream data processing solutions in the Oracle Cloud Infrastructure.
- Oracle WebLogic Scripting Tool (WLST) provides a command line interface that helps you horizontally scale out your WebLogic clusters.

Epics

Assess the source environment

Task	Description	Skills required
Gather software inventory information.	Identify versions and patch levels for each of your source technology stack's software	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator

Task	Description	Skills required
	components, including the following: • Oracle operating system • Oracle Database • Oracle BI 12c • Oracle WebLogic Server • Oracle HTTP Server • Java	
Gather compute and storage inventory information.	In your source environment, review current and historica l utilization metrics for the following: • CPU usage • Memory usage • Storage usage **Important** Make sure that you account for historical spikes in usage.	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator, System Administrator

Task	Description	Skills required
Gather information about the source environment's architecture and its requireme nts.	Obtain a full understanding of your source environment's architecture and its requireme nts, including knowledge of the following: • Oracle WebLogic Server domain configuration • Clustering • Load balancing • Connectivity • Availability • Disaster recovery requireme nts	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator
Identify Java Database Connectivity (JDBC) data sources.	Gather information about your source environment's JDBC data sources and drivers for each database engine that it uses.	Migration Architect, Applicati on Owner, Oracle BI Administr ator, Database Engineer or Administrator
Gather information about environment-specific settings.	Collect information about settings and configurations that are specific to your source environment, including the following: • Custom startup and shutdown scripts • Java and other environme nt variables • Certificates	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator

Task	Description	Skills required
Identify any dependencies on other applications.	Collect information about integrations in your source environment that create dependencies with other applications. **Important** Make sure that you identify any Lightweight Directory Access Protocol (LDAP) integrations and other networking requirements.	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator

Design your target environment

Task	Description	Skills required
Create a high-level design document.	Create a target architecture design document. Make sure that you use the information that you collected when assessing your source environment to inform the design document.	Solutions Architect, Application Architect, Database Engineer, Migration Architect
Obtain approval for the design document.	Review the design document with stakeholders and obtain the required approvals.	Application or Service Owner, Solutions Architect, Application Architect

Deploy the infrastructure

Task	Description	Skills required
Prepare the infrastructure code in CloudFormation.	Create CloudFormation templates to provision your Oracle BI 12c infrastructure in the AWS Cloud. For more information, see [Working with AWS CloudFormation templates]() in the *AWS CloudFormation User Guide*. **Note** It's a best practice to create modular CloudFormation templates for each Oracle BI 12c tier, rather than one large template for all of your resources. For more information about CloudFormation best practices, see [8 best practices when automating your deployments with AWS CloudFormation]() on the AWS Blog.	Cloud Infrastructure Achitect, Solutions Architect, Application Architect
Download the required software.	DDownload the following software along with the	Migration Architect, Database Engineer, Application Architect

Task	Description	Skills required
	required versions and patches from the Oracle website: • Java JDK8 • Oracle WebLogic Server 12c • Oracle BI 12c	
Prepare the installation scripts.	Create software installation scripts that run a silent install. These scripts simplify the deployment automation. For more information, see OBIEE 12c: How to Perform Silent Installation? on the Oracle Support site. You need an Oracle Support account to view the documentation.	Migration Architect, Database Engineer, Application Architect

Task	Description	Skills required
Create an Amazon EBS-backed Linux AMI for your web and application tiers.	1. Deploy and configure Amazon EC2 instances for your web and application tiers. Make sure that the instances meet the prerequisites for running the following: • Oracle operating system environment setup • Oracle operating system user account set up • Java software installation 2. Create Amazon Machine Images (AMIs) of the instances and save copies for future use. For instructions, see Create an Amazon EBS-backed Linux AMI in the *Amazon EC2 User Guide for Linux Instances*.	Migration Architect, Database Engineer, Application Architect
Launch your AWS infrastructure by using CloudFormation.	Deploy your Oracle BI 12c web and application tiers in modules by using the CloudFormation templates that you created. For instructions, see Getting started with AWS CloudFormation in the *AWS CloudFormation User Guide*.	Cloud Infrastructure Architect, Solutions Architect, Application Architect

Migrate Oracle BI 12c to AWS by using a fresh installation

Task	Description	Skills required
Stage the required software.	Stage the required software in a location that is accessible to the Amazon EC2 instances. For example, you could stage the software in Amazon S3 or another Amazon EC2 instance that would be accessible to your web and application servers.	Migration Architect, Oracle BI Architect, Cloud Infrastructure Achitect, Solutions Architect, Application Architect
Prepare your repository database for Oracle BI 12c installation.	Create Oracle BI 12c schemas by running the Oracle Repository Creation Utility (RCU) against a new Amazon RDS for Oracle database instance.	Cloud Infrastructure Achitect, Solutions Architect, Applicati on Architect, Migration Architect, Oracle BI Architect
Install Oracle Fusion Middleware 12c and Oracle BI 12c.	1. Starting with one Amazon EC2 instance, install Oracle Fusion Middleware 12c infrastructure and OBIEE 12c. For more informati on, see the following sections of the *Oracle Fusion Middleware Enterpris e Deployment Guide for Oracle Business Intellige nce*: • Starting the infrastru cture installer on BIHOST1 • Installing Oracle Business Intelligence	Migration Architect, Oracle BI Architect

Task	Description	Skills required
	[in preparation for an enterprise deployment](#) **Note** Use Amazon EFS to host directories that will be shared among Oracle BI 12c cluster nodes. 2. Apply any required patches to the installation. 3. Create AMIs of the instances and save copies for future use.	
Configure your Oracle WebLogic Server domain for Oracle BI 12c.	Configure your Oracle BI 12c domain as a non-clustered deployment. For more information, see [Configuring the BI Domain](#) in the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Business Intellige nce*.	Migration Architect, Oracle BI Architect

Task	Description	Skills required
Perform horizontal scale out of the Oracle BI 12c.	Horizontally scale out the single node to the desired number of nodes. For more information, see Scaling out Oracle Business Intelligence in the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Business Intelligence*.	Migration Architect, Oracle BI Architect
Install Oracle HTTP Server 12c.	1. Install Oracle HTTP Server 12c on the Oracle web tier Amazon EC2 instances. For instructions see Install Oracle HTTP Server 12c in *Install and configure Oracle HTTP Server for Oracle Access Management 12c*. 2. Apply any required patches to the installation. 3. Create AMIs of the instances and save copies for future use.	Migration Architect, Oracle BI Architect
Configure load balancers for SSL termination.	1. Create or import SSL certificates in ACM. 2. Associate the SSL certificates with ELB.	Cloud Infrastructure Architect , Migration Architect

Task	Description	Skills required
Migrate business intelligence metadata artifacts to AWS.	1. Export Oracle Business Intelligence Application Archive (BAR) files from the on-premises Oracle BI 12c installation. To export the BAR files, use the WebLogic Scripting Tool (WLST) to run the exportServiceInstance command. 2. Import the on-premises BAR files into the AWS Oracle BI 12c installation. To import the BAR files, run the **importServiceInstance** WLST command.	Migration Architect, Oracle BI Architect

Task	Description	Skills required
Perform post-migration tasks.	After importing the BAR files, do the following: • Configure any additional JDBC data sources. • Install drivers for other data sources like PostgreSQL, or Amazon Redshift. • Configure Oracle LDAP, SSL, single sign-on (SSO), and WebLogic security store. • Configure AWS Identity and Access Management (IAM) policies. • Activate usage tracking. • Set up integrations to other systems. • Migrate any custom scripts.	Migration Architect, Oracle BI Architect

Test the new environment

Task	Description	Skills required
Test the new Oracle BI 12c environment.	Conduct end-to-end testing on the new Oracle BI 12c environment. Use automation as much as possible. Example of testing activities include the following: • Validating dashboards, reports, and URLs	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator

Task	Description	Skills required
	• User acceptance testing (UAT) • Operational acceptance testing (OAT) **Note** Conduct additional testing and validation as required.	

Cut over to the new environment

Task	Description	Skills required
Disconnect traffic to the on-premises Oracle BI 12c environment.	At the appointed cutover window, stop all traffic to the on-premises Oracle BI 12c environment.	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator
Resynchronize the new Oracle BI 12c repository database with the source database.	Resynchronize the Amazon RDS Oracle Oracle BI 12c repository database with the on-premises database. To synchronize the databases , you can either use an Oracle Data Pump refresh or an AWS DMS change data capture (CDC).	Oracle BI Administrator, Database Engineer/Administr ator
Switch your Oracle BI 12c URLs to point to the new AWS environment.	Update the Oracle BI 12c URLs on your internal DNS	Migration Architect, Solutions Architect, Application Owner, Oracle BI Administrator

Task	Description	Skills required
	servers so that they point to the new AWS installation.	
Monitor the new environme nt.	Monitor the new Oracle BI 12c environment by using any of the following tools: • Amazon CloudWatch • Amazon RDS Performance Insights • Oracle Enterprise Manager	Oracle BI Administrator, Database Engineer/Administr ator, Application Administr ator
Get sign-off on the project.	Review the testing results with stakeholders and obtain the required approvals to wrap up the migration.	Application Owner, Service Owner, Cloud Infrastructure Architect, Migration Architect, Oracle BI Architect

Related resources

- Using the Oracle Repository Creation Utility on RDS for Oracle (*Amazon RDS User Guide*)
- Oracle on Amazon RDS (*Amazon RDS User Guide*)
- Oracle WebLogic Server 12c on AWS (AWS whitepaper)
- Deploying Oracle Business Intelligence for high availability (Oracle Help Center)
- Oracle Business Intelligence Application Archive (BAR) Files (Oracle Help Center)
- How to migrate OBI 12c between environments (Oracle Support)

Additional information

The following is a list of best practices related to migrating Oracle BI 12c to the AWS Cloud.

Repository databases

It's a best practice to host Oracle BI 12c database schemas on an Amazon RDS for Oracle instance. This instance type provides cost-efficient and resizable capacity while automating administration tasks, such as hardware provisioning, database setup, patching, and backups.

For more information, see Using the Oracle Repository Creation Utility on RDS for Oracle in the *Amazon RDS User Guide*.

Web and application tiers

Memory optimized Amazon EC2 instances are often well suited for Oracle BI 12c servers. Whatever instance type you choose, make sure that the instances that you provision meet your system's memory usage requirements. Also, make sure that you configure a sufficient WebLogic Java Virtual Machine (JVM) heap size based on your Amazon EC2 instance's available memory.

Local storage

I/O plays an important part in the overall performance of your Oracle BI 12c application. Amazon Elastic Block Store (Amazon EBS) offers different storage classes that are optimized for different workload patterns. Make sure that you choose an Amazon EBS volume type that fits your use case.

For more information about EBS volume types, see Amazon EBS features in the Amazon EBS documentation.

Shared storage

A clustered Oracle BI 12c domain requires shared storage for the following resources:

- Configuration files
- Oracle BI 12c singleton data directory (SDD)
- Oracle global cache
- Oracle BI Scheduler scripts
- Oracle WebLogic Server binaries

You can meet this shared storage requirement by using Amazon EFS, which provides a scalable, fully managed elastic Network File System (NFS) file system.

Fine tuning shared storage performance

Amazon EFS has two throughput modes: **Provisioned** and **Bursting**. The service also has two performance modes: **General Purpose** and **Max I/O**.

To fine tune performance, start by testing your workloads in the **General Purpose** performance mode and **Provisioned** throughput mode. Doing these tests will help you determine if those baseline modes are sufficient to meet your desired service levels.

For more information, see Amazon EFS performance in the *Amazon EFS User Guide*.

Availability and disaster recovery

It's a best practice to deploy Oracle BI 12c components across multiple Availability Zones to protect those resources in the event of an Availability Zone failure. The following is a list of availability and disaster recovery best practices for specific Oracle BI 12c resources hosted in the AWS Cloud:

- **Oracle BI 12c repository databases**: Deploy a multi-AZ Amazon RDS database instance to your Oracle BI 12crepository database. In a multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different AZ. Running an Oracle BI 12c repository database instance across Availability Zones can enhance availability during planned system maintenance and help protect your databases against instance and Availability Zone failures.
- **Oracle BI 12c Managed Servers**: To achieve fault tolerance, it's a best practice to deploy Oracle BI 12c system components on Managed Servers in an Amazon EC2 Auto Scaling Group configured to span multiple Availability Zones. Auto Scaling replaces faulty instances based on Amazon EC2 health checks. In the event of an Availability Zone failure, Oracle HTTP Servers continue to direct traffic to Managed Servers in the functioning Availability Zone. Then, Auto Scaling launches instances to keep up with your host count requirements. Activating HTTP session state replication is recommended to help make sure that there's a smooth failover of the existing sessions to the functioning Managed Servers.
- **Oracle BI 12c Administration Servers:** To make sure that your Administration Server has high availability, host it in an Amazon EC2 Auto Scaling group configured to span multiple Availability Zones. Then, set the minimum and maximum size of the group set to **1**. If an Availability Zone failure occurs, Amazon EC2 Auto Scaling starts up a replacement Administration Server in an alternate Availability Zone. To recover any failed underlying hosts within the same Availability Zone, you can activate Amazon EC2 Auto Recovery.
- **Oracle Web Tier servers:** It's a best practice to associate your Oracle HTTP Server with your Oracle WebLogic Server domain. For high availability, deploy your Oracle HTTP Server in an Amazon EC2 Auto Scaling group configured to aspen multiple Availability Zoness. Then, place the server behind an ELB elastic load balancer. To provide additional protection against host failure, you can activate Amazon EC2 Auto Recovery.

Scalability

The elasticity of the AWS Cloud helps you scale applications either horizontally or vertically in response to workload requirements.

Vertical scaling

To vertically scale your application, you can change the size and type of the Amazon EC2 instances that are running your Oracle BI 12c components. You don't need to over-provision instances at the start of your deployment and incur unnecessary cost.

Horizontal scaling

Amazon EC2 Auto Scaling helps you horizontally scale your application by automatically adding or removing Managed Servers based on workload requirements.

> **Note**
>
> Horizontal scaling with Amazon EC2 Auto Scaling requires scripting skills and thorough testing to implement.

Backup and recovery

The following is a list of backup and recovery best practices for specific Oracle BI 12c resources hosted in the AWS Cloud:

- **Oracle Business Intelligence metadata repositories:** Amazon RDS automatically creates and saves backups of your database instances. These backups are retained for a period of time that you specify. Make sure that you configure your Amazon RDS backup duration and retention settings based on your data protection requirements. For more information, see Amazon RDS backup and restore.
- **Managed Servers, Administration Servers, and Web Tier servers:** Make sure that you configure Amazon EBS snapshots based on your data protection and retention requirements.
- **Shared storage:** You can manage backup and recovery for files stored in Amazon EFS by using AWS Backup. The AWS Backup service can also be deployed to centrally manage backup and recovery of other services, including Amazon EC2, Amazon EBS, and Amazon RDS. For more information, see What is AWS Backup? In the *AWS Backup Developer Guide*.

Security and compliance

The following is a list of security best practices and AWS services that can help you protect your Oracle BI 12c applications in the AWS Cloud:

- **Encryption at rest:** Amazon RDS, Amazon EFS, and Amazon EBS all support industry standard encryption algorithms. You can use AWS Key Management Service (AWS KMS) to create and manage cryptographic keys and control their use across AWS services and in your applications. You can also configure Oracle Transparent Data Encryption (TDE) on the Amazon RDS for Oracle database instance that's hosting your Oracle BI 12c repository database.
- **Encryption in transit:** It's a best practice to activate either SSL or TLS protocols to protect data in transit between the various layers of your Oracle BI 12c installation. You can use AWS Certificate Manager (ACM) to provision, manage, and deploy public and private SSL and TLS certificates for your Oracle BI 12c resources.
- **Network security**: Make sure that you deploy your Oracle BI 12c resources in an Amazon VPC that has the appropriate access controls configured for your use case. Configure your security groups to filter inbound and outbound traffic from the Amazon EC2 instances that are running your installation. Also, make sure that you configure Network Access Control Lists (NACLs) that allow or deny traffic based on defined rules.
- **Monitoring and logging**: You can use AWS CloudTrail to track API calls to your AWS infrastructure, including your Oracle BI 12c resources. This functionality is useful when tracking changes to infrastructure or when conducting a security analysis. You can also use Amazon CloudWatch to view operational data that can provide you with actionable insight into the performance and health of your Oracle BI 12c application. You can configure alarms and take automated actions based on those alarms, too. Amazon RDS provides additional monitoring tools, including Enhanced Monitoring and Performance Insights.

Migrate an ELK Stack to Elastic Cloud on AWS

Battulga Purevragchaa and Antony Prasad Thevaraj, Amazon Web Services

uday reddy, None

Summary

Elastic has provided services for many years, with their users and customers typically managing Elastic themselves on premises. Elastic Cloud, the managed Elasticsearch service, provides a way to consume the Elastic Stack (ELK Stack) and solutions for enterprise search, observability, and security. You can access Elastic solutions with apps such as Logs, Metrics, APM (application performance monitoring), and SIEM (security information and event management). You can use integrated features such as machine learning, index lifecycle management, Kibana Lens (for drag-and drop visualizations).

When you move from self-managed Elasticsearch to Elastic Cloud, the Elasticsearch service takes care of the following:

- Provisioning and managing the underlying infrastructure
- Creating and managing Elasticsearch clusters
- Scaling clusters up and down
- Upgrades, patching, and taking snapshots

This gives you more time to focus on solving other challenges.

This pattern defines how to migrate on-premises Elasticsearch 7.13 to Elasticsearch on Elastic Cloud on Amazon Web Services (AWS). Other versions might require slight modifications to the processes described in thie pattern. For more information, contact your Elastic representative.

Prerequisites and limitations

Prerequisites

- An active AWS account with access to Amazon Simple Storage Service (Amazon S3) for snapshots
- A secure, sufficiently high-bandwidth private link for copying snapshot data files to Amazon S3
- Amazon S3 Transfer Acceleration

- Elastic Snapshot policies to ensure that data ingestion is archived regularly, either to a sufficiently large local data store or to remote storage (Amazon S3)

You must understand how large your snapshots and the lifecyle policies for accompanying indexes are on premises before initiating your migration. For more information, contact Elastic.

Roles and skills

The migration process also requires the roles and expertise described in the following table.

Role	Expertise	Responsibilities
App support	Familiarity with Elastic Cloud and Elastic on premises	All Elastic related tasks
Systems administrator or DBA	In-depth knowledge of the on-premises Elastic environment and its configura tion	The ability to provision storage, install and use the AWS Command Line Interface (AWS CLI), and identify all data sources feeding Elastic on premises
Network administrator	Knowledge of on-premises to AWS network connectivity, security, and performance	Establishment of network links from on premises to Amazon S3, with an understanding of connectivity bandwidth

Limitations

- Elasticsearch on Elastic Cloud is available only in supported AWS Regions (September 2021).

Product versions

- Elasticsearch 7.13

Architecture

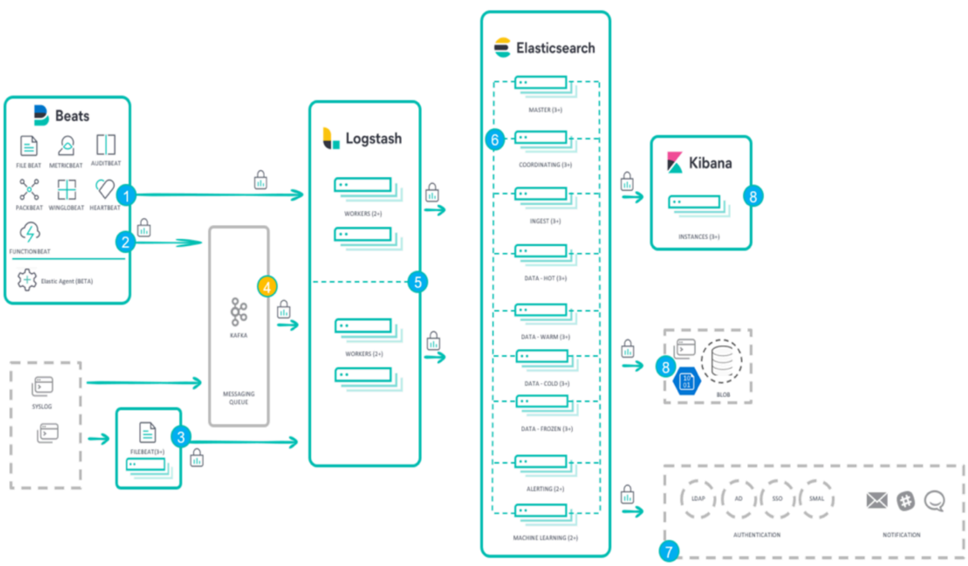
Source technology stack

On-premises Elasticsearch 7.13 or later:

- Cluster snapshots
- Index snapshots
- Beats configuration

Source technology architecture

The following diagram shows a typical on-premises architecture with different ingestion methods, node types, and Kibana. The different node types reflect the Elasticsearch cluster, authentication, and visualization roles.



1. Ingestion from Beats to Logstash
2. Ingestion from Beats to Apache Kafka messaging queue
3. Ingestion from Filebeat to Logstash
4. Ingestion from Apache Kafka messaging queue to Logstash
5. Ingestion from Logstash to an Elasticsearch cluster
6. Elasticsearch cluster

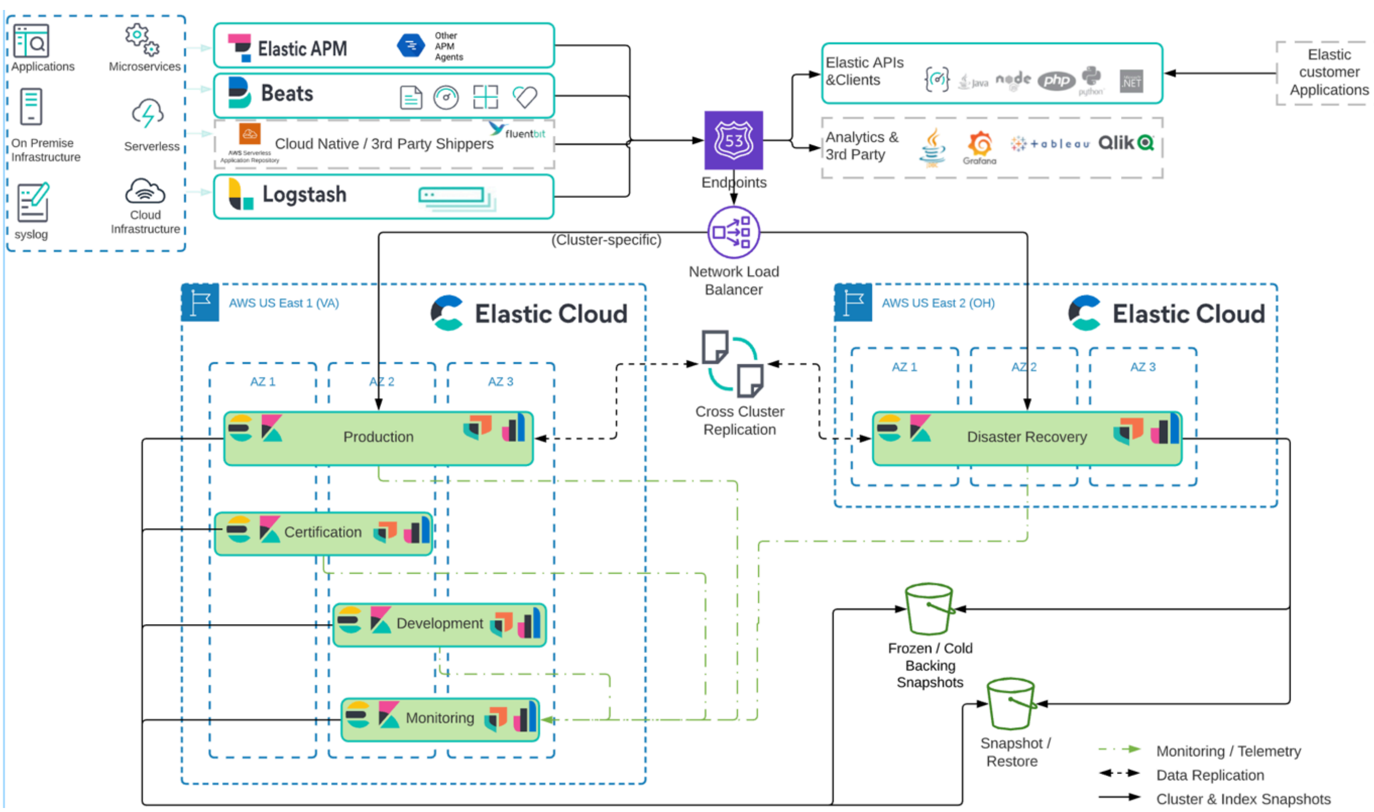
7. Authentication and notification node
8. Kibana and blob nodes

Target technology stack

Elastic Cloud is deployed to your software as a service (SaaS) account in multiple AWS Regions with cross-cluster replication.

- Cluster snapshots
- Index snapshots
- Beats configurations
- Elastic Cloud
- Network Load Balancer
- Amazon Route 53
- Amazon S3

Target architecture



The managed Elastic Cloud infrastructure is:

- Highly available, being present in multiple Availability Zones and multiple AWS Regions.
- Region failure tolerant because data (indexes and snapshots) is replicated using Elastic Cloud cross-cluster replication (CCR)
- Archival, because snapshots are archived in Amazon S3
- Network partition tolerant through a combination of Network Load Balancers and Route 53
- Data ingestion originating from (but not limited to) Elastic APM, Beats, Logstash

High-level migration steps

Elastic has developed its own prescriptive methodology for migrating on-premises Elastic Cluster to Elastic Cloud. The Elastic methodology is directly aligned and complementary to the AWS migration guidance and best practices, including Well-Architected Framework and AWS Migration Acceleration Program (MAP). Typically, the three AWS migration phases are the following:

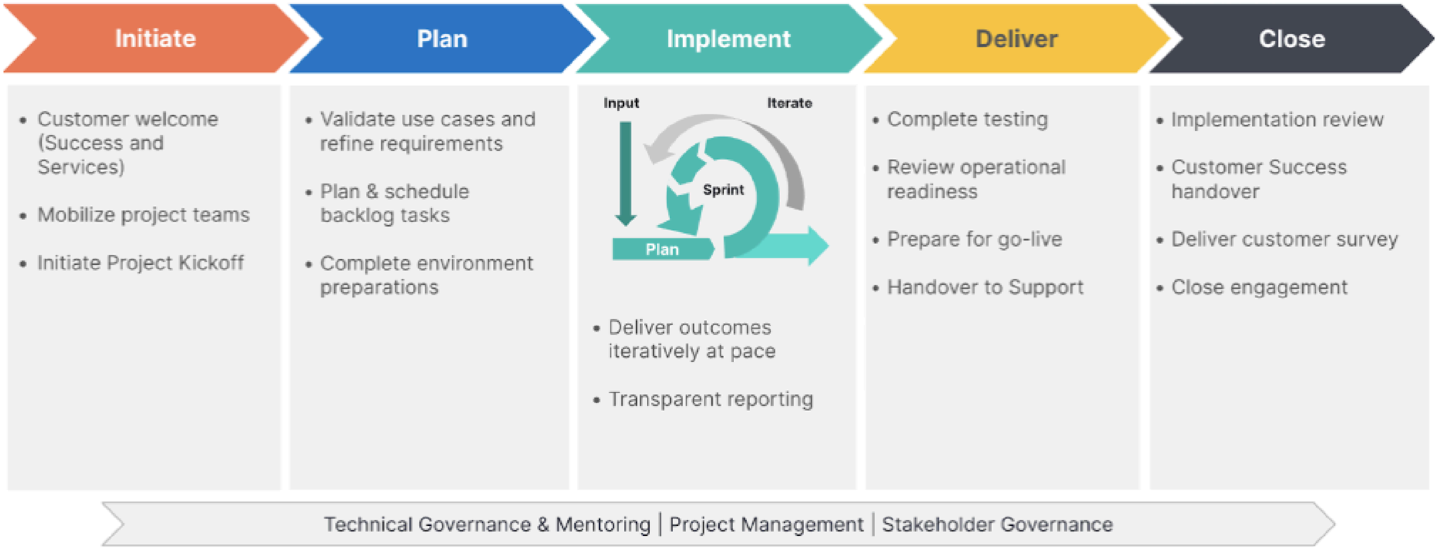
- Assess
- Mobilize
- Migrate and modernize

Elastic follows similar migration phases with complementary terminology:

- Initiate
- Plan
- Implement
- Deliver
- Close

Elastic uses the Elastic Implementation Methodology to facilitate the delivery of project outcomes. This is inclusive by design to ensure that the Elastic, consulting teams, and customer teams work together with clarity to jointly deliver intended outcomes.

The Elastic methodology combines traditional waterfall phasing with Scrum within the implementation phase. Configurations of technical requirements are delivered iteratively in a collaborative manner while minimizing risk.



Tools

AWS services

- Amazon Route 53 – Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can use Route 53 to perform three main functions in any combination: domain registration, DNS routing, and health checking.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. This pattern uses an S3 bucket and Amazon S3 Transfer Acceleration.
- Elastic Load Balancing – Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones.

Other tools

- Beats – Beats ship data from Logstash or Elasticsearch
- Elastic Cloud – Elastic Cloud is a managed service for hosting Elasticsearch.
- Elasticsearch – Elasticsearch is a search and analytics engine that uses the Elastic Stack to centrally store your data for search and analytics that scale. This pattern also uses snapshot creation and cross-cluster replication.
- Logstash – Logstash is a server-side data processing pipeline that ingests data from multiple sources, transforms it, and then sends it to your data storage.

Epics

Prepare the migration

Task	Description	Skills required
Identify servers running the on-premises Elastic solution.	Confirm that Elastic migration is supported.	App owner
Understand the on-premises server configuration.	To understand the server configuration needed to drive workloads successfu lly on premises, find the server hardware footprint, network configuration, and storage characteristics that are currently in use	App Support
Gather user and app account information.	Identify the user names and app names that are used by the on-premises Elastic environment.	Systems administrator, App support
Document Beats and data shipper configuration.	To document the configura tions, look at existing data sources and sinks. For more information, see the Elastic documentation.	App support
Determine the velocity and volume of data.	Establish a baseline for how much data the cluster is handling.	Systems administrator, App support
Document RPO and RTO scenarios.	Document recovery point objective (RPO) and recovery time objective (RTO) scenarios in terms of outages and	App owner, Systems administr ator, App support

Task	Description	Skills required
	service level agreements (SLAs).	
Determine the optimal snapshot lifecycle settings.	Define how often data needs to be secured by using Elastic snapshots *during and after* the migration.	App owner, Systems administr ator, App support
Define post-migration performance expectations.	Generate metrics on current and expected screen refresh, query runtimes, and user interface behaviors.	Systems administrator, App support
Document internet access transport, bandwidth, and availability requirements.	Ascertain speed, latency, and resiliency of internet connections for copying snapshots to Amazon S3.	Network administrator
Document current costs of on-premises runtime for Elastic.	Ensure that the sizing of the AWS targeted environment is designed to be both high performing and cost effective.	DBA, Systems administrator, App support
Identify the authentication and authorization needs.	The Elastic Stack security features provide built-in realms such as Lightweight Directory Access Protocol (LDAP), Security Assertion Markup Language (SAML), and OpenID Connect (OIDC).	DBA, Systems administrator, App support
Understand the specific regulatory requirements based on the geographic location.	Ensure that data is exported and encrypted according to your requirements and to any relevant national requireme nts.	DBA, Systems administrator, App support

Implement the migration

Task	Description	Skills required
Prepare the staging area on Amazon S3.	To receive snapshots on Amazon S3, create an S3 bucket and a temporary AWS Identity and Access Management (IAM) role with full access to your newly created bucket. For more information, see Creating a role to delegate permissions to an IAM user. Use the AWS Security Token Service to request temporary security credentials. Keep the access key ID, secret access key, and session token secured. Enable Amazon S3 Transfer Acceleration on the bucket.	AWS administrator
Install AWS CLI and the Amazon S3 plugin on premises.	On each Elasticsearch node, run the following command. `sudo bin/elasticsearch-plugin install repository-s3` Then reboot the node.	AWS administrator
Configure Amazon S3 client access.	Add the keys created previously by running the following commands.	AWS administrator

Task	Description	Skills required

	`elasticsearch-keystore add s3.client.default.access_key`
	`elasticsearch-keystore add s3.client.default.secret_key`
	`elasticsearch-keystore add s3.client.default.session_token`	

Register a snapshot repository for Elastic data	Use the Kibana Dev Tools to tell the on-premises local cluster which remote S3 bucket to write to.	AWS administrator
Configure snapshot policy.	To configure snapshot lifecycle management, on the Kibana **Policies** tab, choose **SLM policy**, and define which times, data streams, or indexes should be included, and what names to use. Configure a policy that takes frequent snapshots. Snapshots are incremental and make efficient use of storage. Match your readiness assessment decision. A policy can also specify a retention policy and automatically delete snapshots when they are no longer needed.	App support

Task	Description	Skills required
Verify that snapshots work.	In Kibana Dev Tools, run the following command. `GET _snapshot/<your_repo_name>/_all`	AWS administrator, App support,
Deploy a new cluster on Elastic Cloud.	Log in to Elastic, and choose a cluster for "observability, search or security" derived from your business findings in the readiness assessment.	AWS administrator, App support
Set up cluster key store access.	The new cluster needs access to the S3 bucket that will store the snapshots. On the Elasticsearch Service Console, choose **Security**, and enter the access and secret IAM keys that you created earlier.	AWS administrator

Task	Description	Skills required
Configure the Elastic Cloud hosted cluster to access Amazon S3.	Set up new cluster access to the previously created snapshot repository in Amazon S3. Using Kibana, do the following: 1. Choose **Stack Managemen t**, **Snapshot Settings**, **RegisterRepo**. 2. In the **Alias** field, enter the name of the repository. 3. For **S3 Client name**, choose **secondary**. 4. Add the S3 bucket that you created earlier to the repository. 5. Choose **Compress snapshot**. 6. For the **Encryption** settings, keep the default values.	AWS administrator, App Support
Verify the new Amazon S3 repository.	Ensure that you can access your new repository hosted in the Elastic Cloud cluster.	AWS administrator

Task	Description	Skills required

Initilaize the Elasticsearch service cluster.	On the Elasticsearch Service Console, initialize the Elasticsearch service cluster from the S3 snapshot.
	
	Run the following commands as POST.
	
	`*/_close?expand_wildcards=all`
	
	`/_snapshot/<your-repo-name>/ <your-snapshot-name>/_restore`
	
	`*/_open?expand_wildcards=all`
		App Support

Complete the migration

Task	Description	Skills required
Verify that the snapshot restore was successful.	Using Kibana Dev Tools, run the following command. `GET _cat/indices`	App support
Redploy ingestion services.	Connect the endpoints for Beats and Logstash to the new Elasticsearch service endpoint.	App support

Test the cluster environment and clean up

Task	Description	Skills required
Validate the cluster environment.	After the on-premises Elastic cluster environment is migrated to AWS, you can connect to it and use your own user acceptance testing (UAT) tools to validate the new environment.	App support
Clean-up the resources.	After you validate that the cluster migrated successfu lly, remove the S3 bucket and the IAM role used for the migration.	AWS administrator

Related resources

Elastic references

- Elastic Cloud
- Managed Elasticsearch and Kibana on AWS
- Elastic enterprise search
- Elastic integrations
- Elastic observability
- Elastic security
- Beats
- Elastic APM
- Migrate to index lifecycle management
- Elastic subscriptions
- Contact Elastic

Elastic blog posts

- How to migrate from self-managed Elasticsearch to Elastic Cloud on AWS (blog post)
- Migrating to Elastic Cloud (blog post)

Elastic documentation

- Tutorial: Automate backups with SLM
- ILM: Manage the index lifecycle
- Logstash
- Cross-cluster replication (CCR)
- Ingest pipelines
- Run Elasticsearch API requests
- Snapshot retention

Elastic video and webinar

- Elastic cloud migration
- Elastic Cloud: Why are customers migrating (webinar)

AWS references

- Elastic Cloud on AWS Marketplace
- AWS Command Line Interface
- AWS Direct Connect
- AWS Migration Acceleration Program
- Network Load Balancers
- Regions and Availability Zones
- Amazon Route 53
- Amazon Simple Storage Service
- Amazon S3 Transfer Acceleration
- VPN connections
- Well-Architected Framework

Additional information

If you're planning to migrate complex workloads, engage Elastic Consulting Services. If you have basic questions related to configurations and services, contact the Elastic Support team.

Migrate data to the AWS Cloud by using Starburst

Antony Prasad Thevaraj and Suresh Veeragoni, Amazon Web Services

Shaun Van Staden, None

Summary

Starburst helps accelerate your data migration journey to Amazon Web Services (AWS) by providing an enterprise query engine that brings existing data sources together in a single access point. You can run analytics across multiple data sources to get valuable insights, before finalizing any migration plans. Without disrupting business-as-usual analytics, you can migrate the data by using the Starburst engine or a dedicated extract, transform, and load (ETL) application.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC)
- An Amazon Elastic Kubernetes Service (Amazon EKS) cluster
- An Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling group
- A list of current system workloads that need to be migrated
- Network connectivity from AWS to your on-premises environment

Architecture

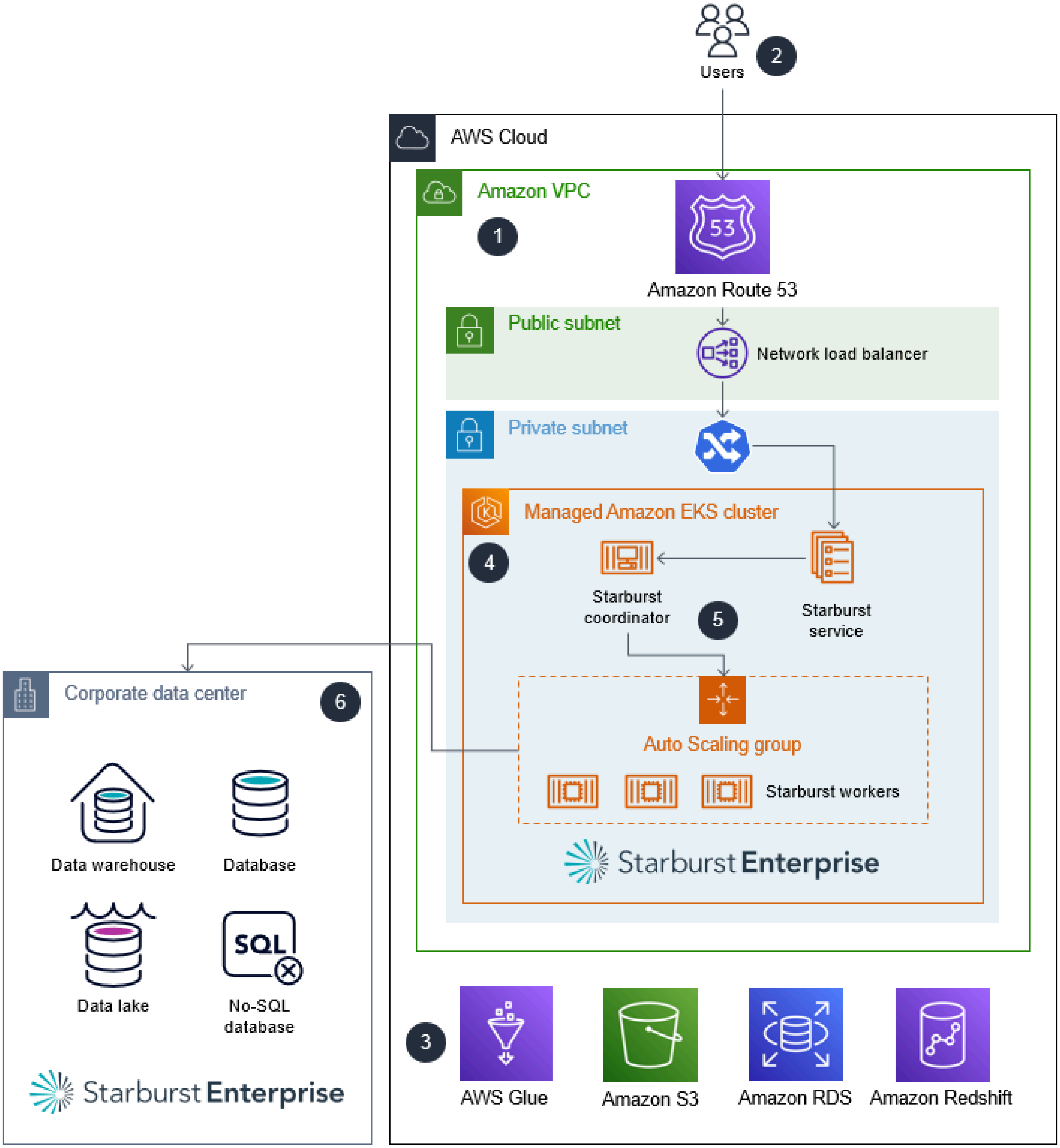
Reference architecture

The following high-level architecture diagram shows the typical deployment of Starburst Enterprise in the AWS Cloud:

1. The Starburst Enterprise cluster runs inside your AWS account.
2. A user authenticates by using Lightweight Directory Access Protocol (LDAP) or Open Authorization (OAuth) and interacts directly with the Starburst cluster.
3. Starburst can connect to several AWS data sources, such as AWS Glue, Amazon Simple Storage Service (Amazon S3), Amazon Relational Database Service (Amazon RDS), and Amazon Redshift.

Starburst provides federated query capabilities across data sources in the AWS Cloud, on premises, or in other cloud environments.

4. You launch Starburst Enterprise in an Amazon EKS cluster by using Helm charts.
5. Starburst Enterprise uses Amazon EC2 Auto Scaling groups and Amazon EC2 Spot Instances to optimize infrastructure.
6. Starburst Enterprise connects directly to your existing on-premises data sources to read data real-time. In addition, if you have an existing Starburst Enterprise deployment in this environment, you can directly connect your new Starburst cluster in the AWS Cloud to this existing cluster.

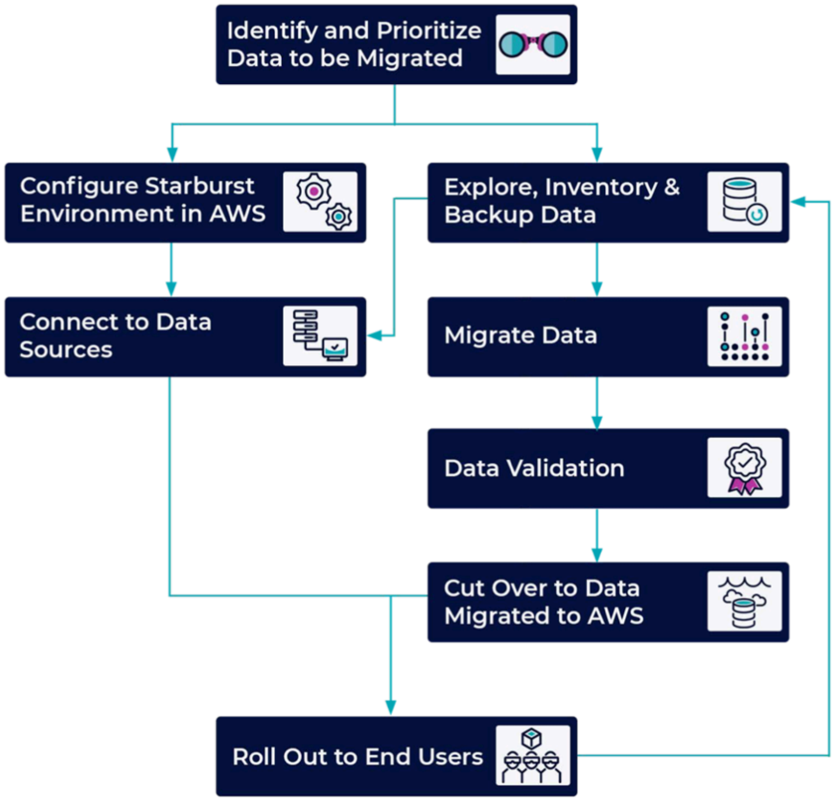


Please note the following:

- Starburst is not a data virtualization platform. It is a SQL-based massively parallel processing (MPP) query engine that forms the basis of an overall data mesh strategy for analytics.
- When Starburst is deployed as part of a migration, it has direct connectivity to the existing on-premises infrastructure.
- Starburst provides several built-in enterprise and open-source connectors that facilitate connectivity to a variety of legacy systems. For a full list of connectors and their capabilities, see Connectors in the *Starburst Enterprise user guide*.
- Starburst can query data in real-time from on-premises data sources. This prevents interruptions of regular business operations while data is being migrated.
- If you are migrating from an existing on-premises Starburst Enterprise deployment, you can use a special connector, *Starburst Stargate*, to connect your Starburst Enterprise cluster in AWS directly to your on-premises cluster. This provides additional performance benefits when business users and data analysts are federating queries from the AWS Cloud to your on-premises environment.

High-level process overview

You can accelerate data migration projects by using Starburst because Starburst enables insights across all of your data, prior to migrating it. The following image shows the typical process for migrating data by using Starburst.



Roles

The following roles are typically required to complete a migration using Starburst:

- **Cloud administrator** – Responsible for making cloud resources available to run the Starburst Enterprise application
- **Starburst administrator** – Responsible for installing, configuring, managing, and supporting the Starburst application
- **Data engineer**– Responsible for:
  - Migrating the legacy data to the cloud
  - Building semantic views to support analytics
- **Solution or system owner** – Responsible for the overall solution implementation

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.

Other tools

- Helm – Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.
- Starburst Enterprise – Starburst Enterprise is a SQL-based massively parallel processing (MPP) query engine that forms the basis of an overall data mesh strategy for analytics.
- Starburst Stargate – Starburst Stargate links catalogs and data sources in one Starburst Enterprise environment, such as a cluster in an on-premises data center, to the catalogs and data sources in another Starburst Enterprise environment, such as a cluster in the AWS Cloud.

Epics

Assess the data

Task	Description	Skills required
Identify and prioritize your data.	Identify the data you want to move. Large, on-premises legacy systems can include core data that you want to migrate alongside data that you don't want to move or can't be moved because of compliance reasons. Starting with a data inventory helps you prioritize which data you should target first. For more information, see Get started with automated portfolio discovery.	Data engineer, DBA

Task	Description	Skills required
Explore, inventory, and back up your data.	Validate the quality, quantity, and relevance of the data for your use case. Back up or create a snapshot of the data as needed, and finalize the target environment for the data.	Data engineer, DBA

Set up the Starburst Enterprise environment

Task	Description	Skills required
Configure Starburst Enterprise in the AWS Cloud.	While data is being catalogued, set up Starburst Enterprise in a managed Amazon EKS cluster. For more information see, Deploying with Kubernetes in the *Starburst Enterprise reference documentation*. This allows business-as-usual analytics while data migration is in process.	AWS administrator, App developer
Connect Starburst to the data sources.	After you have identified the data and set up Starburst Enterprise, connect Starburst to the data sources. Starburst reads data directly from the data source as a SQL query. For more information, see the Starburst Enterprise reference documentation.	AWS administrator, App developer

Migrate the data

Task	Description	Skills required
Build and run the ETL pipelines.	Begin the data migration process. This activity can occur at the same time as business-as-usual analytics . For the migration, you can use a third-party product or Starburst. Starburst has the capability to both read and write data across different sources. For more information, see the Starburst Enterprise reference documentation.	Data engineer
Validate the data.	After the data has been migrated, validate the data to ensure all required data has been moved and is intact.	Data engineer, DevOps engineer

Cut over

Task	Description	Skills required
Cut over the data.	After data migration and validation is complete, you can cut over the data. This involves changing the data connection links in Starburst. Instead of pointing at the on-premises sources, you point to the new cloud sources and update the semantic	Data engineer, Cutover lead

Task	Description	Skills required
	views. For more informati on, see Connectors in the *Starburst Enterprise reference documentation*.	
Roll out to users.	Data consumers begin working off the migrated data sources. This process is invisible to the analytics end users.	Cutover lead, Data engineer

Related resources

AWS Marketplace

- Starburst Galaxy
- Starburst Enterprise
- Starburst Data JumpStart
- Starburst Enterprise with Graviton

Starburst documentation

- Starburst Enterprise user guide
- Starburst Enterprise reference documentation

Other AWS documentation

- Get started with automated portfolio discovery (AWS Prescriptive Guidance)
- Optimizing Cloud Infrastructure Cost and Performance with Starburst on AWS (AWS blog post)

Optimize the ETL ingestion of input file size on AWS

Apoorva Patrikar, Amazon Web Services

Summary

This pattern shows you how to optimize the ingestion step of the extract, transform, and load (ETL) process for big data and Apache Spark workloads on AWS Glue by optimizing file size before processing your data. Use this pattern to prevent or resolve the *small files problem*. That is, when a large number of small files slows down data processing due to the aggregate size of the files. For example, hundreds of files that are only a few hundred kilobytes each can significantly slow down data processing speeds for your AWS Glue jobs. This is because AWS Glue must perform internal list functions on Amazon Simple Storage Service (Amazon S3) and YARN (Yet Another Resource Negotiator) must store a large amount of metadata. To improve data processing speeds, you can use grouping to enable your ETL tasks to read a group of input files into a single in-memory partition. The partition automatically groups smaller files together. Alternatively, you can use custom code to add batch logic to your existing files.

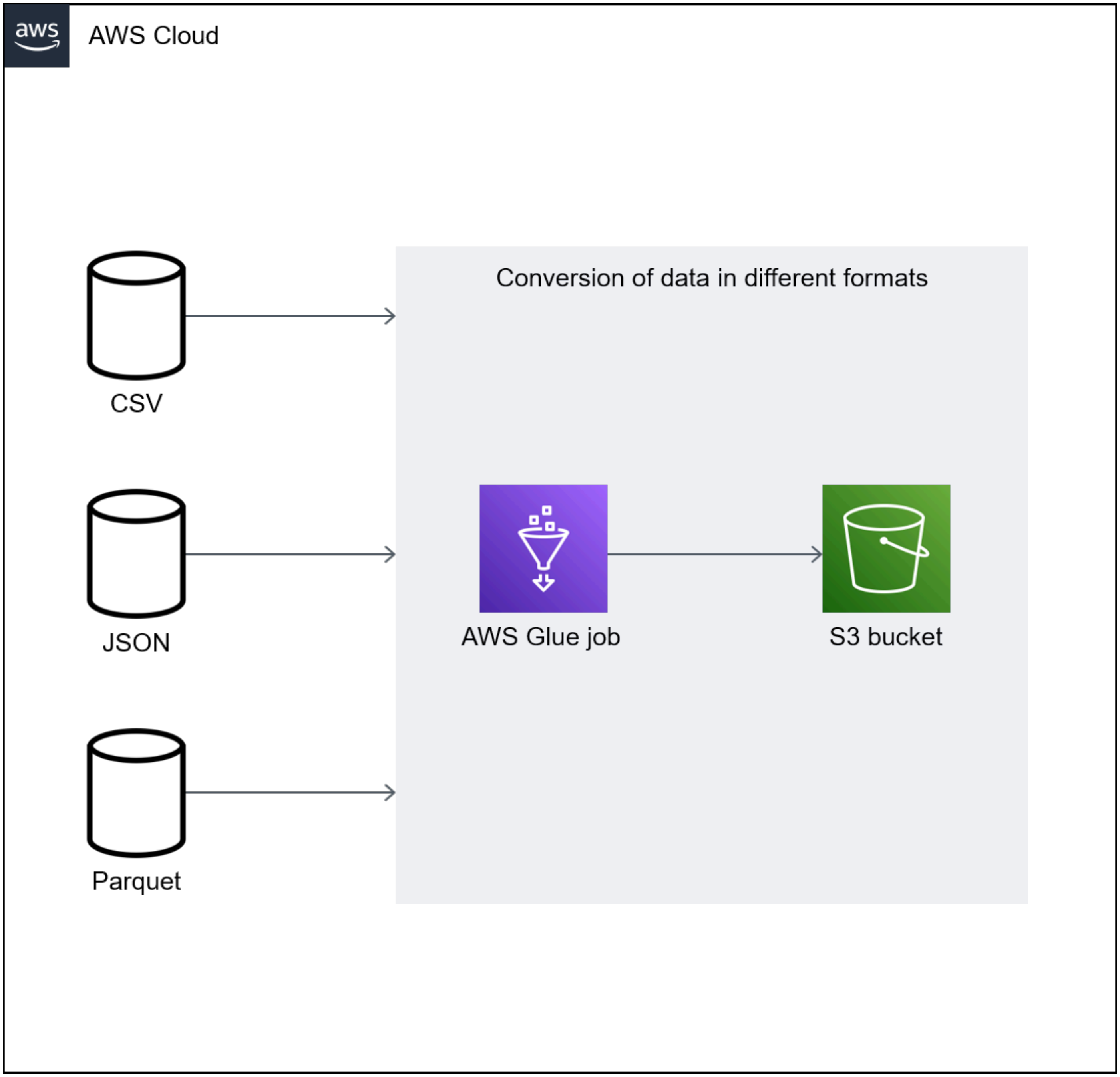
Prerequisites and limitations

Prerequisites

- An active AWS account
- One or more AWS glue jobs
- One or more big data or Apache Spark workloads
- An S3 bucket

Architecture

The following pattern shows how data in different formats is processed by an AWS Glue job and then stored in an S3 bucket to get visibility into performance.



The diagram shows the following workflow:

1. **Note**

   An AWS Glue job converts small files in CSV, JSON, and Parquet format to dynamic frames. : The size of the input file has the most significant impact on the performance of the AWS Glue job.

2. The AWS Glue job performs internal list functions in an S3 bucket.

Tools

- AWS Glue is a fully managed ETL service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Epics

Use grouping to optimize ETL ingestion during reading

Task	Description	Skills required
Specify the group size.	If you have more than 50,000 files, grouping is done by default. However, you can also use grouping for less than 50,000 files by specifying the group size in the `connectionOptions` parameter. The `connectionOptions` parameter is in the `create_dynamic_frame.from_options` method.	Data engineer
Write the grouping code.	Use the `create_dynamic_frame` method to create a dynamic frame. For example: `S3bucket_node1 = glueContext.create_dynamic_frame.from_options(`	Data engineer

Task	Description	Skills required

	`format_options={"multiline": False},`
	`connection_type="s3",`
	`format="json",`
	`connection_options={`
	`"paths": ["s3://bucket/prefix/file.json"],`
	`"recurse": True,`
	`"groupFiles": 'inPartition',`
	`"groupSize": 1048576`
	`},`
	`transformation_ctx="S3bucket_node1",`
	`)`
	
	Note
	Use `groupFiles` to group files in an Amazon S3 partition group. Use `groupSize` to set the target size of the group to be read in memory. Specify `groupSize` in bytes (1048576 = 1 MB).	

Add the code to the workflow.	Add the grouping code to your job workflow in AWS Glue.	Data engineer

Use custom logic to optimize ETL ingestion

Task	Description	Skills required
Choose the language and processing platform.	Choose the scripting language and processing platform tailored to your use case.	Cloud architect
Write the code.	Write the custom logic to batch your files together.	Cloud architect
Add the code to the workflow.	Add the code to your job workflow in AWS Glue. This enables your custom logic to be applied every time the job is run.	Data engineer

Repartition when writing data after transformation

Task	Description	Skills required
Analyze consumption patterns.	Find out how downstream applications will use the data you write. For example, if they query data each day and you only partition data per Region or have very small output files, such as 2.5 KB per file, then this is not optimal for consumption.	DBA
Repartition data before writing.	Repartition based on joins or queries during processin g (based on processing logic) and after processing (based on consumption). For	Data engineer

Task	Description	Skills required
	example, repartition based on byte size, such as `.repartition(100000)`, or repartition based on columns, such as `.repartition("column_name")`.	

Related resources

- Reading input files in larger groups
- Monitoring AWS Glue
- Monitoring AWS Glue using Amazon CloudWatch metrics
- Job monitoring and debugging
- Getting started with serverless ETL on AWS Glue

Additional information

Determining file size

There is no straightforward way to determine if a file size is too big or too small. The impact of file size on processing performance depends on the configuration of your cluster. In core Hadoop, we recommend that you use files that are 128 MB or 256 MB to make the most of the block size.

For most text file workloads on AWS Glue, we recommended a file size between 100 MB and 1 GB for a 5-10 DPU cluster. To figure out the best size of input files, monitor the preprocessing section of your AWS Glue job, and then check the CPU utilization and memory utilization of the job.

Additional considerations

If performance in the early ETL stages is a bottleneck, consider grouping or merging the data files before processing. If you have complete control on the file generation process, it can be even more efficient to aggregate data points on the source system itself before the raw data is sent to AWS.

Orchestrate an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions

Sandip Gangapadhyay, Amazon Web Services

Summary

This pattern describes how to build a serverless extract, transform, and load (ETL) pipeline to validate, transform, compress, and partition a large CSV dataset for performance and cost optimization. The pipeline is orchestrated by AWS Step Functions and includes error handling, automated retry, and user notification features.

When a CSV file is uploaded to an Amazon Simple Storage Service (Amazon S3) bucket source folder, the ETL pipeline starts to run. The pipeline validates the content and the schema of the source CSV file, transforms the CSV file to a compressed Apache Parquet format, partitions the dataset by year, month, and day, and stores it in a separate folder for analytics tools to process.

The code that automates this pattern is available on GitHub, in the ETL Pipeline with AWS Step Functions repository.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) installed and configured with your AWS account, so that you can create AWS resources by deploying an AWS CloudFormation stack. We recommend using AWS CLI version 2. For instructions, see Installing or updating to the latest version of the AWS CLI in the AWS CLI documentation. For configuration instructions, see Configuration and credential file settings in the AWS CLI documentation.
- An Amazon S3 bucket.
- A CSV dataset with the correct schema. (The code repository included with this pattern provides a sample CSV file with the correct schema and data type that you can use.)
- A web browser that supports the AWS Management Console. (See the list of supported browsers.)
- AWS Glue console access.

- AWS Step Functions console access.

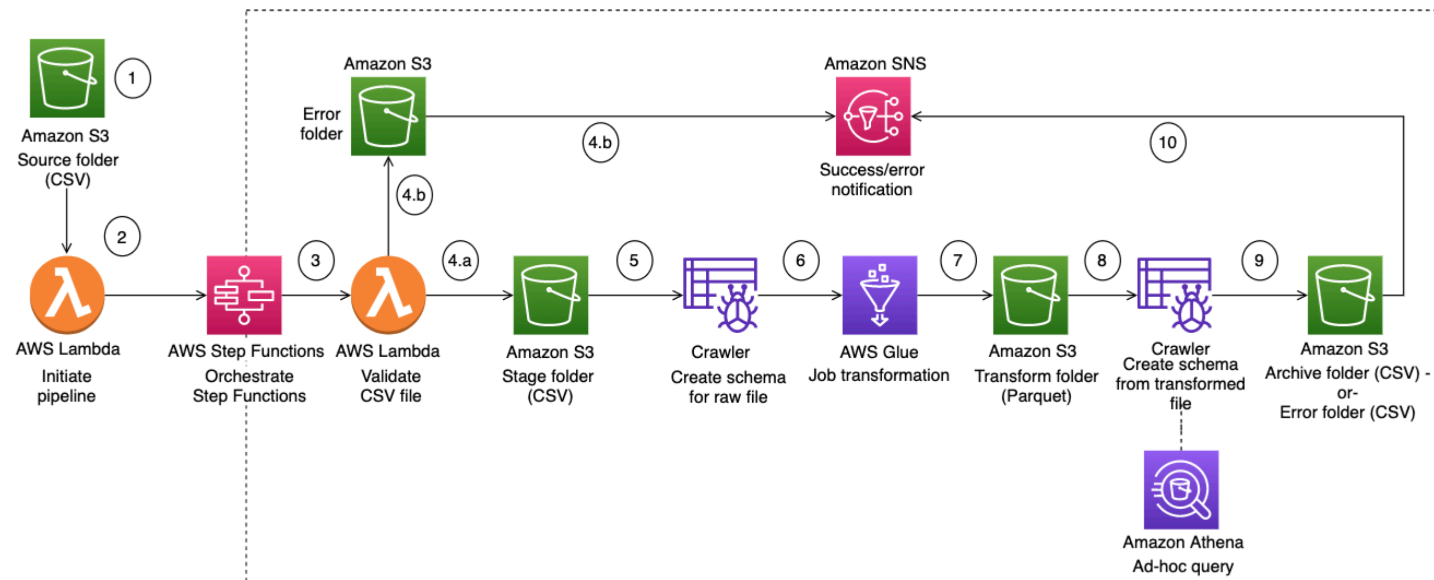
Limitations

- In AWS Step Functions, the maximum limit for keeping history logs is 90 days. For more information, see Step Functions service quotas in the AWS Step Functions documentation.

Product versions

- Python 3.13 for AWS Lambda
- AWS Glue version 4.0

Architecture



The workflow illustrated in the diagram consists of these high-level steps:

1. The user uploads a CSV file into the source folder in Amazon S3.
2. An Amazon S3 notification event initiates an AWS Lambda function that starts the AWS Step Functions state machine.
3. The Lambda function validates the schema and data type of the raw CSV file.
4. Depending on the validation results:
    a. If validation of the source file succeeds, the file moves to the stage folder for further processing.

   b. If validation fails, the file moves to the error folder, and an error notification is sent through Amazon Simple Notification Service (Amazon SNS).

5. An AWS Glue crawler creates the schema of the raw file from the stage folder in Amazon S3.
6. An AWS Glue job transforms, compresses, and partitions the raw file into Parquet format.
7. The AWS Glue job also moves the file to the transform folder in Amazon S3.
8. The AWS Glue crawler creates the schema from the transformed file. The resulting schema can be used by any analytics job. You can also use Amazon Athena to run ad-hoc queries.
9. If the pipeline completes without errors, the schema file is moved to the archive folder. If any errors are encountered, the file is moved to the error folder instead.
10. Amazon SNS sends a notification that indicates success or failure based on the pipeline completion status.

All the AWS resources used in this pattern are serverless. There are no servers to manage.

Tools

AWS services

- AWS Glue – AWS Glue is a fully managed ETL service that makes it easy for customers to prepare and load their data for analytics.
- AWS Step Functions – AWS Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications. Through the AWS Step Functions graphical console, you see your application's workflow as a series of event-driven steps.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a highly available, durable, secure, fully managed pub/sub messaging service that enables you to decouple microservices, distributed systems, and serverless applications.
- AWS Lambda – AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

Code

The code for this pattern is available on GitHub, in the [ETL Pipeline with AWS Step Functions] repository. The code repository contains the following files and folders:

- `template.yml` – AWS CloudFormation template for creating the ETL pipeline with AWS Step Functions.
- `parameter.json` – Contains all parameters and parameter values. You update this file to change parameter values, as described in the *Epics* section.
- `myLayer/python` folder – Contains Python packages needed to create the required AWS Lambda layer for this project.
- `lambda` folder – Contains the following Lambda functions:
  - `move_file.py` – Moves the source dataset to the archive, transform, or error folder.
  - `check_crawler.py` – Checks the status of the AWS Glue crawler as many times as configured by the `RETRYLIMIT` environment variable before it sends a failure message.
  - `start_crawler.py` – Starts the AWS Glue crawler.
  - `start_step_function.py` – Starts AWS Step Functions.
  - `start_codebuild.py` – Starts the AWS CodeBuild project.
  - `validation.py` – Validates the input raw dataset.
  - `s3object.py` – Creates the required directory structure inside the Amazon S3 bucket.
  - `notification.py` – Sends success or error notifications at the end of the pipeline.

To use the sample code, follow the instructions in the *Epics* section.

Epics

Prepare the source files

Task	Description	Skills required
Clone the sample code repository.	1. Open the ETL Pipeline with AWS Step Functions repository. 2. Choose **Code** on the main repository page, above the file list, and copy the URL	Developer

Task	Description	Skills required
	listed under **Clone with HTTPS**. 3. Change your working directory to the location where you want to store the sample files. 4. At a terminal or command prompt, type the command: `git clone <repoURL>` where `<repoURL>` refers to the URL you copied in step 2.	

Task	Description	Skills required
Update parameter values.	In your local copy of the repository, edit the `parameter.json` file and update the default parameter values as follows: • `pS3BucketName` — The name of the Amazon S3 bucket for storing the datasets. The template will create this bucket for you. The bucket name must be globally unique. • `pSourceFolder` — The name of the folder inside the Amazon S3 bucket that will be used to upload the source CSV file. • `pStageFolder` — The name of the folder inside the Amazon S3 bucket that will be used as the staging area during the process. • `pTransformFolder` — The name of the folder inside the Amazon S3 bucket that will be used to store transformed and partitioned datasets. • `pErrorFolder` — The folder inside the Amazon S3 bucket that the source CSV file will be moved to if it can't be validated.	Developer

Task	Description	Skills required
	• `pArchiveFolder` – The name of the folder inside the Amazon S3 bucket that will be used to archive the source CSV file. • `pEmailforNotification` – A valid email address for receiving success/error notifications. • `pPrefix` – A prefix string that will be used in the AWS Glue crawler name. • `pDatasetSchema` – The dataset schema that the source file will be validated against. The Cerberus Python package is used for source dataset validation. For more information, see the Cerberus website.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Upload the source code to the Amazon S3 bucket.</td><td>Before you deploy the AWS CloudFormation template that automates the ETL pipeline, you must package the source files for the template and upload them to an Amazon S3 bucket. To do this, run the following AWS CLI command with your preconfigured profile:

<pre><code>aws cloudformation
 package --template-
file template.yml --s3-
bucket <bucket_name>
 --output-template-
file packaged.template
 --profile <profile_
name></code></pre>
where:
<code><bucket_name></code> is the name of an existing Amazon S3 bucket in the AWS Region where you want to deploy the stack. This bucket is used to store the source code package for the CloudFormation template.
<code><profile_name></code> is a valid AWS CLI profile that you preconfigured when you set up AWS CLI.</td><td>Developer</td></tr>
</table>

Create the stack

Task	Description	Skills required
Deploy the CloudFormation template.	To deploy the AWS CloudFormation template, run the following AWS CLI command: `aws cloudformation deploy --stack-name <stack_name> --template-file packaged.template --parameter-overrides file://parameter.json --capabilities CAPABILITY_IAM --profile <profile_name>` where: • `<stack_name>` is a unique identifier for the CloudFormation stack. • `<profile-name>` is your preconfigured AWS CLI profile.	Developer
Check progress.	On the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/), check the progress of stack development. When the status is `CREATE_COMPLETE`, the stack has been deployed successfully.	Developer
Note the AWS Glue database name.	The **Outputs** tab for the stack displays the name of the AWS	Developer

Task	Description	Skills required
	Glue database. The key name is `GlueDBOutput` .	

Test the pipeline

Task	Description	Skills required
Start the ETL pipeline.	1. Navigate to the source folder (`source`, or the folder name you set in the `parameter.json` file) inside the Amazon S3 bucket. 2. Upload a sample CSV file to this folder. (The code repository provides a sample file called `Sample_Bank_Transaction_Raw_Dataset.csv` that you can use.) Uploading the file will start the ETL pipeline through Step Functions. 3. On the [Step Functions console](https://console.aws.amazon.com/states/), check the ETL pipeline status.	Developer
Check for the partitioned dataset.	When the ETL pipeline completes, verify that the partitioned dataset is available in the Amazon S3 transform folder (`transform` , or the folder name you set	Developer

Task	Description	Skills required
	in the `parameter.json` file).	
Check for the partitioned AWS Glue database.	1. On the AWS Glue console, select the AWS Glue database created by the stack (this is the database that you noted in the previous epic). 2. Verify that the partitioned table is available in the AWS Glue Data Catalog.	Developer
Run queries.	(Optional) Use Amazon Athena to run ad-hoc queries on the partitioned and transformed database. For instructions, see Run SQL queries in Amazon Athena in the AWS documentation.	Database analyst

Troubleshooting

Issue	Solution
AWS Identity and Access Management (IAM) permissions for the AWS Glue job and crawler	If you further customize the AWS Glue job or the crawler, be sure to grant the appropriate IAM permissions in the IAM role used by the AWS Glue job, or provide data permissions to AWS Lake Formation. For more information, see the AWS documentation.

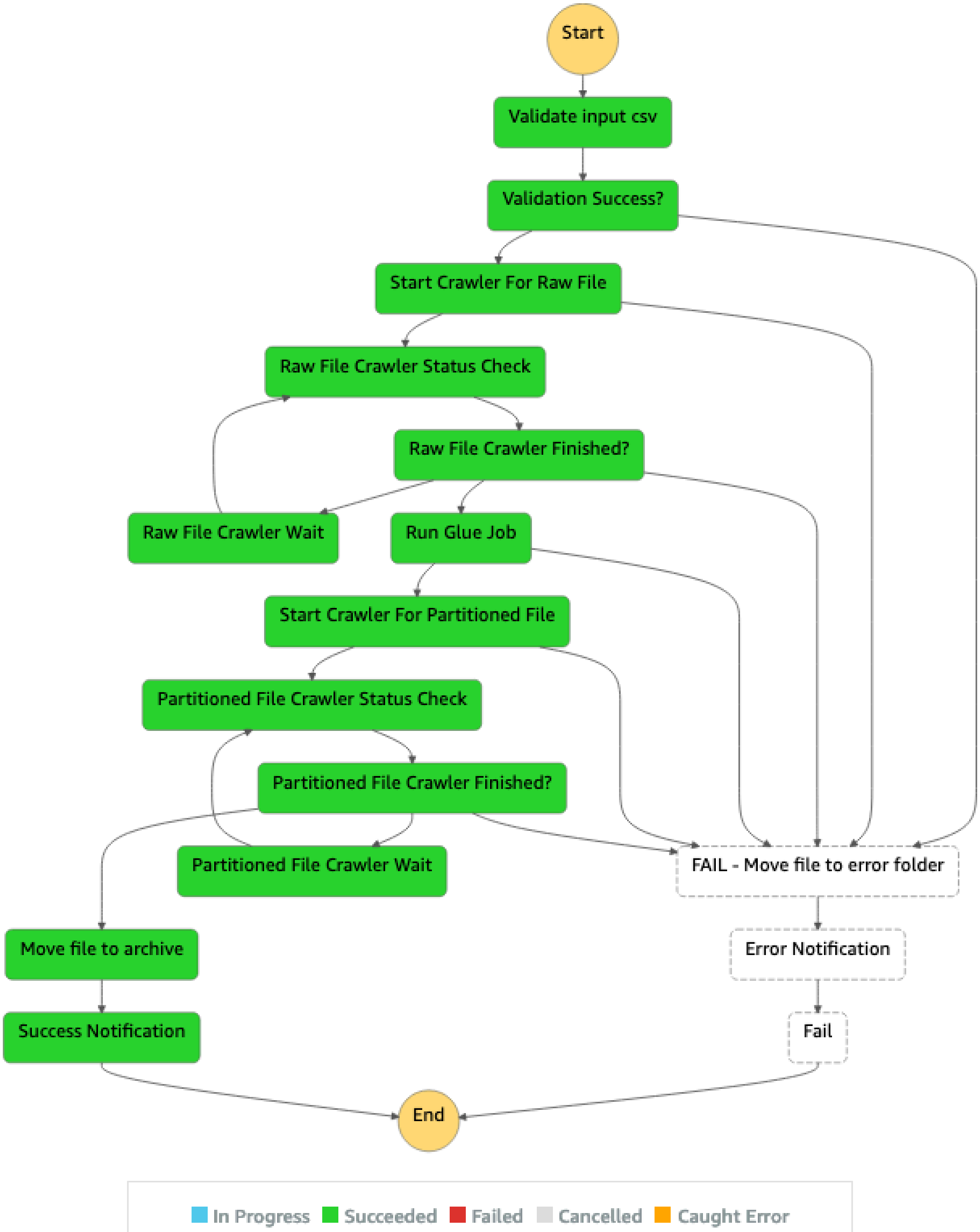
Related resources

AWS service documentation

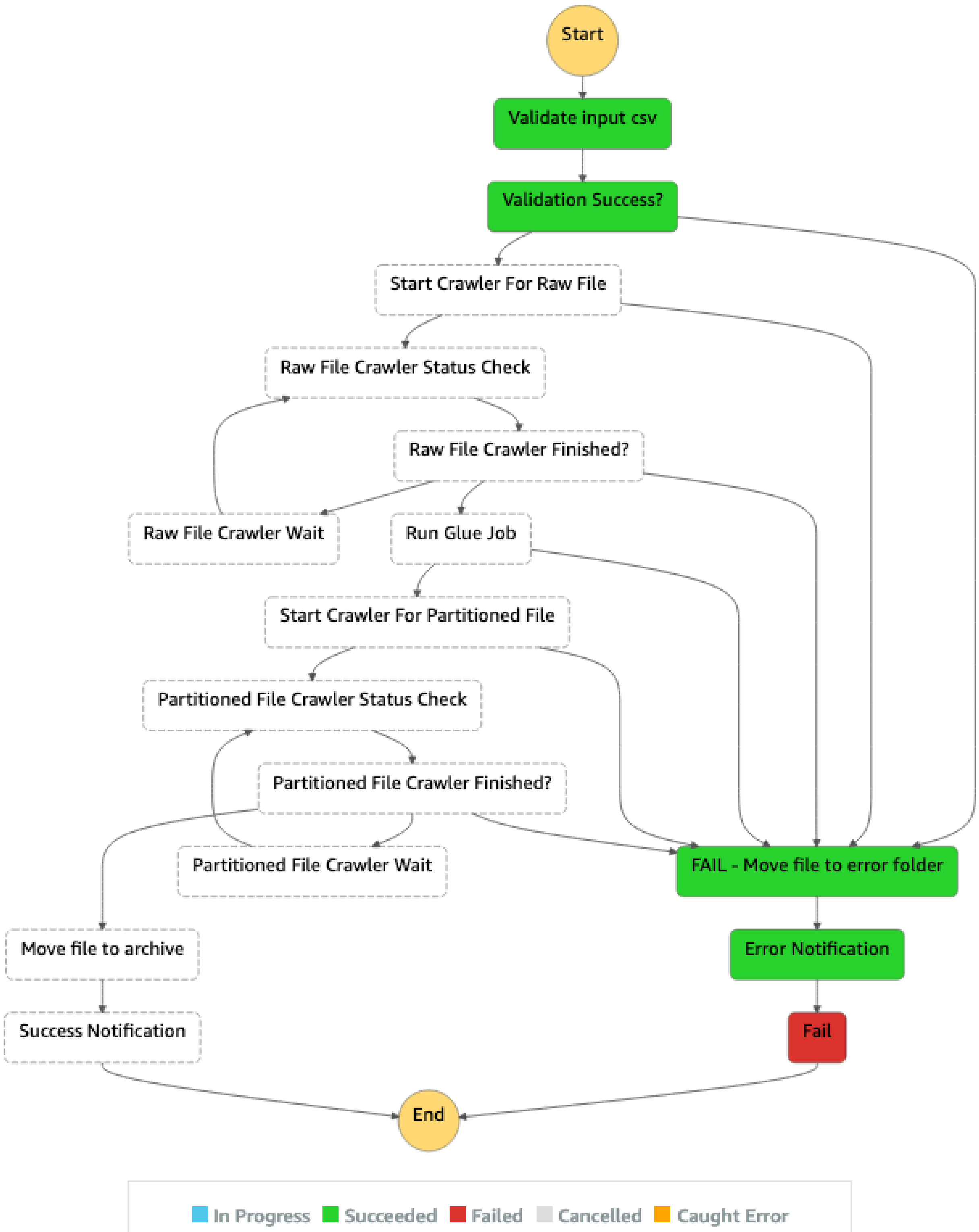
- AWS Step Functions
- AWS Glue
- AWS Lambda
- Amazon S3
- Amazon SNS

Additional information

The following diagram shows the AWS Step Functions workflow for a successful ETL pipeline, from the AWS Step Functions **Inspector** panel.



The following diagram shows the AWS Step Functions workflow for an ETL pipeline that fails because of an input validation error, from the Step Functions **Inspector** panel.


Start
Validate input csv
Validation Success?
Start Crawler For Raw File
Raw File Crawler Status Check
Raw File Crawler Finished?
Raw File Crawler Wait
Run Glue Job
Start Crawler For Partitioned File
Partitioned File Crawler Status Check
Partitioned File Crawler Finished?
Partitioned File Crawler Wait
FAIL - Move file to error folder
Move file to archive
Error Notification
Success Notification
Fail
End
In Progress
Succeeded
Failed
Cancelled
Caught Error

Perform advanced analytics using Amazon Redshift ML

Po Hong and Chyanna Antonio, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, you can use Amazon Redshift machine learning (Amazon Redshift ML) to perform ML analytics on data stored in either an Amazon Redshift cluster or on Amazon Simple Storage Service (Amazon S3). Amazon Redshift ML supports supervised learning, which is typically used for advanced analytics. Use cases for Amazon Redshift ML include revenue forecasting, credit card fraud detection, and customer lifetime value (CLV) or customer churn predictions.

Amazon Redshift ML makes it easy for database users to create, train, and deploy ML models by using standard SQL commands. Amazon Redshift ML uses Amazon SageMaker Autopilot to automatically train and tune the best ML models for classification or regression based on your data, while you retain control and visibility.

All interactions between Amazon Redshift, Amazon S3, and Amazon SageMaker are abstracted away and automated. After the ML model is trained and deployed, it becomes available as a user-defined function (UDF) in Amazon Redshift and can be used in SQL queries.

This pattern complements the Create, train, and deploy ML models in Amazon Redshift using SQL with Amazon Redshift ML from the AWS Blog, and the Build, train, and deploy an ML model with Amazon SageMaker tutorial from the Getting Started Resource Center.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Existing data in an Amazon Redshift table

Skills

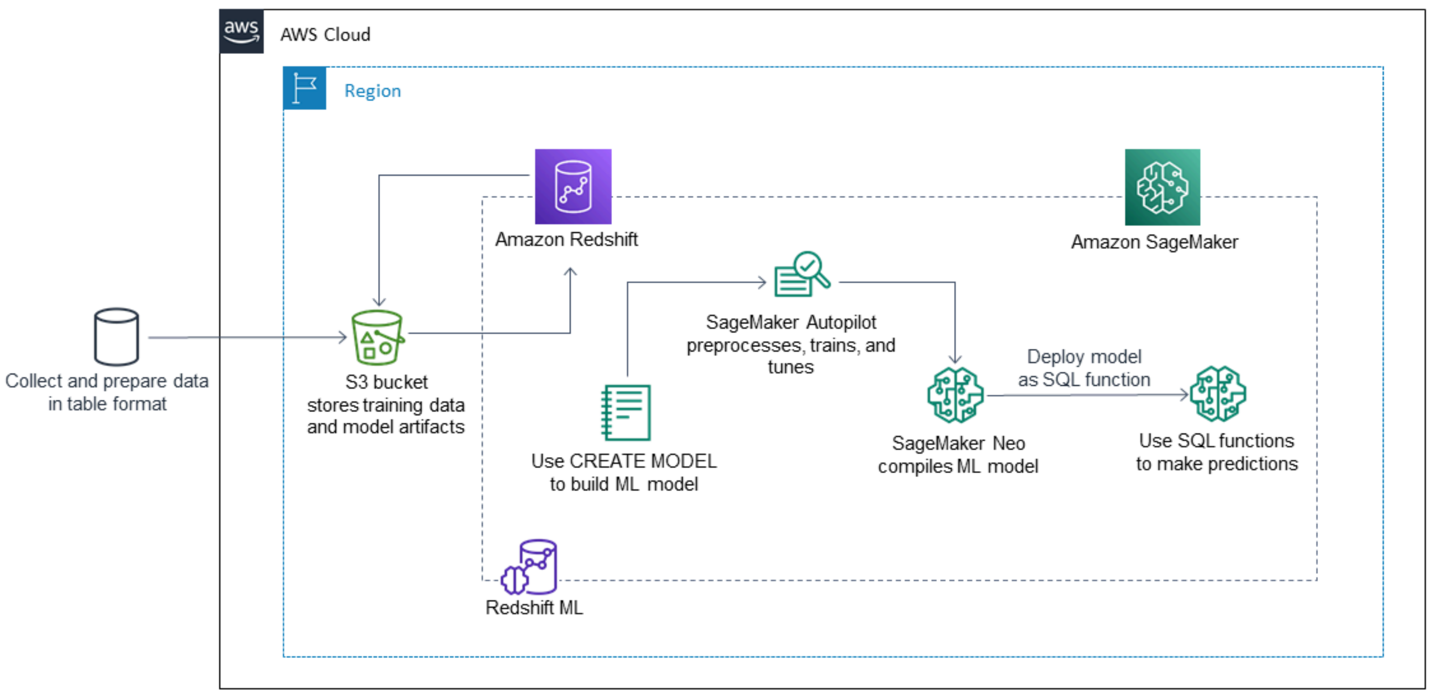
- Familiarity with terms and concepts used by Amazon Redshift ML, including *machine learning*, *training*, and *prediction*. For more information about this, see Training ML models in the Amazon Machine Learning (Amazon ML) documentation.

- Experience with Amazon Redshift user setup, access management, and standard SQL syntax. For more information about this, see Getting started with Amazon Redshift in the Amazon Redshift documentation.
- Knowledge and experience with Amazon S3 and AWS Identity and Access Management (IAM).
- Experience running commands in AWS Command Line Interface (AWS CLI) is also beneficial but not required.

Limitations

- The Amazon Redshift cluster and S3 bucket must be located in the same AWS Region.
- This pattern's approach only supports supervised learning models such as regression, binary classification, and multiclass classification.

Architecture



The following steps explain how Amazon Redshift ML works with SageMaker to build, train, and deploy an ML model:

1. Amazon Redshift exports training data to an S3 bucket.
2. SageMaker Autopilot automatically preprocesses the training data.

3. After the `CREATE MODEL` statement is invoked, Amazon Redshift ML uses SageMaker for training.
4. SageMaker Autopilot searches for and recommends the ML algorithm and optimal hyper-parameters that optimize the evaluation metrics.
5. Amazon Redshift ML registers the output ML model as a SQL function in the Amazon Redshift cluster.
6. The ML model's function can be used in a SQL statement.

Technology stack

- Amazon Redshift
- SageMaker
- Amazon S3

Tools

- Amazon Redshift – Amazon Redshift is an enterprise-level, petabyte scale, fully managed data warehousing service.
- Amazon Redshift ML – Amazon Redshift machine learning (Amazon Redshift ML) is a robust, cloud-based service that makes it easy for analysts and data scientists of all skill levels to use ML technology.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- Amazon SageMaker – SageMaker is a fully managed ML service.
- Amazon SageMaker Autopilot – SageMaker Autopilot is a feature-set that automates key tasks of an automatic machine learning (AutoML) process.

Code

You can create a supervised ML model in Amazon Redshift by using the following code:

```
"CREATE MODEL customer_churn_auto_model
FROM (SELECT state,
             account_length,
             area_code,
             total_charge/account_length AS average_daily_spend,
```

```
            cust_serv_calls/account_length AS average_daily_cases,
            churn
      FROM customer_activity
      WHERE  record_date < '2020-01-01'
     )
TARGET churn
FUNCTION ml_fn_customer_churn_auto
IAM_ROLE 'arn:aws:iam::XXXXXXXXXXXX:role/Redshift-ML'
SETTINGS (
  S3_BUCKET 'your-bucket'
);")
```

Note

The SELECT state can refer to Amazon Redshift regular tables, Amazon Redshift Spectrum external tables, or both.

Epics

Prepare a training and test dataset

Task	Description	Skills required
Prepare a training and test dataset.	Sign in to the AWS Management Console and open the Amazon SageMaker console. Follow the instructi ons from the Build, train, and deploy a machine learning model tutorial to create a .csv or Apache Parquet file that has a label column (*supervised training*) and no header. **Note** We recommend that you shuffle and split	Data scientist

Task	Description	Skills required
	the raw dataset into a training set for the model's training (70 percent) and a test set for the model's performance evaluation (30 percent).	

Prepare and configure the technology stack

Task	Description	Skills required
Create and configure an Amazon Redshift cluster.	On the Amazon Redshift console, create a cluster according to your requirements. For more information about this, see [Create a cluster](#) in the Amazon Redshift documentation. **Important** Amazon Redshift clusters must be created with the `SQL_PREVIEW` maintenance track. For more information about preview tracks, see [Choosing cluster maintenance tracks](#) in	DBA, Cloud architect

Task	Description	Skills required
	the Amazon Redshift documentation.	
Create an S3 bucket to store training data and model artifacts.	On the Amazon S3 console, create an S3 bucket for the training and test data. For more information about creating an S3 bucket, see [Create an S3 bucket](#) from AWS Quick Starts. **Important** Make sure that your Amazon Redshift cluster and S3 bucket are in the same Region.	DBA, Cloud architect
Create and attach an IAM policy to the Amazon Redshift cluster.	Create an IAM policy to allow the Amazon Redshift cluster to access SageMaker and Amazon S3. For instructions and steps, see [Cluster setup for using Amazon Redshift ML](#) in the Amazon Redshift documentation.	DBA, Cloud architect

Task	Description	Skills required
Allow Amazon Redshift users and groups to access schemas and tables.	Grant permissions to allow users and groups in Amazon Redshift to access internal and external schemas and tables. For steps and instructions, see Managing permissions and ownership in the Amazon Redshift documentation.	DBA

Create and train the ML model in Amazon Redshift

Task	Description	Skills required
Create and train the ML model in Amazon Redshift.	Create and train your ML model in Amazon Redshift ML. For more information, see the `CREATE MODEL` statement in the Amazon Redshift documentation.	Developer, Data scientist

Perform batch inference and prediction in Amazon Redshift

Task	Description	Skills required
Perform inference using the generated ML model function.	For more information about performing inference by using the generated ML model function, see Prediction in the Amazon Redshift documentation.	Data scientist, Business intelligence user

Related resources

Prepare a training and test dataset

- Building, training, and deploying a machine learning model with Amazon SageMaker

Prepare and configure the technology stack

- Creating an Amazon Redshift cluster
- Choosing Amazon Redshift cluster maintenance tracks
- Creating an S3 bucket
- Setting up an Amazon Redshift cluster for using Amazon Redshift ML
- Managing permissions and ownership in Amazon Redshift

Create and train the ML model in Amazon Redshift

- CREATE MODEL statement in Amazon Redshift

Perform batch inference and prediction in Amazon Redshift

- Prediction in Amazon Redshift

Other resources

- Getting started with Amazon Redshift ML
- Creating, training, and deploying ML models in Amazon Redshift using SQL with Amazon Redshift ML
- Amazon Redshift partners
- AWS machine learning competency partners

Query Amazon DynamoDB tables with SQL by using Amazon Athena

Gavin Perrie, Ajit Ambike, and Brad Yates, Amazon Web Services

Summary

If your data includes sources other than Amazon Simple Storage Service (Amazon S3), you can use federated queries to access those relational, non-relational, object, or custom data sources. This pattern shows how to configure federated query access through Amazon Athena to Amazon DynamoDB by using an SQL data source connector.

Using this pattern you can do the following:

- Query DynamoDB tables by using SQL.
- Run federated SQL queries in Athena and join DynamoDB tables with other supported data sources.

Prerequisites and limitations

Prerequisites

- A DynamoDB table.
- An Athena workgroup set to use Athena engine version 2. For instructions, see the Athena documentation.
- An S3 bucket where the `AthenaDynamoDBConnector` AWS Lambda function can spill the data. The S3 bucket and the Lambda function must be in the same AWS Region.

If this is your first time accessing Athena, you will need an additional S3 bucket to use as the query result location. For instructions, see the Athena documentation.

Limitations

- Write operations such as INSERT INTO are not supported.

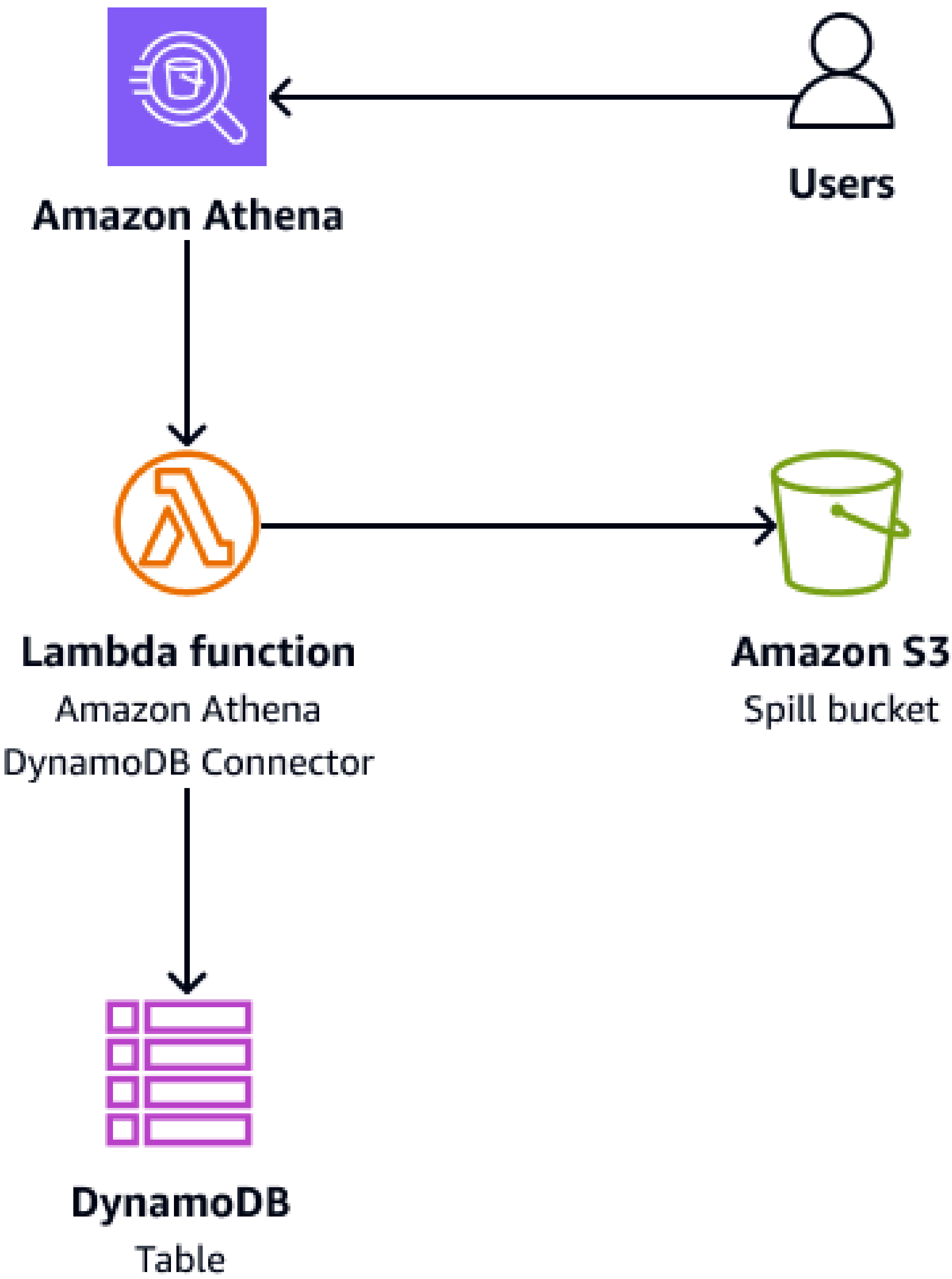
Product versions

- Athena Query Federation releases on GitHub

Architecture

Target architecture

The following diagram shows the connection flow after the pattern is established. The user connects to Amazon Athena to provide the query. Athena passes the query and target to the DynamoDB data source connector Lambda function, which retrieves and returns the data to Athena. If large amounts of data are returned, Athena stores the temporary results in the spill bucket before packaging and returning the complete dataset.



Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon Simple Storage Service (Amazon S3) by using standard SQL. This pattern uses Amazon Athena

DynamoDB Connector, a tool built using the Amazon Athena Query Federation SDK and installed as an AWS Lambda application through the AWS Serverless Application Repository.

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code repository

The code for this pattern is available in the GitHub Athena Query Federation repository.

Epics

Set up and test the DynamoDB data source connector

Task	Description	Skills required
Deploy the AthenaDynamoDBConnector application.	To deploy AthenaDynamoDBConnecter, do the following: 1. Sign in to the AWS Management Console, and choose the AWS Region that you're using for the DynamoDB table and the spill bucket. 2. Open the **Serverless Application Repository** at https://console.aws.amazon.com/serverlessrepo/.	AWS DevOps

Task	Description	Skills required
	3. In the navigation pane, choose **Available Applications**. 4. For AWS Identity and Access Management (IAM) access, under the search bar, select the **Show apps that create custom IAM roles or resource policies** check box. 5. Search for and select **AthenaDynamoDBConnector**, and ensure that the author listed is **Amazon Athena Federation**. 6. In **Application Settings**, enter the following values: • **SpillBucket** – Location that the function can spill data to. • **AthenaCatalogName** – The name of the Lambda function that will be created. The name will also be used as the **Data Source Name** in Athena. 7. Select the check box to acknowledge the creation of IAM roles and policies. 8. Choose **Deploy**.	

Task	Description	Skills required
Create a data source for Athena.	To create the data source, do the following: 1. Open the Athena console at https://console.aws.amazon.com/athena/. 2. Expand the navigation pane, and choose **Data sources**. 3. Choose **Create data source**. 4. Choose **Amazon DynamoDB**. 5. Enter the **Data Source Name**. 6. Select the Lambda function that you created. 7. Review the details, and choose **Create data source**.	AWS DevOps

Task	Description	Skills required
Use Athena to query the DynamoDB table.	To query the DynamoDB table, do the following: 1. On the Athena console, expand the navigation pane, and choose **Query editor**. 2. On the **Data source** dropdown list, choose the data source that you created. 3. Confirm that the DynamoDB tables are listed under **Tables**. 4. Run the query.	App developer

Troubleshooting

Issue	Solution
Query fails with `GENERIC_INTERNAL_ERROR: The bucket is in this region: <region>`.	Make sure that the Athena spill bucket and Lambda function are created in the same AWS Region.
The newly created data source isn't visible on the Athena console.	Athena data catalogs are regional. Ensure the `AthenaDynamoDBConnector` has been deployed in the Region where you are trying to use Athena.
You're unable to run the query against the newly created data source.	Check that the **Query Result Location** has been set.

Related resources

- Amazon Athena DynamoDB Connector
- Amazon Athena Federated Query

Access, query, and join Amazon DynamoDB tables using Athena

Moinul Al-Mamun, Amazon Web Services

Summary

This pattern shows you how to set up a connection between Amazon Athena and Amazon DynamoDB by using the Amazon Athena DynamoDB connector. The connector uses an AWS Lambda function to query the data in DynamoDB. You don't need to write any code to set up the connection. After the connection is established, you can quickly access and analyze DynamoDB tables by using Athena Federated Query to run SQL commands from Athena. You can also join one or more DynamoDB tables to each other or to other data sources, such as Amazon Redshift or Amazon Aurora.

Prerequisites and limitations

Prerequisites

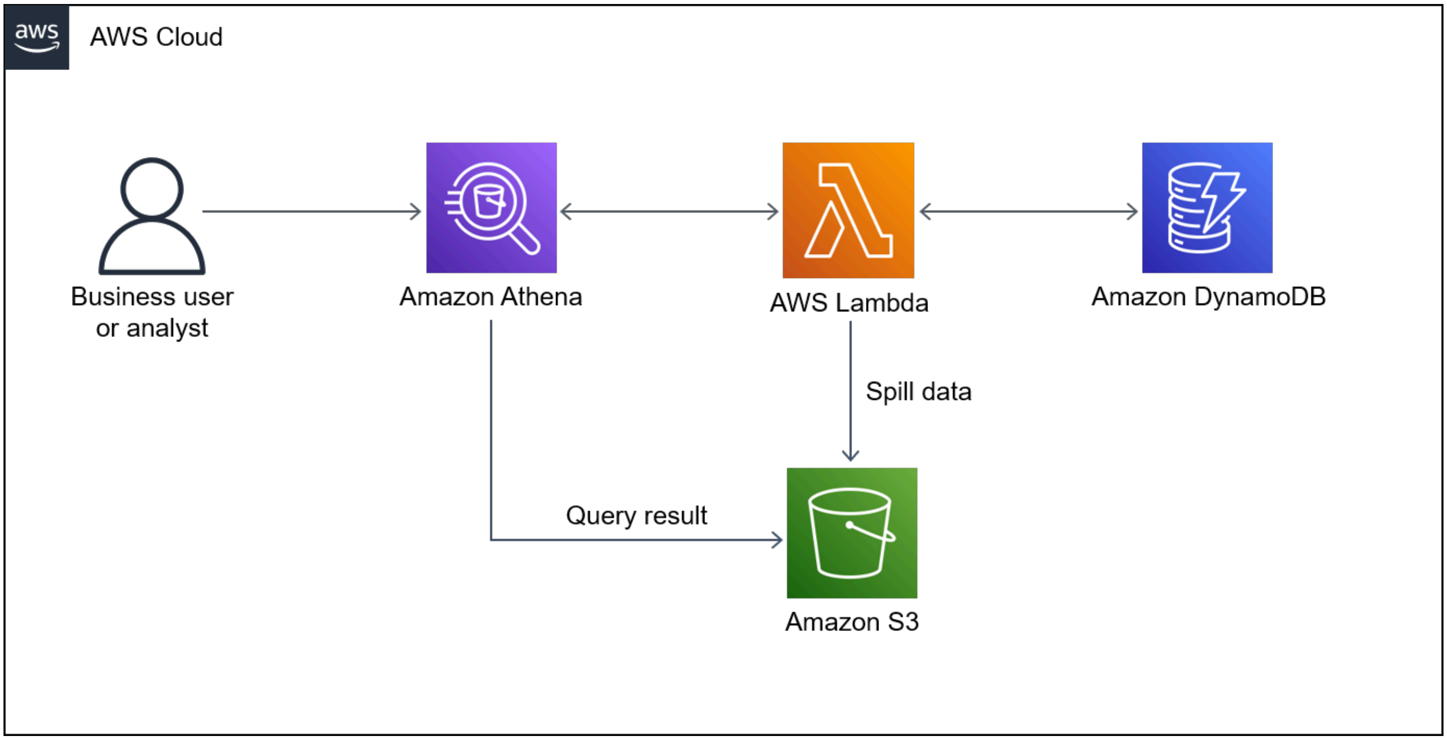
- An active AWS account with permissions to manage DynamoDB tables, Athena Data sources, Lambda, and AWS Identity and Access Management (IAM) roles
- An Amazon Simple Storage Service (Amazon S3) bucket where Athena can store query results
- An S3 bucket where the Athena DynamoDB Connector can save the data in the short term
- An AWS Region that supports Athena engine version 2
- IAM permissions to access Athena and the required S3 buckets
- Amazon Athena DynamoDB Connector, installed

Limitations

There is a cost for querying DynamoDB tables. Table sizes exceeding a few gigabytes (GBs) can incur a high cost. We recommend that you consider cost before performing any full table SCAN operation. For more information, see Amazon DynamoDB pricing. To reduce costs and achieve high performance, we recommend that you always use LIMIT in your query (for example, `SELECT * FROM table1 LIMIT 10`). Also, before you perform a JOIN or GROUP BY query in a production environment, consider the size of your tables. If your tables are too large, consider alternative options such as migrating the table to Amazon S3.

Architecture

The following diagram shows how a user can run a SQL query on a DynamoDB table from Athena.



The diagram shows the following workflow:

1. To query a DynamoDB table, a user runs a SQL query from Athena.
2. Athena initiates a Lambda function.
3. The Lambda function queries the requested data in the DynamoDB table.
4. DynamoDB returns the requested data to the Lambda function. Then, the function transfers the query results to the user through Athena.
5. The Lambda function stores data in the S3 bucket.

Technology stack

- Amazon Athena
- Amazon DynamoDB
- Amazon S3
- AWS Lambda

Tools

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- Amazon Athena DynamoDB Connector is an AWS tool that enables Athena to connect with DynamoDB and access your tables by using SQL queries.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Epics

Create sample DynamoDB tables

Task	Description	Skills required
Create the first sample table.	1. Sign in to the AWS Management Console and open the DynamoDB console. 2. Choose **Create table**. 3. For **Table name**, enter **dydbtable1**. 4. For **Partition key**, enter **PK1**. 5. For **Sort key**, enter **SK1**. 6. In the **Table settings** section, choose **Customize settings**. 7. In the **Table class** section, choose **DynamoDB Standard**.	Developer

Task	Description	Skills required
	8. In the **Read/write capacity settings** section, for **Capacity mode**, choose **On-demand**. 9. In the **Encryption at rest** section, choose **Owned by Amazon DynamoDB**. 10. Choose **Create table**.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Insert sample data into the first table.</td><td>
1. Open the DynamoDB console.
2. In the navigation pane, choose Table, and then choose your table in the Name column.
3. Choose Actions, and then choose Create item.
4. Choose JSON view.
5. In the title bar of the Attributes editor, turn off View DynamoDB JSON.
6. In the Attributes editor, enter the following sample data one by one:
<pre>{
  "PK1": "1234",
  "SK1": "info",
  "Salary": "5000"
 }</pre>
<pre>{
  "PK1": "1235",
  "SK1": "info",
  "Salary": "5200"
 }</pre>
</td><td>Developer</td></tr>
</table>

Task	Description	Skills required
Create the second sample table.	1. Open the [DynamoDB console](). 2. Choose **Create table**. 3. For **Table name**, enter **dydbtable2**. 4. For **Partition key**, enter **PK2**. 5. For **Sort key**, enter **SK2**. 6. In the **Table settings** section, choose **Customize settings**. 7. In the **Table class** section, choose **DynamoDB Standard**. 8. In the **Read/write capacity settings** section, for **Capacity mode**, choose **On-demand**. 9. In the **Encryption at rest** section, choose **Owned by Amazon DynamoDB**. 10. Choose **Create table**.	Developer

Task	Description	Skills required
Insert sample data into the second table.	1. Open the [DynamoDB console](). 2. In the navigation pane, choose **Table**, and then choose your table in the **Name** column. 3. Choose **Actions**, and then choose **Create item**. 4. In the title bar of the **Attributes** editor, turn off **View DynamoDB JSON**. 5. In the **Attributes** editor, enter the following sample data one by one: `{ "PK2": "1234", "SK2": "bonus", "Bonus": "500" }` `{ "PK2": "1235", "SK2": "bonus", "Bonus": "1000" }`	Developer

Create a data source in Athena for DynamoDB

Task	Description	Skills required
Set up the data source connector.	Create a data source for DynamoDB, and then create a	Developer

Task	Description	Skills required
	Lambda function to connect to that data source. 1. Sign in to the AWS Management Console and open the [Athena console](#). 2. In the navigation pane, choose **Data sources**, and then choose **Create data source**. 3. Choose the **Amazon DynamoDB** data source, and then choose **Next**. 4. In the **Data source details** section, for **Data source name**, enter **testDynamoDB**. 5. In the **Connection details** section, select a Lambda function that's already deployed or choose **Create Lambda function** if you don't have a Lambda function to use for this pattern. **Note**: For more information on creating a Lambda function, see [Getting started with Lambda](#) in the Lambda Developer Guide. 6. (Optional) If you choose **Create Lambda function**, then you must configure the AWS CloudFormation	

Task	Description	Skills required
	template that's included by the Java application before deploying that stack. The template includes ApplicationName, SpillBucket, AthenaCatalogName, and other application settings. **Note**: After you deploy this Java-based application, the stack creates a Lambda function that enables Athena to communicate with DynamoDB. This makes your tables accessible through SQL commands. 7. Deploy your Lambda function. 8. Choose **Next**.	

Task	Description	Skills required
Verify that the Lambda function can access the S3 spill bucket.	1. Open the [Lambda console](https://console.aws.amazon.com/lambda/). 2. In the navigation pane, choose **Functions**, and then choose the function that you created earlier. 3. Choose the **Configuration** tab. 4. In the left pane, choose **Environment variables**, and then confirm that the value for the key is `spill_bucket` . 5. In the left pane, choose **Permissions**, and then in the **Execution role** section, choose the attached IAM role. **Note**: You are directed to the IAM role that's attached to your Lambda function in the IAM console. 6. Confirm that you have write permission on `spill_bucket` bucket. If you experience errors, see the *Additional information* section in this pattern for guidance.	Developer

Access DynamoDB tables from Athena

Task	Description	Skills required
Query the DynamoDB tables.	1. Sign in to the AWS Management Console and open the [Athena console](). 2. In the navigation pane, choose **Data sources**, and then choose **Create data source**. 3. In the navigation pane, choose **Query editor**. 4. On the **Editor** tab, in the **Data** section, for **Data source**, choose your data source for **Data source**. 5. For **Database**, choose your database. 6. For Query 1, enter the following query: `SELECT * FROM dydbtable1 t1;` 7. Choose **Run**, and then verify the output in the table. 8. For Query 2, enter the following query: `SELECT * FROM dydbtable2 t2;` 9. Choose **Run**, and then verify the output in the table.	Developer
Join the two DynamoDB tables.	DynamoDB is a NoSQL data store and doesn't support the SQL join operation.	Developer

Task	Description	Skills required
	Consequently, you must perform a join operation on two DynamoDB tables: 1. Choose the plus icon to create another query. 2. For Query 3, enter the following query: `SELECT pk1, salary, bonus FROM dydbtable1 t1 JOIN dydbtable2 t2 ON t1.pk1 = t2.pk2;`	

Related resources

- Amazon Athena DynamoDB Connector (AWS Labs)
- Query any data source with Amazon Athena's new federated query (AWS Big Data Blog)
- Athena engine version reference (Athena User Guide)
- Simplify Amazon DynamoDB data extraction and analysis by using AWS Glue and Amazon Athena (AWS Database Blog)

Additional information

If you run a query in Athena with `spill_bucket` in the `{bucket_name}/folder_name/` format, then you can receive the following error message:

```
"GENERIC_USER_ERROR: Encountered an exception[java.lang.RuntimeException] from your
 LambdaFunction[arn:aws:lambda:us-east-1:xxxxxx:function:testdynamodb] executed in
 context[retrieving meta-data] with message[You do NOT own the spill bucket with the
 name: s3://amzn-s3-demo-bucket/athena_dynamodb_spill_data/]
```

```
This query ran against the "default" database, unless qualified by the query. Please
 post the error message on our forum  or contact customer support with Query Id:
 [query-id]"
```

To resolve this error, update the Lambda function's environment variable `spill_bucket` to `{bucket_name_only}`, and then update the following Lambda IAM policy for bucket write access:

```
{
             "Action": [
                 "s3:GetObject",
                 "s3:ListBucket",
                 "s3:GetBucketLocation",
                 "s3:GetObjectVersion",
                 "s3:PutObject",
                 "s3:PutObjectAcl",
                 "s3:GetLifecycleConfiguration",
                 "s3:PutLifecycleConfiguration",
                 "s3:DeleteObject"
            ],
             "Resource": [
                 "arn:aws:s3:::spill_bucket",
                 "arn:aws:s3:::spill_bucket/*"
            ],
             "Effect": "Allow"
        }
```

Alternatively, you can remove the Athena data source connector that you created earlier, and recreate it by using only `{bucket_name}` for `spill_bucket`.

Set up a minimum viable data space to share data between organizations

Ramy Hcini, Michael Miller, and Jorge Hernandez Suarez, Amazon Web Services

Ismail Abdellaoui and Malte Gasseling, Think-it

Summary

Data spaces are federated networks for data exchange with trust and control over one's data as core principles. They enable organizations to share, exchange, and collaborate on data at scale by offering a cost-effective and technology-agnostic solution.

Data spaces have the potential to significantly drive efforts for a sustainable future by using data-driven problem solving with an end-to-end approach that involves all relevant stakeholders.

This pattern guides you through the example of how two companies can use data space technology on Amazon Web Services (AWS) to drive their carbon emissions–reduction strategy forward. In this scenario, company X provides carbon-emissions data, which company Y consumes. See the [Additional information](#) section for the following data space specification details:

- Participants
- Business case
- Data space authority
- Data space components
- Data space services
- Data to be exchanged
- Data model
- Tractus-X EDC connector

The pattern includes steps for the following:

- Deploying the infrastructure needed for a basic data space with two participants running on AWS.
- Exchanging carbon emissions–intensity data by using the connectors in a secure way.

This pattern deploys a Kubernetes cluster that will host data space connectors and their services through Amazon Elastic Kubernetes Service (Amazon EKS).

The Eclipse Dataspace Components (EDC) control plane and data plane are both deployed on Amazon EKS. The official Tractus-X Helm chart deploys PostgreSQL and HashiCorp Vault services as dependencies.

In addition, the identity service is deployed on Amazon Elastic Compute Cloud (Amazon EC2) to replicate a real-life scenario of a minimum viable data space (MVDS).

Prerequisites and limitations

Prerequisites

- An active AWS account to deploy the infrastructure in your chosen AWS Region
- An AWS Identity and Access Management (IAM) user with access to Amazon S3 that will be used temporarily as a technical user (The EDC connector currently doesn't support the use of roles. We recommend that you create one IAM user specifically for this demo and that this user will have limited permissions associated with it.)
- AWS Command Line Interface (AWS CLI) installed and configured in your chosen AWS Region
- AWS security credentials
- eksctl on your workstation
- Git on your workstation
- kubectl
- Helm
- Postman
- An AWS Certificate Manager (ACM) SSL/TLS certificate
- A DNS name that will point to an Application Load Balancer (the DNS name must be covered by the ACM certificate)
- HashiCorp Vault (For information about using AWS Secrets Manager to manage secrets, see the Additional information section.)

Product versions

- AWS CLI version 2+

- Postman Collection v2.1

Limitations

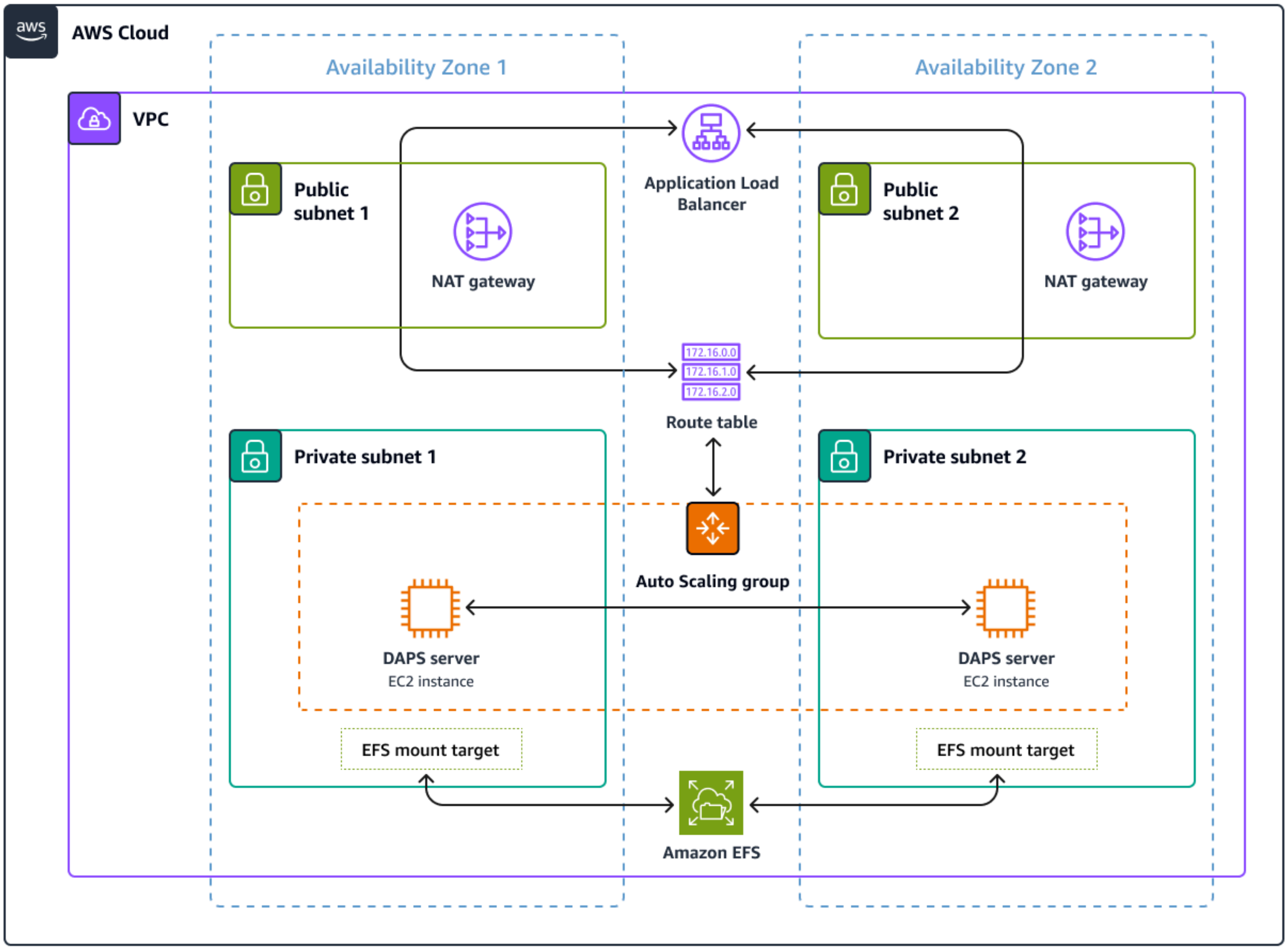
- **Connector selection** – This deployment uses an EDC-based connector. However, be sure to consider the strengths and functionalities of both the EDC and FIWARE True connectors to make an informed decision that aligns with the specific needs of the deployment.
- **EDC connector build** – The chosen deployment solution relies on the Tractus-X EDC Connector Helm chart, a well-established and extensively tested deployment option. The decision to use this chart is driven by its common usage and the inclusion of essential extensions in the provided build. While PostgreSQL and HashiCorp Vault are default components, you have the flexibility to customize your own connector build if needed.
- **Private cluster access** – Access to the deployed EKS cluster is restricted to private channels. Interaction with the cluster is performed exclusively through the use of `kubectl` and IAM. Public access to the cluster resources can be enabled by using load balancers and domain names, which must be implemented selectively to expose specific services to a broader network. However, we do not recommend providing public access.
- **Security focus** – Emphasis is placed on abstracting security configurations to default specifications so that you can concentrate on the steps involved in EDC connector data exchange. Although default security settings are maintained, it's imperative to enable secure communications before you expose the cluster to the public network. This precaution ensures a robust foundation for secure data handling.
- **Infrastructure cost** – An estimation of the infrastructure's cost can be found by using the AWS Pricing Calculator. A simple calculation shows that costs can be up to 162.92 USD per month for the deployed infrastructure.

Architecture

The MVDS architecture comprises two virtual private clouds (VPCs), one for the Dynamic Attribute Provisioning System (DAPS) identity service and one for Amazon EKS.

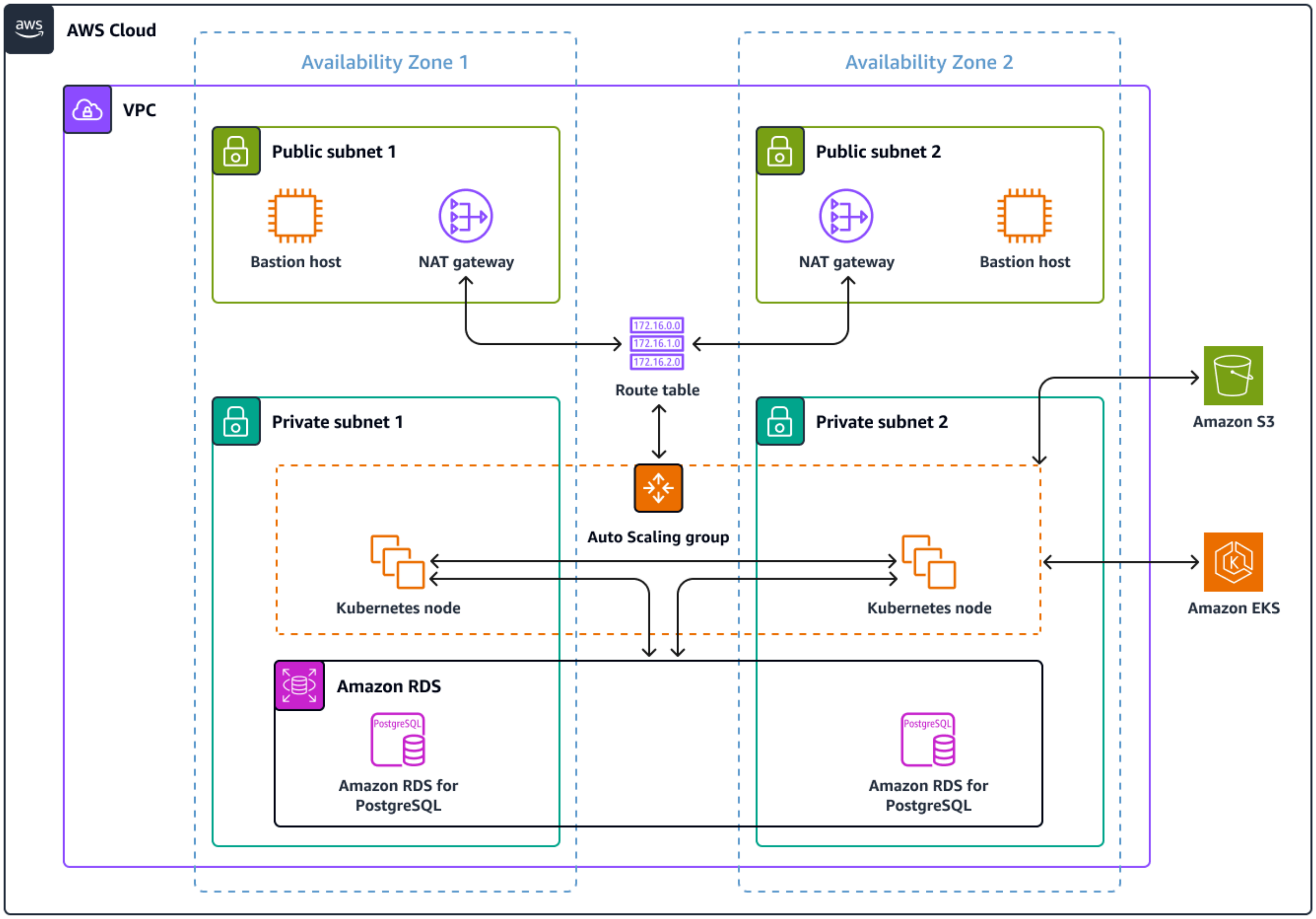
DAPS architecture

The following diagram shows DAPS running on EC2 instances controlled by an Auto Scaling group. An Application Load Balancer and route table expose the DAPS servers. Amazon Elastic File System (Amazon EFS) synchronizes the data among the DAPS instances.



Amazon EKS architecture

Data spaces are designed to be technology-agnostic solutions, and multiple implementations exist. This pattern uses an Amazon EKS cluster to deploy the data space technical components. The following diagram shows the deployment of the EKS cluster. Worker nodes are installed in private subnets. The Kubernetes pods access the Amazon Relational Database Service (Amazon RDS) for PostgreSQL instance that is also in the private subnets. The Kubernetes pods store shared data in Amazon S3.



Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across EC2 instances, containers, and IP addresses in one or more Availability Zones.

Other tools

- eksctl is a command-line utility for creating and managing Kubernetes clusters on Amazon EKS.
- Git is an open source, distributed version control system.
- HashiCorp Vault provides secure storage with controlled access for credentials and other sensitive information.
- Helm is an open source package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.
- kubectl is a command-line interface that helps you run commands against Kubernetes clusters.
- Postman is an API platform.

Code repository

The Kubernetes configuration YAML files and Python scripts for this pattern are available in the GitHub aws-patterns-edc repository. The pattern also uses the Tractus-X EDC repository.

Best practices

Amazon EKS and isolation of participants’ infrastructures

Namespaces in Kubernetes will separate the company X provider’s infrastructure from the company Y consumer’s infrastructure in this pattern. For more information, see EKS Best Practices Guides.

In a more realistic situation, each participant would have a separate Kubernetes cluster running within their own AWS account. The shared infrastructure (DAPS in this pattern) would be accessible by data space participants while being completely separated from participants' infrastructures.

Epics

Set up the environment, and provision an EKS cluster and EC2 instances

Task	Description	Skills required
Clone the repository.	To clone the repository to your workstation, run the following command: `git clone https://github.com/Think-iT-Labs/aws-patterns-edc` The workstation must have access to your AWS account.	DevOps engineer
Provision the Kubernetes cluster and set up namespaces.	To deploy a simplified default EKS cluster in your account, run the following `eksctl` command on the workstation where you cloned the repo: `eksctl create cluster` The command creates the VPC and private and public subnets that span three different Availability Zones. After the network layer is created, the command creates two `m5.large` EC2 instances within an Auto Scaling group. For more information and example output, see the eksctl guide.	DevOps engineer

Task	Description	Skills required

	After you provision the private cluster, add the new EKS cluster to your local Kubernetes configuration by running the following command:
	`aws eks update-kubeconfig --name <EKS CLUSTER NAME> --region <AWS REGION>`
	This pattern uses the `eu-west-1` AWS Region to run all commands. However, you can run the same commands in your preferred AWS Region.
	To confirm that your EKS nodes are running and are in the ready state, run the following command:
	`kubectl get nodes`	

Task	Description	Skills required

Set up the namespaces.	To create namespaces for the provider and the consumer, run the following commands:
	`kubectl create ns provider`
	`kubectl create ns consumer`
	In this pattern, it's important to use `provider` and `consumer` as the namespaces to fit the configurations in the next steps.
		DevOps engineer

Deploy the identity service

Task	Description	Skills required
Deploy DAPS by using AWS CloudFormation.	For ease of managing DAPS operations, the DAPS server is installed on EC2 instances. To install DAPS, use the [AWS CloudFormation template](). You will need the ACM certificate and DNS name from the *Prerequisites* section. The template deploys and configures the following: • Application Load Balancer • Auto Scaling group	DevOps engineer

Task	Description	Skills required
	• EC2 instances configured with user data to install all necessary packages • IAM roles • DAPS You can deploy the AWS CloudFormation template by signing in to the AWS Management Console and using the [AWS CloudFormation console](). You can also deploy the template by using an AWS CLI command such as the following: `aws cloudformation create-stack --stack-name daps \ --template-body file://aws-patterns-edc/cloudformation.yml --parameters \ ParameterKey=CertificateARN,ParameterValue=<ACM Certificate ARN> \ ParameterKey=DNSName,ParameterValue=<DNS name> \ ParameterKey=InstanceType,ParameterValue=<EC2 instance type> \ ParameterKey=EnvironmentName,ParameterValue=<Environment Name> --capabil`	

Task	Description	Skills required
	`ities CAPABILIT Y_NAMED_IAM` The environment name is your own choice. We recommend using a meaningful term, such as `DapsInfrastructure` , because it will be reflected in the AWS resource tags. For this pattern, `t3.small` is large enough to run the DAPS workflow, which has three Docker containers. The template deploys the EC2 instances in private subnets. This means that the instances are not directly accessibl e through SSH (Secure Shell) from the internet. The instances are provisioned with the necessary IAM role and AWS Systems Manager Agent to enable access to the running instances through [Session Manager](), a capability of AWS Systems Manager. We recommend using Session Manager for access. Alternati vely, you could provision a bastion host to allow SSH access from the internet. When using the bastion host	

Task	Description	Skills required

	approach, the EC2 instance might take a few more minutes to start running.
	
	After the AWS CloudFormation template is successfully deployed, point the DNS name to your Application Load Balancer DNS name. To confirm, run the following command:
	
	`dig <DNS NAME>`
	
	The output should be similar to the following:
	
	`; <<>> DiG 9.16.1-Ubuntu <<>> edc-pattern.think-it.io`
	`;; global options: +cmd`
	`;; Got answer:`
	`;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42344`
	`;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1`
	
	`;; OPT PSEUDOSECTION:`
	`; EDNS: version: 0, flags:; udp: 65494`
	`;; QUESTION SECTION:`
	`;edc-pattern.think-it.io.           IN      A`
	
	`;; ANSWER SECTION:`	

Task	Description	Skills required
	``` edc-pattern.think- it.io. 276 IN CNAME daps- alb-iap9zmwy3kn8-13287 73120.eu-west-1.el b.amazonaws.com. daps-alb-iap9zmwy3k n8-1328773120.eu-w est-1.elb.amazonaw s.com. 36 IN A 52.208.240.129 daps-alb-iap9zmwy3kn8 -1328773120.eu-wes t-1.elb.amazonaws. com. 36 IN A 52.210.15 5.124 ```	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Register the participants’ connectors to the DAPS service.</td><td>From within any of the EC2 instances provisioned for DAPS, register participants:

1. Run the available script on the EC2 instance by using the root user:
<code>cd /srv/mvds/omejdn-daps</code>
2. Register the provider:
<code>bash scripts/register_connector.sh <provider_name></code>
3. Register the consumer:
<code>bash scripts/register_connector.sh <consumer_name></code>

The choice of the names doesn't impact the next steps. We recommend using either <code>provider</code> and <code>consumer</code> or <code>companyx</code> and <code>companyy</code>.

The registration commands will also automatically configure the DAPS service with the needed informati on fetched from the created certificates and keys.</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	While you are logged in to a DAPS server, gather information needed for later steps in the installation: 1. From `omejdn-daps/config/clients.yml` get the `client id` for the provider and the consumer. The `client id` values are long strings of hexadecimal digits. 2. From the `omejdn-daps/keys` directory, copy the contents of the `consumer.cert` , `consumer.key` , `provider.cert` , and `provider.key` files. We recommend copying and pasting the text into similarly named files prefixed with `daps-` on your workstation. You should have the client IDs for the provider and consumer and should have four files in your working directory on your workstation: • Source file name `consumer.cert` becomes workstation file name `daps-consumer.cert` .	

Task	Description	Skills required
	• Source file name `consumer.key` becomes workstation file name `daps-consumer.key` . • Source file name `provider.cert` becomes workstation file name `daps-provider.cert` . • Source file name `provider.key` becomes workstation file name `daps-provider.key` .	

Deploy the participants' connectors

Task	Description	Skills required
Clone the Tractus-X EDC repository and use the 0.4.1 version.	The Tractus-X EDC connector 's build requires PostgreSQ L (assets database) and HashiCorp Vault (secrets management) services to be deployed and available. There are many different versions of Tractus-X EDC Helm charts. This pattern specifies version 0.4.1 because it uses the DAPS server. The latest versions use Managed Identity Wallet (MIW) with a distributed	DevOps engineer

Task	Description	Skills required

	implementation of the identity service.
	
	On the workstation where you created the two Kubernetes namespaces, clone the [tractusx-edc repository](), and check out the `release/0.4.1` branch.
	
	`git clone https://github.com/eclipse-tractusx/tractusx-edc`
	
	`cd tractusx-edc`
	
	`git checkout release/0.4.1`	

Task	Description	Skills required

Configure the Tractus-X EDC Helm chart.	Modify the Tractus-X Helm chart template configuration to enable both connectors to interact together.
	
	To do this, you would add the namespace to the DNS name of the service so that it could be resolved by other services in the cluster. These modifications should be made to the `charts/tractusx-connector/templates/_helpers.tpl` file. This pattern provides a [final modified version of this file](#) for you to use. Copy it and put it in the daps section of the file `charts/tractusx-connector/templates/_helpers.tpl` .
	
	Make sure to comment all DAPS dependencies in `charts/tractusx-connector/Chart.yaml` :
	
	`dependencies:`
	`  # IDS Dynamic Attribute Provisioning Service (IAM)`
	`  #  - name: daps`
	`  # version: 0.0.1`
	`  # repository: "file://./subcharts/omejdn"`
	`  # alias: daps`
		DevOps engineer

Task	Description	Skills required
	``` # condition: install.daps```	

Task	Description	Skills required

Configure the connectors to use PostgreSQL on Amazon RDS.	(Optional) Amazon Relational Database Service (Amazon RDS) instance is not required in this pattern. However, we highly recommend using Amazon RDS or Amazon Aurora because they provide features such as high availability and backup and recovery.
	
	To replace PostgreSQL on Kubernetes with Amazon RDS, do the following:
	
	1. Provision the [Amazon RDS for PostgreSQL instance](#).
	2. In `Chart.yaml`, comment the `PostgreSQL` section.
	3. In `provider_values.yml` and `consumer_values.yml`, configure the `postgresql` section as follows:
	
	`postgresql:`
	`  auth:`
	`    database: edc`
	`    password: <RDS PASSWORD>`
	`    username: <RDS Username>`
		DevOps engineer

Task	Description	Skills required

	``
	  jdbcUrl: jdbc:post
	gresql://<RDS DNS
	 NAME>:5432/edc
	  username: <RDS
	 Username>
	  password: <RDS
	 PASSWORD>
	  primary:
	    persistence:
	      enabled: false
	  readReplicas:
	    persistence:
	      enabled: false
	``	

Task	Description	Skills required
Configure and deploy the provider connector and its services.	To configure the provider connector and its services, do the following: 1. To download the `provider_edc.yaml` file from the `edc_helm_configs` directory to the current Helm chart folder, run the following command: `wget -q https://raw.githubusercontent.com/Think-iT-Labs/aws-patterns-edc/main/edc_helm_configs/provider_edc.yaml> -P charts/tractusx-connector/` 2. Replace the following variables (also marked in the file) with their values: • `CLIENT_ID` – The ID generated by the DAPS. The `CLIENT_ID` should be in `/srv/mvds/omejdn-daps/config/clients.yml/config/clients.yml` on the DAPS server. It should a string of hexadecimal characters.	DevOps engineer

Task	Description	Skills required

	`DAPS_URL` – The URL of the DAPS server. It should be `https://{DNS name}` using the DNS name that you set up when you ran the AWS CloudFormation template. `VAULT_TOKEN` – The token to be used for Vault authorization. Choose any value. `vault.fullnameOverride` – `vault-provider`. `vault.hashicorp.url` – `http://vault-provider:8200/`. The previous values assume that the deployment name and the namespace name are provider.
	3. To run the Helm chart from your workstation, use the following commands:
	`cd charts/tractusx-connector`
	
	`helm dependency build`	

Task	Description	Skills required
	```helm upgrade --install provider ./ -f provider_edc.yaml -n provider```	

Task	Description	Skills required

Add the certificate and keys to the provider vault.	To avoid confusion, produce the following certificates outside of the `tractusx-edc/charts` directory.
	
	For example, run the following command to change to your home directory:
	
	`cd ~`
	
	You now need to add the secrets that are needed by the provider into the vault.
	
	The names of the secrets within the vault are the values of the keys in the `secretNames:` section of the `provider_edc.yml` file. By default, they are configured as follows:
	
	`secretNames:`
	`transferProxyTokenSignerPrivateKey: transfer-proxy-token-signer-private-key`
	`transferProxyTokenSignerPublicKey: transfer-proxy-token-signer-public-key`
	`transferProxyTokenEncryptio`
		DevOps engineer

Task	Description	Skills required

	`nAesKey: transfer-proxy-token-encryption-aes-key`
	`dapsPrivateKey: daps-private-key`
	`dapsPublicKey: daps-public-key`
	
	An Advanced Encryption Standard (AES) key, private key, public key, and self-signed certificate are generated initially. These are subsequently added as secrets to the vault.
	
	Furthermore, this directory should contain the `daps-provider.cert` and `daps-provider.key` files that you copied from the DAPS server.
	
	1. Run the following commands:
	
	`# generate a private key`
	`openssl ecparam -name prime256v1 -genkey -noout -out provider-private-key.pem`
	`# generate corresponding public key`
	`openssl ec -in provider-private-key.pem -pubout -out`	

Task	Description	Skills required
	``` provider-public-ke y.pem # create a self-sign ed certificate openssl req -new - x509 -key provider- private-key.pem -out provider-cert.pem - days 360 # generate aes key openssl rand -base64 32 > provider- aes.key ``` 2. Before adding the secrets to the vault, convert them from multiple lines to single lines by replacing line breaks with \n: ``` cat provider-private- key.pem | sed 's/$/\ \\\n/'\|tr -d '\\n' > provider-private-k ey.pem.line cat provider-public- key.pem \| sed 's/$/\ \\\n/'|tr -d '\\n' > provider-public-ke y.pem.line cat provider-cert.pem | sed 's/$/\\\\ n/'\|tr -d '\\n' > provider-cert.pem. line cat provider-aes.key \| sed 's/$/\\\\ n/'|tr -d '\\n' > provider-aes.key.l ine ```	

Task	Description	Skills required

	<pre>## The following block is for daps certificate and key
	openssl x509 -in daps-provider.cert -outform PEM | sed 's/$/\\\\n/'|tr -d '\\n' > daps-provider.cert.line
	cat daps-provider.key | sed 's/$/\\\\n/'| tr -d '\\n' > daps-provider.key.line</pre> 3. To format the secrets that will be added to Vault, run the following commands: <pre>JSONFORMAT='{"content": "%s"}'
	#create a single line in JSON format
	printf "${JSONFORMAT}\\n" "`cat provider-private-key.pem.line`" > provider-private-key.json
	printf "${JSONFORMAT}\\n" "`cat provider-public-key.pem.line`" > provider-public-key.json
	printf "${JSONFORMAT}\\n" "`cat provider-cert.pem.line`" > provider-cert.json
	printf "${JSONFORMAT}\\n" "`cat</pre>	

Task	Description	Skills required
	``` provider-aes.key.l ine`" > provider- aes.json  printf "${JSONFO RMAT}\\n" "`cat daps- provider.key.line`"  > daps-provider.key. json printf "${JSONFO RMAT}\\n" "`cat daps- provider.cert.line`" > daps-provider.cert .json ``` The secrets are now in JSON format and are ready to be added to the vault. 4. To get the pod name for the vault, run the following command: ``` kubectl get pods - n provider|egrep "vault|NAME" ``` The pod name will be similar to `"vault-provider-0"`. This name is used when creating a port forward to the vault. The port forward gives you to access the vault to add the secret. You should run this from a workstation that has AWS credentials configured.	

Task	Description	Skills required
	5. To access the vault, use `kubectl` to configure a port forward: `kubectl port-forward <VAULT_POD_NAME> 8200:8200 -n provider` You should now be able to access the vault through your browser or the CLI. **Browser** 1. Using the browser, navigate to http://127.0.0.1:8200, which will use the port forward that you configured. 2. Log in using the token you that configured previously in `provider_edc.yml` . In the secrets engine, create three secrets. Each secret will have a `Path for this secret` value, which is the secret name shown in the following list. Within the `secret data` section, the name of the key will be `content` and the value will be the single line of text from the respective file named `.line`.	

Task	Description	Skills required
	3. The secret names are sourced from the `secretNames` section in the `provider_edc.yml` file. 4. Create the following secrets: • Secret `transfer-proxy-token-signer-private-key` with file name `provider-private-key.pem.line` • Secret `transfer-proxy-token-signer-public-key` with file name `provider-cert.pem.line` • Secret `transfer-proxy-token-encryption-aes-key` with file name `provider-aes.key.line` • Secret `daps-private-key` with file name `daps-provider.key.line` • Secret `daps-public-key` with file name `daps-provider.cert.line`	

Task	Description	Skills required

	Vault CLI
	
	The CLI will also use the port forward that you configured.
	
	1. On your workstation, install Vault CLI by following the instructions in the [HashiCorp Vault documentation]().
	2. To log in to the vault by using the token that you set up in `provider_edc.yml`, run the following command:
	`vault login -address=http://127.0.0.1:8200`
	With the correct token, you should see the message `"Success! You are now authenticated."`
	3. To create the secrets by using the JSON formatted files that you created previously, run the following code:
	`vault kv put -address=http://127.0.0.1:8200 secret/transfer-proxy-token-signer-private-key @provider-private-key.json`	

Task	Description	Skills required
	```vault kv put -address=http://127.0.0.1:8200 secret/transfer-proxy-token-signer-public-key @provider-cert.json vault kv put -address=http://127.0.0.1:8200 secret/transfer-proxy-token-encryption-aes-key @provider-aes.json vault kv put -address=http://127.0.0.1:8200 secret/daps-private-key @daps-provider.key.json vault kv put -address=http://127.0.0.1:8200 secret/daps-public-key @daps-provider.cert.json```	

Task	Description	Skills required

Configure and deploy the consumer connector and its services.	The steps for configuring and deploying the consumer are similar to those you completed for the provider:
	
	1. To copy the `consumer_edc.yaml` from the [aws-patterns-edc](https://github.com/Think-iT-Labs/aws-patterns-edc) repo into the `tractusx-edc/charts/tractusx-connector` folder, run the following commands:
	
	`cd tractusx-edc`
	
	`wget -q https://raw.githubusercontent.com/Think-iT-Labs/aws-patterns-edc/main/edc_helm_configs/consumer_edc.yaml -P charts/tractusx-connector/`
	
	2. Update the following variables with their actual values:
	
	• `CONSUMER_CLIENT_ID` – The ID generated by DAPS. The `CONSUMER_CLIENT_ID` should be in `config/clients.yml` on the DAPS server.	

Task	Description	Skills required

	• `DAPS_URL` – The same DAPS URL that you used for the provider.
	• `VAULT_TOKEN` – The token to be used for Vault authorization. Choose any value.
	• `vault.fullnameOverride` – `vault-consumer`
	• `vault.hashicorp.url` – `http://vault-provider:8200/`
	
	The previous values assume that the deployment name and the namespace name are `consumer`.
	
	3. To run the Helm chart, use the following commands:
	
	`cd charts/tractusx-connector`
	
	`helm upgrade --install consumer ./ -f consumer_edc.yaml -n consumer`	

Task	Description	Skills required

Add the certificate and keys to the consumer vault.	From a security standpoint, we recommend regenerating the certificates and keys for each data space participant. This pattern regenerates certificates and keys for the consumer.
	
	The steps are very similar to those for the provider. You can verify the secret names in the `consumer_edc.yml` file.
	
	The names of the secrets within the vault are the values of the keys in the `secretNames:` section of the `consumer_edc.yml file`. By default, they are configured as follows:
	
	`secretNames:`
	`    transferProxyTokenSignerPrivateKey: transfer-proxy-token-signer-private-key`
	`    transferProxyTokenSignerPublicKey: transfer-proxy-token-signer-public-key`
	`    transferProxyTokenEncryptionAesKey: transfer-proxy-token-encryption-aes-key`
		DevOps engineer

Task	Description	Skills required

	``
	                 dapsPriva
	teKey: daps-private-
	key
	                 dapsPubli
	cKey: daps-public-key
	``
	
	The `daps-consumer.cert` and `daps-consumer.key` files that you copied from the DAPS server should already exist in this directory.
	
	1. Run the following commands:
	
	``
	# generate a private key
	openssl ecparam -name prime256v1 -genkey -noout -out consumer-private-key.pem
	# generate corresponding public key
	openssl ec -in consumer-private-key.pem -pubout -out consumer-public-key.pem
	# create a self-signed certificate
	openssl req -new -x509 -key consumer-private-key.pem -out consumer-cert.pem -days 360
	# generate aes key
	openssl rand -base64 32 > consumer-aes.key
	``	

Task	Description	Skills required

	2. Manually edit the files to replace line breaks with \n, or use three commands similar to the following:
	
	`cat consumer-private-key.pem | sed 's/$/\\\\n/'|tr -d '\\n' > consumer-private-key.pem.line`
	`cat consumer-public-key.pem | sed 's/$/\\\\n/'|tr -d '\\n' > consumer-public-key.pem.line`
	`cat consumer-cert.pem | sed 's/$/\\\\n/'|tr -d '\\n' > consumer-cert.pem.line`
	`cat consumer-aes.key | sed 's/$/\\\\n/'|tr -d '\\n' > consumer-aes.key.line`
	`cat daps-consumer.cert | sed 's/$/\\\\n/'|tr -d '\\n' > daps-consumer.cert.line`
	`cat daps-consumer.key | sed 's/$/\\\\n/'|tr -d '\\n' > daps-consumer.key.line`
	
	3. To format the secrets that will be added to Vault, run the following commands:	

Task	Description	Skills required
	```JSONFORMAT='{"content": "%s"}' #create a single line in JSON format printf "${JSONFORMAT}\\n" "`cat consumer-private-key.pem.line`" > consumer-private-key.json printf "${JSONFORMAT}\\n" "`cat consumer-public-key.pem.line`" > consumer-public-key.json printf "${JSONFORMAT}\\n" "`cat consumer-cert.pem.line`" > consumer-cert.json printf "${JSONFORMAT}\\n" "`cat consumer-aes.key.line`" > consumer-aes.json printf "${JSONFORMAT}\\n" "`cat daps-consumer.key.line`" > daps-consumer.key.json printf "${JSONFORMAT}\\n" "`cat daps-consumer.cert.line`" > daps-consumer.cert.json```	

Task	Description	Skills required

	The secrets are now in JSON format and are ready to be added to the vault.
	4. To get the pod name for the consumer vault, run the following command:
	`kubectl get pods -n consumer | egrep "vault|NAME"`
	The pod name will be similar to `"vault-consumer-0"` . This name is used when creating a port forward to the vault. The port forward gives you to access the vault to add the secret. You should run this from a workstation that has AWS credentials configured.
	5. To access the vault, use `kubectl` to configure a port forward:
	`kubectl port-forward <VAULT_POD_NAME> 8201:8200 -n consumer`
	The local port is 8201 this time so that you can have port forwards in place for	

Task	Description	Skills required
	both the producer and consumer. **Browser** You can use your browser to connect to http://localhost:8201/ to access the consumer vault and create the secrets with names and content as outlined. The secrets and files that contain the content are the following: • Secret `transfer-proxy-token-signer-private-key` with file name `consumer-private-key.pem.line` • Secret `transfer-proxy-token-signer-public-key` with file name `consumer-cert.pem.line` • Secret `transfer-proxy-token-encryption-aes-key` with file name `consumer-aes.key.line` **Vault CLI**	

Task	Description	Skills required

	Using Vault CLI, you can run the following commands to log in to the vault and create the secrets:
	
	1. Log in to the vault by using the token that you configured within `consumer_edc.yml` :
	
	`vault login -address=http://127.0.0.1:8201`
	
	With the correct token, you should see the message `"Success! You are now authenticated."`
	2. To create the secrets using the JSON formatted files that you created previously, run the following code:
	
	`vault kv put -address=http://127.0.0.1:8201 secret/transfer-proxy-token-signer-private-key @consumer-private-key.json`
	`vault kv put -address=http://127.0.0.1:8201 secret/transfer-proxy-token-signer-public-key @consumer-cert.json`
	`vault kv put -address=http://127.0.0.1:8201 secret/transfer-`	

Task	Description	Skills required
	```proxy-token-encryption-aes-key @consumer-aes.json vault kv put -address=http://127.0.0.1:8201 secret/daps-private-key @daps-consumer.key.json vault kv put -address=http://127.0.0.1:8201 secret/daps-public-key @daps-consumer.cert.json```	

Set up an HTTP client to interact with the management API of the connectors

Task	Description	Skills required
Set up port forwarding.	1. To check the status of the pods, run the following commands: ```kubectl get pods -n provider kubectl get pods -n consumer``` 2. To make sure that the Kubernetes deployments were successful, look at the logs of the provider and consumer Kubernetes pods by running the following commands:	DevOps engineer

Task	Description	Skills required

	`kubectl logs -n provider <producer control plane pod name>`
	`kubectl logs -n consumer <consumer control plane pod name>`
	
	The cluster is private and is not accessible publicly. To interact with the connector s, use the Kubernetes port-forwarding feature to forward traffic generated by your machine to the connector control plane.
	
	1. On the first terminal, forward the consumer' s requests to the management API through port 8300:
	`kubectl port-forward deployment/consumer-tractusx-connector-controlplane 8300:8081 -n consumer`
	2. On the second terminal, forward the provider' s requests to the management API through port 8400:	

Task	Description	Skills required
	```kubectl port-forward deployment/provider-tractusx-connector-controlplane 8400:8081 -n provider```	

Task	Description	Skills required

Create S3 buckets for the provider and the consumer.	The EDC connector currently doesn't use temporary AWS credentials, such as those provided by assuming a role. The EDC supports only the use of an [IAM access key ID and secret access key combination]().
	
	 Two S3 buckets are required for later steps. One S3 bucket is used for storing data made available by the provider. The other S3 bucket is for data received by the consumer.
	
	 The IAM user should have permission to read and write objects only in the two named buckets.
	
	 An access key ID and secret access key pair needs to be created and kept safe. After this MVDS has been decommissioned, the IAM user should be deleted.
	
	 The following code is an example IAM policy for the user:
	
	 `{`
	 `  "Version": "2012-10-17",`
	 `  "Statement": [`
	 `    {`
		DevOps engineer

Task	Description	Skills required

```
      "Sid": "Stmt1708
699805237",
      "Action": [
        "s3:GetObject",
        "s3:GetOb
jectVersion",
        "s3:ListA
llMyBuckets",
        "s3:ListB
ucket",
        "s3:ListB
ucketMultipartUplo
ads",
        "s3:ListB
ucketVersions",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:
s3:::<S3 Provider
 Bucket>",
        "arn:aws:
s3:::<S3 Consumer
 Bucket>",
        "arn:aws:
s3:::<S3 Provider
 Bucket>/*",
        "arn:aws:
s3:::<S3 Consumer
 Bucket>/*"
        ]    }
  ]
}
```

Task	Description	Skills required
Set up Postman to interact with the connector.	You can now interact with the connectors through your EC2 instance. Use Postman as an HTTP client, and provide Postman Collections for both the provider and the consumer connectors. Import the collections from the `aws-pattern-edc` repository into your Postman instance. This pattern uses Postman collection variables to provide input to your requests.	App developer, Data engineer

Provide company X carbon-emissions footprint data through the connector

Task	Description	Skills required
Prepare the carbon-emissions intensity data to be shared.	First you need to decide on the data asset to be shared. The data of company X represents the carbon-em issions footprint of its vehicle fleet. Weight is Gross Vehicle Weight (GVW) in tonnes, and emissions are in grams of CO2 per tonne-kilometer (g CO2 e/t-km) according to the Wheel-to-Well (WTW) measurement:	Data engineer, App developer

Task	Description	Skills required
	- Vehicle type: Van; weight: < 3.5; emissions: 800 - Vehicle type: Urban truck; weight: 3.5–7.5; emissions: 315 - Vehicle type: Medium goods vehicle (MGV); weight: 7.5–20; emissions: 195 - Vehicle type: Heavy goods vehicle (HGV); weight: > 20; emissions: 115 The example data is in the `carbon_emissions_data.json` file in the `aws-patterns-edc` repository. Company X uses Amazon S3 to store objects. Create the S3 bucket and store the example data object there. The following commands create an S3 bucket with default security settings. We highly recommend consulting [Security best practices for Amazon S3](). `aws s3api create-bucket <BUCKET_NAME> --region <AWS_REGION>`	

Task	Description	Skills required
	``` # You need to add '--create-bucket-configuration # LocationConstraint=<AWS_REGION>' if you want to create # the bucket outside of us-east-1 region aws s3api put-object --bucket <BUCKET_NAME> \ --key <S3 OBJECT NAME> \ --body <PATH OF THE FILE TO UPLOAD> ``` The S3 bucket name should be globally unique. For more information about naming rules, see the AWS documentation.	

Task	Description	Skills required

Register the data asset to the provider's connector by using Postman.	An EDC connector data asset holds the name of the data and its location. In this case, the EDC connector data asset will point to the created object in the S3 bucket:
	
	• **Connector:** Provider
	• **Request:** Create Asset
	• **Collection Variables:** Update `ASSET_NAME` . Choose a meaningful name that represents the asset.
	• **Request Body:** Update the request body with the S3 bucket that you created for the provider.
	
	`"dataAddress": {`
	`    "edc:type": "AmazonS3",`
	`    "name": "Vehicle Carbon Footprint",`
	`    "bucketName": "<REPLACE WITH THE SOURCE BUCKET NAME>",`
	`    "keyName": "<REPLACE WITH YOUR OBJECT NAME>",`
	`    "region": "<REPLACE WITH THE BUCKET REGION>",`
	`    "accessKeyId": "<REPLACE WITH YOUR ACCESS KEY ID>",`
	`    "secretAccessKey": "<REPLACE`
		App developer, Data engineer

Task	Description	Skills required

	``
	 WITH SECRET ACCESS
	 KEY>"
	}
	``
	- **Response:** A successful request returns the created time and the asset ID of the newly created asset.
	``
	{
	 "@id": "c89aa31c-ec4c-44ed-9e8c-1647f19d7583"
	}
	``
	- **Collection variable ASSET_ID:** Update the Postman collection variable `ASSET_ID` with the ID that was generated automatically by the EDC connector after creation.	

Task	Description	Skills required

Define the usage policy of the asset.	An EDC data asset must be associated with clear usage policies. First, create the Policy Definition in the provider connector.
	
	The policy of company X is to allow participants of the data space to use the carbon-emissions footprint data.
	
	• **Request Body:**
	  ◦ **Connector:** Provider
	  ◦ **Request:** Create Policy
	  ◦ **Collection Variables:** Update the `Policy Name` variable with the name of the policy.
	• **Response:** A successful request returns the created time and the policy ID of the newly created policy. Update the collection variable `POLICY_ID` with the ID of the policy generated by the EDC connector after creation.
		App developer, Data engineer

Task	Description	Skills required
Define an EDC Contract Offer for the asset and its usage policy.	To allow other participants to request access to your data, offer it in a contract that specifies the usage conditions and permissions: • **Connector:** Provider • **Request:** Create Contract Definition • **Collection Variables:** Update the `Contract Name` variable with a name for the contract offer or definition.	App developer, Data engineer

Discover the assets and reach agreement on the defined contracts

Task	Description	Skills required
Request the data catalog shared by company X.	As a data consumer in the data space, company Y first needs to discover the data that is being shared by other participants. In this basic setup, you can do this by asking the consumer connector to request the catalog of available assets from the provider connector directly. • **Connector:** Consumer • **Request:** Request Catalog	App developer, Data engineer

Task	Description	Skills required

	• **Response:** All available data assets from the provider together with their attached usage policies. As a data consumer, look for the contract of your interest and update the following collection variables accordingly.
	  • `CONTRACT_OFFER_ID` – The ID of the contract offer the consumer wants to negotiate
	  • `ASSET_ID` – The ID of the asset the consumer wants to negotiate
	  • `PROVIDER_CLIENT_ID` – The ID of the provider connector to negotiate with	

Task	Description	Skills required
Initiate a contract negotiation for the carbon-emissions intensity data from company X.	Now that you have identified the asset that you want to consume, initiate a contract negotiation process between the consumer and provider connectors. • **Connector:** Consumer • **Request:** Contract Negotiation • **Collection Variables:** Update the `CONSUMER_CLIENT_ID` variable with the ID of the consumer connector to negotiate with. The process might take some time before reaching the VERIFIED state. You can check the state of the Contract Negotiation and the corresponding Agreement ID by using the `Get Negotiation` request.	App developer, Data engineer

Consume the data by using the contract agreement

Task	Description	Skills required
Consume data from HTTP endpoints.	(Option 1) To use HTTP data plane to consume data in the data space, you can use	App developer, Data engineer

Task	Description	Skills required
	webhook.site to emulate an HTTP server, and initiate the transfer process in the consumer connector: • **Connector:** Consumer • **Request:** Contract Negotiation • **Collection Variables:** Update the `Contract Agreement ID` variable with the ID of the contract agreement generated by the EDC connector. • **Request Body:** Update the request body to specify HTTP as a `dataDestination` alongside the webhook URL: `{ "dataDestination": { "type": "HttpProxy" }, "privateProperties": { "receiverHttpEndpoint": "<WEBHOOK URL>" } }` The connector will send the information necessary to	

Task	Description	Skills required
	download the file directly to the webhook URL. The received payload is similar to the following: `{ "id": "dcc90391-3819-4b54-b401-1a005a029b78", "endpoint": "http://consumer-tractusx-connector-dataplane.consumer:8081/api/public", "authKey": "Authorization", "authCode": "<AUTH CODE YOU RECEIVE IN THE ENDPOINT>", "properties": { "https://w3id.org/edc/v0.0.1/ns/cid": "vehicle-carbon-footprint-contract:4563abf7-5dc7-4c28-bc3d-97f45e32edac:b073669b-db20-4c83-82df-46b583c4c062" } }` Use the received credentials to get the S3 asset that was shared by the provider.	
	In this last step, you must send the request to the	

Task	Description	Skills required
	consumer data plane (forward ports properly), as stated in the payload (`endpoint`).	

Task	Description	Skills required
Consume data from S3 buckets directly.	(Option 2) Use Amazon S3 integration with the EDC connector, and directly point to the S3 bucket in the consumer infrastructure as a destination: • **Request Body:** Update the request body to specify the S3 bucket as a dataDesti nation. This should be the S3 bucket that you previousl y created for storing data received by the consumer. `{ "dataDestination": { "type": "AmazonS3", "bucketName": "{{ REPLACE WITH THE DESTINATION BUCKET NAME }}", "keyName": "{{ REPLACE WITH YOUR OBJECT NAME }}", "region": "{{ REPLACE WITH THE BUCKET REGION }}", "accessKeyId": "{{ REPLACE WITH YOUR ACCESS KEY ID }}", "secretAccessKey": "{{ REPLACE WITH SECRET ACCESS KEY }}"`	Data engineer, App developer

Task	Description	Skills required

	``
	    }
	  }
	}
	``	

Troubleshooting

Issue	Solution
The connector might raise an issue about the certificate PEM format.	Concatenate the contents of each file to a single line by adding \n.

Related resources

- DSSC
- Building data spaces for sustainability use cases (AWS Prescriptive Guidance strategy by Think-it)
- AWS for data spaces
- Tractus-X documentation
- DAPS
- Enabling data sharing through data spaces and AWS (blog post)

Additional information

Data space specifications

Participants

Participant	Description of the company	Focus of the company
Company X	Operates a fleet of vehicles across Europe and South America to transport various goods.	Aims to make data-driv en decisions to reduce its carbon-emissions footprint intensity.

Company Y	An environmental regulatory authority	Enforces environmental regulations and policies designed to monitor and mitigate the environmental impact of businesses and industries, including carbon-emissions intensity.

Business case

Company X uses data space technology to share carbon footprint data with a compliance auditor, company Y, to evaluate and address the environmental impact of company X's logistics operations.

Data space authority

The data space authority is a consortium of the organizations governing the data space. In this pattern, both company X and company Y form the governance body and represent a federated data space authority.

Data space components

Component	Chosen implementation	Additional information
Dataset exchange protocol	Dataspace Protocol version 0.8	• JSON-LD • Data Catalog Vocabulary (DCAT)
Data space connector	Tractus-X EDC Connector version 0.4.1	• EDC extensions
Data exchange policies	Default USE Policy	• Open Digital Rights Language (ODRL)

Data space services

Service	Implementation	Additional information

Identity service	[Dynamic Attribute Provisioning System (DAPS)]()	"A Dynamic Attribute Provisioning System (DAPS) has the intent to ascertain certain attributes to organizations and connectors. Hence, third parties do not need to trust the latter **provided they trust the DAPS assertions**." — DAPS To focus on the connector's logic, the data space is deployed on an Amazon EC2 machine using Docker Compose.
Discovery service	[Gaia-X Federated Catalogue]()	"The Federated Catalogue constitutes an indexed repository of Gaia-X Self-Descriptions to enable the discovery and selection of Providers and their service offerings. The Self-Descriptions are the information given by Participants about themselves and about their services in the form of properties and claims." — Gaia-X Ecosystem Kickstarter

Data to be exchanged

Data assets	Description	Format
Carbon emissions data	Intensity values for different vehicle types in the specified	JSON file

	region (Europe and South America) from the entire fleet of vehicles

Data model

```
{
  "region": "string",
  "vehicles": [
// Each vehicle type has its Gross Vehicle Weight (GVW) category and its emission
 intensity in grams of CO2 per Tonne-Kilometer (g CO2 e/t-km) according to the "Well-
to-Wheel" (WTW) measurement.
    {
      "type": "string",
      "gross_vehicle_weight": "string",
      "emission_intensity": {
        "CO2": "number",
        "unit": "string"
      }
    }
  ]
}
```

Tractus-X EDC connector

For documentation of each Tractus-X EDC parameter, see the original values file.

The following table lists all of services, along with their corresponding exposed ports and endpoints for reference.

Service name	Port and path
Control plane	• **management**: – Port: 8081 Path: `/management` • **control** – Port: 8083 Path: `/control` • **protocol** Port: 8084 Path: `/api/v1/dsp` • **metrics** – Port: 9090 Path: `/metrics`

	• **observability** – Port: 8085 Path: `/observability`
Data plane	**default** – Port: 8080 Path: `/api` **public** – Port: 8081 Path: `/api/dataplane/control` **proxy** – Port: 8186 Path: `/proxy` **metrics** – Port: 9090 Path: `/metrics` **observability** – Port: 8085 Path: `/observability`
Vault	Port: 8200
PostgreSQL	Port: 5432

Using AWS Secrets Manager Manager

It's possible to use Secrets Manager instead of HashiCorp Vault as the secrets manager. To do so you, must use or build the AWS Secrets Manager EDC extension.

You will be responsible for creating and maintaining your own image, because Tractus-X doesn't provide support for Secrets Manager.

To accomplish that, you need to modify the build Gradle files of both the control plane and the data plane of the connector by introducing your AWS Secrets Manager EDC extension (see this maven artifact for an example), then build, maintain, and reference the Docker image.

For more insights on refactoring the Tractus-X connector Docker image, see Refactor Tractus-X EDC Helm charts.

For simplicity purposes, we avoid to rebuilding the connector image in this pattern and use HashiCorp Vault.

Set up language-specific sorting for Amazon Redshift query results using a scalar Python UDF

Ethan Stark, Amazon Web Services

Summary

This pattern provides steps and sample code for using a scalar Python UDF (user-defined function) to set up case insensitive linguistic sorting for Amazon Redshift query results. It's necessary to use a scalar Python UDF because Amazon Redshift returns results based on binary UTF-8 ordering and doesn't support language-specific sorting. A Python UDF is non-SQL processing code that's based on a Python 2.7 program and runs in a data warehouse. You can run Python UDF code with a SQL statement in a single query. For more information, see the Introduction to Python UDFs in Amazon Redshift AWS Big Data Blog post.

The sample data in this pattern is based on the Turkish alphabet for demonstration purposes. The scalar Python UDF in this pattern is built to make the default query results of Amazon Redshift conform to the linguistic ordering of characters in the Turkish language. For more information, see *Turkish language example* in the *Additional information* section of this pattern. You can modify the scalar Python UDF in this pattern for other languages.

Prerequisites and limitations

Prerequisites

- Amazon Redshift cluster with a database, schema, and tables
- Amazon Redshift user with CREATE TABLE and CREATE FUNCTION permissions
- Python 2.7 or later

Limitations

The linguistic sorting used by the queries in this pattern is case insensitive.

Architecture

Technology stack

- Amazon Redshift

- Python UDF

Tools

AWS services

- Amazon Redshift is a managed petabyte-scale data warehouse service in the AWS Cloud. Amazon Redshift is integrated with your data lake, which enables you to use your data to acquire new insights for your business and customers.

Other tools

- Python (UDFs) user-defined functions are functions that you can write in Python and then call in SQL statements.

Epics

Develop code to sort query results in linguistic order

Task	Description	Skills required

Create a table for your sample data.	To create a table in Amazon Redshift and insert your sample data into the table, use the following SQL statements:
	``
	CREATE TABLE my_table
	 (first_name varchar(30));
	
	INSERT INTO my_table
	 (first_name)
	VALUES
	    ('ali'),
	    ('Ali'),
	    ('ırmak'),
	    ('IRMAK'),
	    ('irem'),
	``
		Data engineer

Task	Description	Skills required

	``
	    ('İREM'),
	    ('oğuz'),
	    ('OĞUZ'),
	    ('ömer'),
	    ('ÖMER'),
	    ('sedat'),
	    ('SEDAT'),
	    ('şule'),
	``
	
	Note
	
	The first names in the sample data include special character s from the Turkish alphabet. For more information about Turkish language considerations for this example, see *Turkish language example* in the *Additional information* section of this pattern.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Check the default sorting of the sample data.</td><td>To see the default sorting of your sample data in Amazon Redshift, run the following query:
<pre>SELECT first_name
 FROM my_table ORDER BY
 first_name;</pre>The query returns the list of first names from the table that you created earlier:
<pre>first_name

Ali
IRMAK
OĞUZ
SEDAT
ali
irem
oğuz
sedat
ÖMER
ömer
İREM
ırmak
ŞULE
şule</pre>The query results aren’t in the correct order because the default binary UTF-8 ordering doesn’t accommodate the linguistic ordering of the Turkish special characters.</td><td>Data engineer</td></tr>
</table>

Task	Description	Skills required

Create a scalar Python UDF.	To create a scalar Python UDF, use the following SQL code:
	<pre>CREATE OR REPLACE
	 FUNCTION collate_sort
	 (value varchar)
	RETURNS varchar
	IMMUTABLE
	AS
	$$
	    def sort_str(val):
	        import string
	
	        dictionary = {
	            'I': 'ı',
	            'ı': 'h~',
	            'İ': 'i',
	            'Ş': 's~',
	            'ş': 's~',
	            'Ğ': 'g~',
	            'ğ': 'g~',
	            'Ü': 'u~',
	            'ü': 'u~',
	            'Ö': 'o~',
	            'ö': 'o~',
	            'Ç': 'c~',
	            'ç': 'c~'
	        }
	
	        for key, value
	 in dictionary.items()
	:
	            val =
	 val.replace(key,
	 value)
	
	        return val.lower
	()
	
	    return sort_str(
	value)</pre>
		Data engineer

Task	Description	Skills required
	``` $$ LANGUAGE plpythonu; ```	

Query the sample data.	To query the sample data by using the Python UDF, run the following SQL query:
	`SELECT first_name FROM my_table ORDER BY collate_order(first_name);`
	The query now returns the sample data in Turkish linguistic order:
	`first_name`
	`---------------`
	`ali`
	`Ali`
	`ırmak`
	`IRMAK`
	`irem`
	`İREM`
	`oğuz`
	`OĞUZ`
	`ömer`
	`Ömer`
	`sedat`
	`SEDAT`
	`şule`
	`ŞULE`
		Data engineer

Related resources

- ORDER BY clause (Amazon Redshift documentation)
- Creating a scalar Python UDF (Amazon Redshift documentation)

Additional information

Turkish language example

Amazon Redshift returns query results based on binary UTF-8 sort ordering, not language-specific sort ordering. This means that if you query an Amazon Redshift table containing Turkish characters, then the query results aren't sorted according to the linguistic ordering of the Turkish language. The Turkish language contains six special characters (ç, ı, ğ, ö, ş, and ü) that don't appear in the Latin alphabet. These special characters are placed at the end of a sorted result set based on binary UTF-8 ordering, as the following table shows.

Binary UTF-8 ordering	Turkish linguistic ordering
a	a
b	b
c	c
d	**ç (*)**
e	d
f	e
g	f
h	g
i	**ğ (*)**
j	h
k	**ı (*)**
l	i
m	j
n	k
o	l

p	m
r	n
s	o
t	**ö (*)**
u	p
v	r
y	s
z	**ş (*)**
ç (*)	t
ğ (*)	u
ı (*)	**ü (*)**
ö (*)	v
ş (*)	y
ü (*)	z

Note

The asterisk (*) indicates a special character in the Turkish language.

As the table above illustrates, special character **ç** is between **c** and **d** in Turkish linguistic ordering, but appears after **z** in binary UTF-8 ordering. The scalar Python UDF in this pattern uses the following character replacement dictionary to replace the Turkish special characters with corresponding Latin-equivalent characters.

Turkish special character	Latin-equivalent character

ç	c~
ı	h~
ğ	g~
ö	o~
ş	s~
ü	u~

Note

A tilde (~) character is appended to the end of the Latin characters that replace their corresponding Turkish special characters.

Modify a scalar Python UDF function

To modify the scalar Python UDF function from this pattern so that the function accepts a locate parameter and supports a multiple transaction dictionary, use the following SQL code:

```
CREATE OR REPLACE FUNCTION collate_sort (value varchar, locale varchar)
RETURNS varchar
IMMUTABLE
AS
$$
    def sort_str(val):
       import string
       # Turkish Dictionary
       if locale == 'tr-TR':
            dictionary = {
               'I': 'ı',
               'ı': 'h~',
               'İ': 'i',
               'Ş': 's~',
               'ş': 's~',
               'Ğ': 'g~',
               'ğ': 'g~',
               'Ü': 'u~',
```

```
                'ü': 'u~',
                'Ö': 'o~',
                'ö': 'o~',
                'Ç': 'c~',
                'ç': 'c~'
        }
        # German Dictionary
        if locale == 'de-DE':
            dictionary = {
                ....
                ....
        }

        for key, value in dictionary.items():
            val = val.replace(key, value)

        return val.lower()

    return sort_str(value)

$$ LANGUAGE plpythonu;
```

The following example code shows how to query the modified Python UDF:

```
SELECT first_name FROM my_table ORDER BY collate_order(first_name, 'tr-TR');
```

Subscribe a Lambda function to event notifications from S3 buckets in different AWS Regions

Suresh Konathala, Amazon Web Services

Andrew Preston and Arindom Sarkar, None

Summary

Amazon Simple Storage Service (Amazon S3) Event Notifications publishes notifications for certain events in your S3 bucket (for example, object created events, object removal events, or restore object events). You can use an AWS Lambda function to process these notifications according to your application's requirements. However, the Lambda function can't directly subscribe to notifications from S3 buckets that are hosted in different AWS Regions.

This pattern's approach deploys a fanout scenario to process Amazon S3 notifications from cross-Region S3 buckets by using an Amazon Simple Notification Service (Amazon SNS) topic for each Region. These Regional SNS topics send the Amazon S3 event notifications to an Amazon Simple Queue Service (Amazon SQS) queue in a central Region that also contains your Lambda function. The Lambda function subscribes to this SQS queue and processes the event notifications according to your organization's requirements.

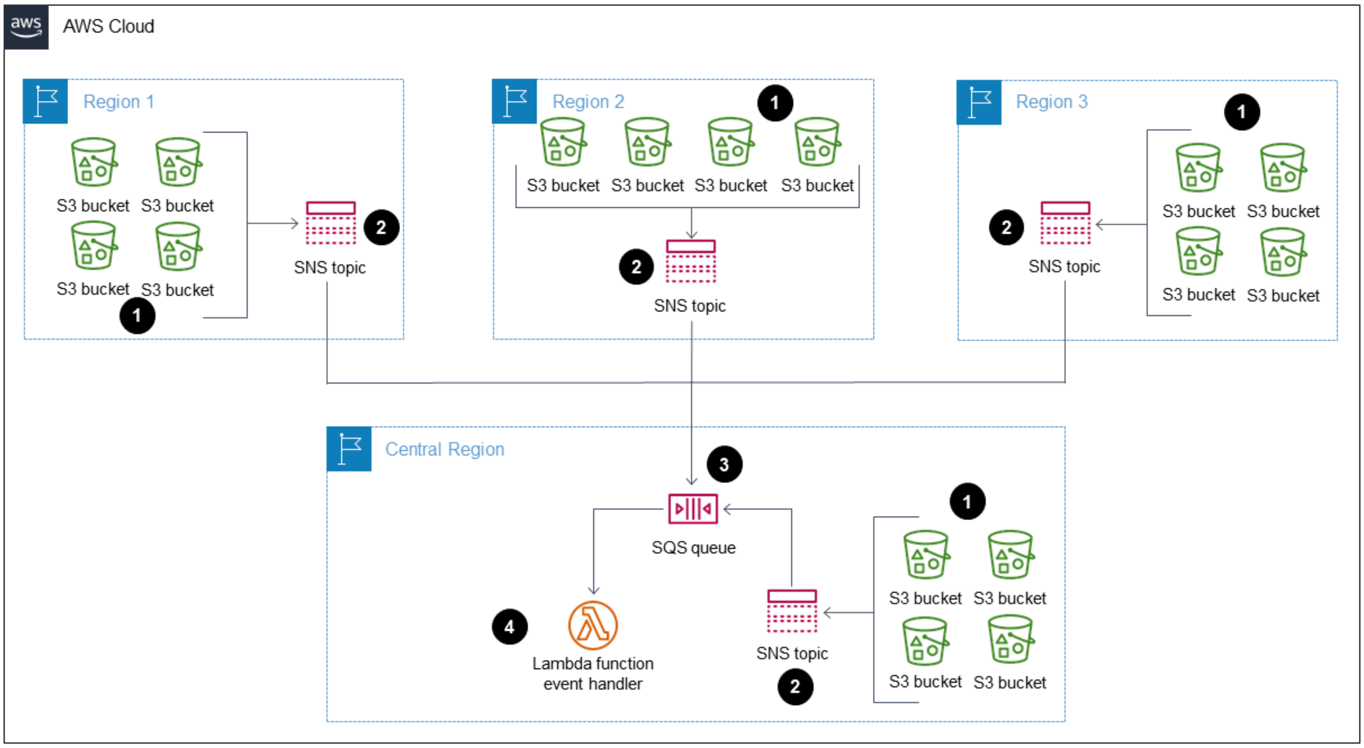
Prerequisites and limitations

Prerequisites

- An active AWS account.
- Existing S3 buckets in multiple Regions, including a central Region to host the Amazon SQS queue and Lambda function.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
- Familiarity with the fanout scenario in Amazon SNS. For more information about this, see Common Amazon SNS scenarios in the Amazon SNS documentation.

Architecture

The following diagram shows the architecture for this pattern's approach.



The diagram shows the following workflow:

1. Amazon S3 sends event notifications about S3 buckets (for example, object created, object removed, or object restored) to an SNS topic in the same Region.
2. The SNS topic publishes the event to an SQS queue in the central Region.
3. The SQS queue is configured as the event source for your Lambda function and buffers the event messages for the Lambda function.
4. The Lambda function polls the SQS queue for messages and processes the Amazon S3 event notifications according to your application's requirements.

Technology stack

- Lambda
- Amazon SNS
- Amazon SQS
- Amazon S3

Tools

- AWS CLI – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.
- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.
- Amazon SQS – Amazon Simple Queue Service (Amazon SQS) offers a secure, durable, and available hosted queue that lets you integrate and decouple distributed software systems and components. Amazon SQS supports both standard and FIFO queues.

Epics

Create the SQS queue and Lambda function in your central Region

Task	Description	Skills required
Create an SQS queue with a Lambda trigger.	Sign in to the AWS Management Console and use the instructions from the tutorial Using Lambda with Amazon SQS in the AWS Lambda documentation to	AWS DevOps, Cloud architect

Task	Description	Skills required
	create the following resources in your central Region: • A Lambda execution role • A Lambda function to process the Amazon S3 events • An SQS queue **Note** Make sure that you configure the SQS queue as the event source for your Lambda function.	

Create an SNS topic and set up event notifications for the S3 buckets in each required Region

Task	Description	Skills required
Create an SNS topic to receive Amazon S3 event notificat ions.	Create an SNS topic in a Region that you want to receive Amazon S3 event notifications from. For more information about this, see [Creating an SNS topic]() in the Amazon SNS documentation. **Important** Make sure that you record your SNS topic's Amazon	AWS DevOps, Cloud architect

Task	Description	Skills required
	Resource Name (ARN).	
Subscribe the SNS topic to the central SQS queue.	Subscribe your SNS topic to the SQS queue hosted by your central Region. For more information about this, see Subscribing to an SNS topic in the Amazon SNS documentation.	AWS DevOps, Cloud architect

Task	Description	Skills required

Update the SNS topic's access policy.	1. Open the Amazon SNS console, choose **Topics**, and then choose the SNS topic that you created earlier.
	2. Choose **Edit** and then expand the **Access policy - optional** section.
	3. Attach the following access policy to your SNS topic to allow `sns:publish` permission for Amazon S3 and then choose **Save**:
	<pre>{
	  "Version": "2012-10-17",
	  "Statement": [
	    {
	      "Sid": "0",
	      "Effect": "Allow",
	      "Principal": {
	        "Service": "s3.amazonaws.com"
	      },
	      "Action": "sns:Publish",
	      "Resource": "arn:aws:sns:us-west-2::s3Events-SNSTopic-us-west-2"
	    }
	  ]
	}</pre>
		AWS DevOps, Cloud architect

Task	Description	Skills required
Set up notifications for each S3 bucket in the Region.	Set up event notifications for each S3 bucket in the Region. For more information about this, see Enabling and configuring event notifications using the Amazon S3 console in the Amazon S3 documentation. **Note** In the **Destination** section, choose **SNS topic** and specify the ARN of the SNS topic that you created earlier.	AWS DevOps, Cloud architect
Repeat this epic for all required Regions.	**Important** Repeat the tasks in this epic for each Region that you want to receive Amazon S3 event notifications from, including your central Region.	AWS DevOps, Cloud architect

Related resources

- Configuring an access policy (Amazon SQS documentation)
- Configuring an SQS queue as an event source (AWS Lambda documentation)
- Configuring an SQS queue to initiate a Lambda function (Amazon SQS documentation)

- AWS::Lambda::Function resource (AWS CloudFormation documentation)

Three AWS Glue ETL job types for converting data to Apache Parquet

Adnan Alvee, Nith Govindasivan, and Karthikeyan Ramachandran, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, AWS Glue is a fully managed extract, transform, and load (ETL) service. AWS Glue makes it cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams.

This pattern provides different job types in AWS Glue and uses three different scripts to demonstrate authoring ETL jobs.

You can use AWS Glue to write ETL jobs in a Python shell environment. You can also create both batch and streaming ETL jobs by using Python (PySpark) or Scala in a managed Apache Spark environment. To get you started with authoring ETL jobs, this pattern focuses on batch ETL jobs using Python shell, PySpark, and Scala. Python shell jobs are meant for workloads requiring lesser compute power. The managed Apache Spark environment is meant for workloads requiring high compute power.

Apache Parquet is built to support efficient compression and encoding schemes. It can speed up your analytics workloads because it stores data in a columnar fashion. Converting data to Parquet can save you storage space, cost, and time in the longer run. To learn more about Parquet, see the blog post Apache Parquet: How to be a hero with the open-source columnar data format.

Prerequisites and limitations

Prerequisites

- AWS Identity and Access Management (IAM) role (If you don't have a role, see the Additional information section.)

Architecture

Target technology stack

- AWS Glue
- Amazon Simple Storage Service (Amazon S3)

- Apache Parquet

Automation and scale

- AWS Glue workflows support full automation of an ETL pipeline.
- You can change the number of data processing units (DPUs), or worker types, to scale horizontally and vertically.

Tools

AWS services

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Glue is a fully managed ETL service for categorizing, cleaning, enriching, and moving your data between various data stores and data streams.

Other tools

- Apache Parquet is an open-source column-oriented data file format designed for storage and retrieval.

Configuration

Use the following settings for configuring the compute power of AWS Glue ETL. To reduce costs, use the minimal settings when you run the workload that is provided in this pattern.

- **Python shell –** You can use 1 DPU to utilize 16 GB of memory or 0.0625 DPU to utilize 1 GB of memory. This pattern uses 0.0625 DPU, which is the default in the AWS Glue console.
- **Python or Scala for Spark –** If you choose the Spark-related job types in the console, AWS Glue by default uses 10 workers and the G.1X worker type. This pattern uses two workers, which is the minimum number allowed, with the standard worker type, which is sufficient and cost effective.

The following table displays the different AWS Glue worker types for the Apache Spark environment. Because a Python shell job does not use the Apache Spark environment to run Python, it is not included in the table.

	Standard	G.1X	G.2X
vCPU	4	4	8
Memory	16 GB	16 GB	32 GB
Disk space	50 GB	64 GB	128 GB
Executor per worker	2	1	1

Code

For the code that is used in this pattern, including the IAM role and parameter configuration, see the Additional information section.

Epics

Upload the data

Task	Description	Skills required
Upload the data into a new or existing S3 bucket.	Create or use an existing S3 bucket in your account. Upload the sample_data.csv file from the Attachments section, and note the S3 bucket and prefix location.	General AWS

Create and run the AWS Glue job

Task	Description	Skills required
Create the AWS Glue job.	Under the ETL section of the AWS Glue console, add an AWS Glue job. Select the appropriate job type, AWS Glue version, and the	Developer, cloud or data

Task	Description	Skills required
	corresponding DPU/Worker type and number of workers. For details, see the *Configura tion* section.	
Change the input and output locations.	Copy the code corresponding to your AWS Glue job, and change the input and output location that you noted in the **Upload the data** epic.	Developer, cloud or data

Task	Description	Skills required
Configure the parameters.	You can use the snippets provided in the [Additional information](#) section to set parameters for your ETL job. AWS Glue uses four argument names internally: • `--conf` • `--debug` • `--mode` • `--JOB_NAME` The `--JOB_NAME` parameter must be explicitly entered on the AWS Glue console. Choose **Jobs**, **Edit Job**, **Security configuration, script libraries, and job parameters (optional)**. Enter `--JOB_NAME` as the key and provide a value. You can also use the AWS Command Line Interface (AWS CLI) or the AWS Glue API to set this parameter. The `--JOB_NAME` parameter is used by Spark and is not needed in a Python shell environment job. You must add `--` before every parameter name; otherwise, the code will not work. For example, for the code snippets, the location	Developer, cloud or data

Task	Description	Skills required
	parameters must be invoked by `--input_loc` and `--output_loc` .	
Run the ETL job.	Run your job and check the output. Note how much space was reduced from the original file.	Developer, cloud or data

Related resources

References

- Apache Spark
- AWS Glue: How it works
- AWS Glue pricing

Tutorials and videos

- What is AWS Glue?

Additional information

IAM role

When you create the AWS Glue jobs, you can use either an existing IAM role that has the permissions shown in the following code snippet or a new role.

To create a new role, use the following YAML code.

```
# (c) 2022 Amazon Web Services, Inc. or its affiliates. All Rights Reserved. This AWS
 Content is provided subject to the terms of the AWS Customer
# Agreement available at https://aws.amazon.com/agreement/ or other written agreement
 between Customer and Amazon Web Services, Inc.

AWSTemplateFormatVersion: "2010-09-09"
```

```
Description: This template will setup IAM role for AWS Glue service.

Resources:
  rGlueRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - "glue.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
      Policies:
        - PolicyName: !Sub "${AWS::StackName}-s3-limited-read-write-inline-policy"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - "s3:PutObject"
                  - "s3:GetObject"
                Resource: "arn:aws:s3:::*/*"
      Tags:
        - Key  : "Name"
          Value : !Sub "${AWS::StackName}"

Outputs:
  oGlueRoleName:
    Description: AWS Glue IAM role
    Value:
      Ref: rGlueRole
    Export:
      Name: !Join [ ":", [ !Ref "AWS::StackName", rGlueRole ] ]
```

AWS Glue Python Shell

The Python code uses the Pandas and PyArrow libraries to convert data to Parquet. The Pandas library is already available. The PyArrow library is downloaded when you run the pattern, because it is a one-time run. You can use wheel files to convert PyArrow to a library and provide the file as a library package. For more information about packaging wheel files, see [Providing your own Python library](#).

AWS Glue Python shell parameters

```
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["input_loc", "output_loc"])
```

AWS Glue Python shell code

```
from io import BytesIO
import pandas as pd
import boto3
import os
import io
import site
from importlib import reload
from setuptools.command import easy_install
install_path = os.environ['GLUE_INSTALLATION']
easy_install.main( ["--install-dir", install_path, "pyarrow"] )
reload(site)
import pyarrow


input_loc = "s3://bucket-name/prefix/sample_data.csv"
output_loc = "s3://bucket-name/prefix/"


input_bucket = input_loc.split('/', 1)[0]
object_key = input_loc.split('/', 1)[1]


output_loc_bucket = output_loc.split('/', 1)[0]
output_loc_prefix = output_loc.split('/', 1)[1]


s3 = boto3.client('s3')
obj = s3.get_object(Bucket=input_bucket, Key=object_key)
```

```
df = pd.read_csv(io.BytesIO(obj['Body'].read()))


parquet_buffer = BytesIO()
s3_resource = boto3.resource('s3')
df.to_parquet(parquet_buffer, index=False)
s3_resource.Object(output_loc_bucket, output_loc_prefix +  'data' +
 '.parquet').put(Body=parquet_buffer.getvalue())
```

AWS Glue Spark job with Python

To use an AWS Glue Spark job type with Python, choose **Spark** as the job type. Choose **Spark 3.1, Python 3 with improved job startup time (Glue Version 3.0)** as the AWS Glue version.

AWS Glue Python parameters

```
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME", "input_loc", "output_loc"])
```

AWS Glue Spark job with Python code

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
from awsglue.dynamicframe import DynamicFrame
from awsglue.utils import getResolvedOptions
from awsglue.job import Job


sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)

input_loc = "s3://bucket-name/prefix/sample_data.csv"
output_loc = "s3://bucket-name/prefix/"

inputDyf = glueContext.create_dynamic_frame_from_options(\
    connection_type = "s3", \
    connection_options = {
        "paths": [input_loc]}, \
```

```
    format = "csv",
    format_options={
        "withHeader": True,
        "separator": ","
    })


outputDF = glueContext.write_dynamic_frame.from_options(\
    frame = inputDyf, \
    connection_type = "s3", \
    connection_options = {"path": output_loc \
        }, format = "parquet")
```

For a large number of compressed big files (for example, 1,000 files that are each about 3 MB), use the `compressionType` parameter with the `recurse` parameter to read all the files that are available within the prefix, as shown in the following code.

```
input_loc = "bucket-name/prefix/"
output_loc = "bucket-name/prefix/"

inputDyf = glueContext.create_dynamic_frame_from_options(
                    connection_type = "s3",
                    connection_options = {"paths": [input_loc],
                                            "compressionType":"gzip","recurse" :"True",
                                            },
                    format = "csv",
                    format_options={"withHeader": True,"separator": ","}
                    )
```

For a large number of compressed small files (for example, 1,000 files that are each about 133 KB), use the `groupFiles` parameter, along with both the `compressionType` and the `recurse` parameters. The `groupFiles` parameter groups small files into multiple big files, and the `groupSize` parameter controls the grouping to the specified size in bytes (for example, 1 MB). The following code snippet provides an example of using these parameters within the code.

```
input_loc = "bucket-name/prefix/"
output_loc = "bucket-name/prefix/"

inputDyf = glueContext.create_dynamic_frame_from_options(
                    connection_type = "s3",
                    connection_options = {"paths": [input_loc],
                                            "compressionType":"gzip","recurse" :"True",
```

```
                                                "groupFiles" :"inPartition",
 "groupSize" :"1048576",
                                               },
                    format = "csv",
                    format_options={"withHeader": True,"separator": ","}
                    )
```

Without any change in the worker nodes, these settings enable the AWS Glue job to read multiple files (large or small, with or without compression) and write them to the target in Parquet format.

AWS Glue Spark job with Scala

To use an AWS Glue Spark job type with Scala, choose **Spark** as the job type and **Language** as **Scala**. Choose **Spark 3.1, Scala 2 with improved job startup time (Glue Version 3.0)** as the AWS Glue version. To save on storage space, the following AWS Glue with Scala sample also uses the `applyMapping` feature to convert data types.

AWS Glue Scala parameters

```
import com.amazonaws.services.glue.util.GlueArgParser val args =
 GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME", "inputLoc",
 "outputLoc").toArray)
```

AWS Glue Spark job with Scala code

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.DynamicFrame
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueScalaApp {
  def main(sysArgs: Array[String]) {

    @transient val spark: SparkContext = SparkContext.getOrCreate()
    val glueContext: GlueContext = new GlueContext(spark)
```

```
    val inputLoc = "s3://bucket-name/prefix/sample_data.csv"
    val outputLoc = "s3://bucket-name/prefix/"

    val readCSV = glueContext.getSource("csv", JsonOptions(Map("paths" ->
 Set(inputLoc)))).getDynamicFrame()

    val applyMapping = readCSV.applyMapping(mappings = Seq(("_c0", "string", "date",
 "string"), ("_c1", "string", "sales", "long"),
    ("_c2", "string", "profit", "double")), caseSensitive = false)

    val formatPartition = applyMapping.toDF().coalesce(1)

    val dynamicFrame = DynamicFrame(formatPartition, glueContext)

    val dataSink = glueContext.getSinkWithFormat(
        connectionType = "s3",
        options = JsonOptions(Map("path" -> outputLoc )),
        transformationContext = "dataSink", format =
 "parquet").writeDynamicFrame(dynamicFrame)
  }
}
```

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Visualize Amazon Redshift audit logs using Amazon Athena and Amazon QuickSight

Sanket Sirsikar and Gopal Krishna Bhatia, Amazon Web Services

Summary

Security is an integral part of database operations on the Amazon Web Services (AWS) Cloud. Your organization should ensure that it monitors database user activities and connections to detect potential security incidents and risks. This pattern helps you monitor your databases for security and troubleshooting purposes, which is a process known as database auditing.

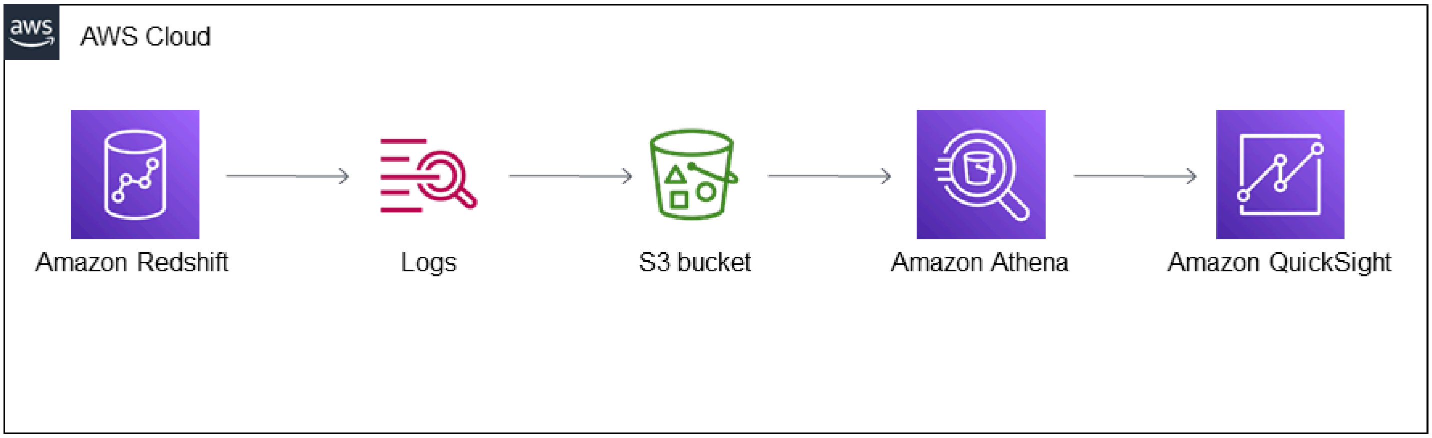
This pattern provides a SQL script that automates the creation of an Amazon Athena table and views for a reporting dashboard in Amazon QuickSight that helps you audit Amazon Redshift logs. This ensures that users responsible for monitoring database activities have convenient access to data security features.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing Amazon Redshift cluster. For more information about this, see Create an Amazon Redshift cluster in the Amazon Redshift documentation.
- Access to an existing Athena workgroup. For more information, see How workgroups work in the Amazon Athena documentation.
- An existing Amazon Simple Storage Service (Amazon S3) source bucket with the required AWS Identity and Access Management (IAM) permissions. For more information, see Bucket permissions for Amazon Redshift audit logging from Database audit logging in the Amazon Redshift documentation.

Architecture



Technology stack

- Athena
- Amazon Redshift
- Amazon S3
- QuickSight

Tools

- Amazon Athena – Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.
- Amazon QuickSight – QuickSight is a scalable, serverless, embeddable, machine learning-powered business intelligence (BI) service.
- Amazon Redshift – Amazon Redshift is an enterprise-level, petabyte scale, fully managed data warehousing service.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.

Epics

Configure the Amazon Redshift cluster

Task	Description	Skills required
Enable audit logging for the Amazon Redshift cluster.	1. Sign in to the AWS Management Console, open the Amazon Redshift console, choose **CLUSTERS**, and then choose the cluster that you want to enable logging for. 2. Choose the **Properties** tab and then enable auditing by following the instructi ons from [Configuring auditing using the console](#) in the Amazon Redshift documentation.	DBA, Data engineer
Enable logging in the Amazon Redshift cluster parameter group.	You can enable auditing of connection logs, user logs, and user activity logs at the same time by using the AWS Management Console, the Amazon Redshift API reference, or AWS Command Line Interface (AWS CLI). For auditing of user activity logs, you must enable the `enable_user_activi ty_logging` database parameter. If you only enable the audit logging feature but not the associated parameter	DBA, Data engineer

Task	Description	Skills required
	, the database audit logs the logging information for the connection and user logs but not for the user activity logs. The `enable_user_activity_logging` parameter is not enabled by default, but you can enable it by changing it from `false` to `true`. **Important** You need to create a new cluster parameter group with the `user_activity_logging` parameter enabled and attach it to your Amazon Redshift cluster. For more information about this, see [Modifying a cluster](#) in the Amazon Redshift documentation. For more information about this task, see [Amazon Redshift parameter groups](#) and [Configuring auditing using the console](#) in the Amazon Redshift documentation.	

Task	Description	Skills required
Configure S3 bucket permissions for Amazon Redshift cluster logging.	When you enable logging, Amazon Redshift collects logging information and uploads it to log files stored in an S3 bucket. You can use an existing S3 bucket or create a new bucket. **Important** Make sure that Amazon Redshift has the required IAM permissions to access the S3 bucket. For more information about this, see [Bucket permissions for Amazon Redshift audit logging] from [Database audit logging] in the Amazon Redshift documentation.	DBA, Data engineer

Create the Athena table and views

Task	Description	Skills required
Create the Athena table and views to query Amazon Redshift audit log data from the S3 bucket.	Open the Amazon Athena console and use the data definition language (DDL) query from the `AuditLogging.sql` SQL script	Data engineer

Task	Description	Skills required
	(attached) to create the table and views for user activity logs, user logs, and connectio n logs. For more information and instructions, see the Create tables and run queries tutorial from the Amazon Athena Workshop.	

Set up log monitoring in the QuickSight dashboard

Task	Description	Skills required
Create a QuickSight dashboard using Athena as the data source.	Open the Amazon QuickSigh t console and create a QuickSight dashboard by following the instructions in the Visualize with QuickSigh t using Athena tutorial from the Amazon Athena Workshop.	DBA, Data engineer

Related resources

- Create tables and run queries in Athena
- Visualize with QuickSight using Athena

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Visualize IAM credential reports for all AWS accounts using Amazon QuickSight

Parag Nagwekar and Arun Chandapillai, Amazon Web Services

Summary

Warning: IAM users have long-term credentials, which presents a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed.

You can use AWS Identity and Access Management (IAM) credential reports to help you meet the security, auditing, and compliance requirements of your organization. Credential reports provide a list of all the users in your AWS accounts and show the status of their credentials, such as passwords, access keys, and multi-factor authentication (MFA) devices. You can use credential reports for multiple AWS accounts managed by AWS Organizations.

This pattern includes steps and code to help you create and share IAM credential reports for all the AWS accounts in your organization by using Amazon QuickSight dashboards. You can share the dashboards with stakeholders in your organization. The reports can help your organization achieve the following targeted business outcomes:

- Identify security incidents related to IAM users
- Track real-time migration of IAM users to single sign-on (SSO) authentication
- Track AWS Regions accessed by IAM users
- Stay compliant
- Share information with other stakeholders

Prerequisites and limitations

Prerequisites

- An active AWS account
- An organization with member accounts

- An [IAM role] with permissions to access accounts in Organizations
- AWS Command Line Interface (AWS CLI) version 2, [installed] and [configured]
- A [subscription] to [Amazon QuickSight Enterprise edition]

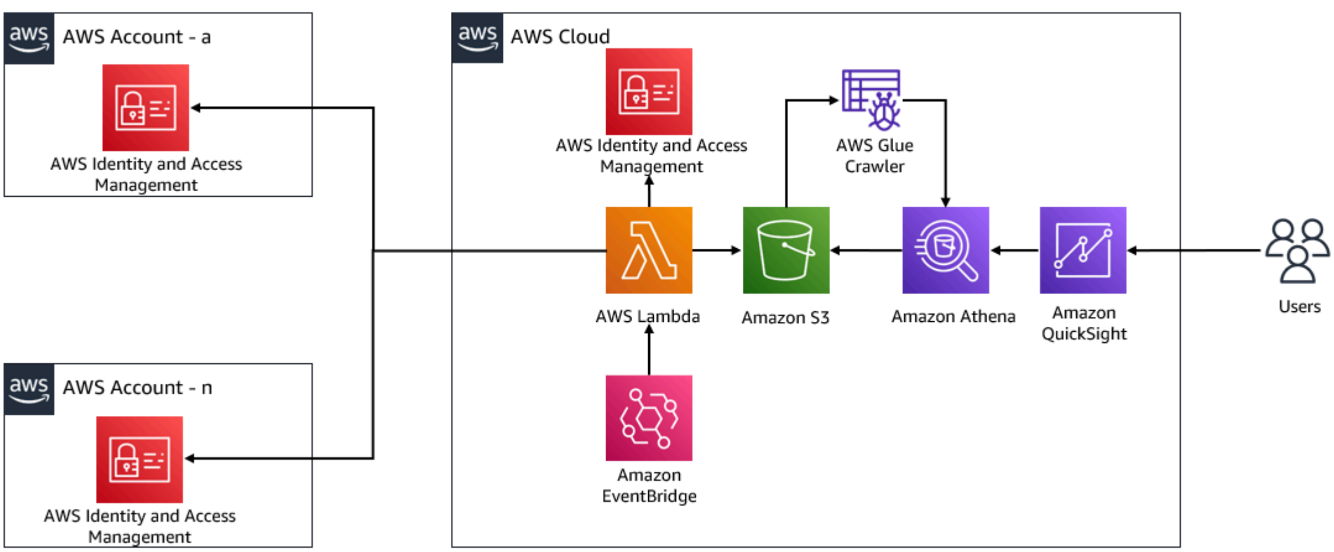
Architecture

Technology stack

- Amazon Athena
- Amazon EventBridge
- Amazon QuickSight
- Amazon Simple Storage Service (Amazon S3)
- AWS Glue
- AWS Identity and Access Management (IAM)
- AWS Lambda
- AWS Organizations

Target architecture

The following diagram shows an architecture for setting up a workflow that captures IAM credential report data from multiple AWS accounts.



1. EventBridge invokes a Lambda function daily.

2. The Lambda function assumes an IAM role in every AWS account across the organization. Then, the function creates the IAM credentials report and stores the report data in a centralized S3 bucket. You must enable encryption and deactivate public access on the S3 bucket.
3. An AWS Glue crawler crawls the S3 bucket daily and updates the Athena table accordingly.
4. QuickSight imports and analyzes the data from the credential report and builds a dashboard that can be visualized by and shared with stakeholders.

Tools

AWS services

- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 by using standard SQL.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data in a single dashboard.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Code

The code for this pattern is available in the GitHub getiamcredsreport-allaccounts-org repository. You can use the code from this repository to create IAM credential reports across AWS accounts in Organizations and store them in a central location.

Epics

Set up the infrastructure

Task	Description	Skills required
Set up Amazon QuickSight Enterprise edition.	1. Activate the Amazon QuickSight Enterprise edition in your AWS account. For more information, see Managing user access inside Amazon QuickSight in the QuickSight documentation. 2. To grant dashboard permissions, get the Amazon Resource Name (ARN) of the QuickSight users.	AWS administrator, AWS DevOps, Cloud administrator, Cloud architect
Integrate Amazon QuickSight with Amazon S3 and Athena.	You must authorize QuickSight to use Amazon S3 and Athena before you deploy the AWS CloudFormation stack.	AWS administrator, AWS DevOps, Cloud administrator, Cloud architect

Deploy the infrastructure

Task	Description	Skills required
Clone the GitHub repository.	1. Clone the GitHub getiamcredsreport-allaccounts-org repository to your local machine by running the following command: `git clone https://github.com`	AWS administrator

Task	Description	Skills required
	`/aws-samples/getiamcredsreport-allaccounts-org`	

Task	Description	Skills required
Deploy the infrastructure.	1. Sign in to the AWS Management Console and open the [CloudFormation console](https://console.aws.amazon.com/cloudformation/). 2. In the navigation pane, choose **Create stack**, and then choose **With new resources (standard)**. 3. On the **Identify resources** page, choose **Next**. 4. On the **Specify template** page, for **Template source**, select **Upload a template file**. 5. Choose **Choose file**, select the `Cloudformation-createcredrepo.yaml` file from your cloned GitHub repository, and then choose **Next**. 6. In **Parameters**, update `IAMRoleName` with your IAM role. This should be the IAM role that you want Lambda to assume in every account of the organization. This role creates the credential report. **Note**: The role doesn't have to be present in all accounts at this step of stack creation. 7. In **Parameters**, update `S3BucketName` with the	AWS administrator

Task	Description	Skills required
	name of the S3 bucket where Lambda can store the credentials for all accounts. 8. For **Stack name**, enter your stack name. 9. Choose **Submit**. 10Note the Lambda function's role name.	

Create an IAM permission policy.	[Create an IAM policy](#) for every AWS account across your organization with the following permissions:
	<pre>{
	  "Version": "2012-10-
	17",
	  "Statement": [
	    {
	      "Effect": "Allow",
	      "Action": [
	        "iam:Gene
	rateCredentialRepo
	rt",
	        "iam:GetC
	redentialReport"
	      ],
	      "Resource": "*"
	    }
	  ]
	}</pre>
		AWS DevOps, Cloud administrator, Cloud architect, Data engineer

Task	Description	Skills required
Create an IAM role with a trust policy.	1. [Create an IAM role](#) for the AWS accounts and attach the permission policy that you created in the previous step. 2. Attach the following trust policy to the IAM role: `{ "Version":"2012-10-17", "Statement":[ { "Effect": "Allow", "Principal":{ "AWS":[ "arn:aws:iam::<MasterAccountID>:role/<LambdaRole>" ] }, "Action": "sts:AssumeRole" } ] }` **Important** Replace `arn:aws:iam::<MasterAccountID>:role/<LambdaRole>` with the ARN of the	Cloud administrator, Cloud architect, AWS administrator

Task	Description	Skills required
	Lambda role that you noted previously. **Note** Organizations typically use automation to create IAM roles for their AWS accounts. We recommend that you use this automation, if available. Alternatively, you can use the `CreateRoleforOrg.py` script from the code repository. The script requires an existing administrative role or any other IAM role that has permission to create an IAM policy and role in every AWS account.	

Task	Description	Skills required
Configure Amazon QuickSight to visualize the data.	1. Sign in to QuickSight with your credentials. 2. Create a dataset by using Athena (using the `iamcredreportdb` database and `"cfn_iamcredreport"` table), and then automatically refresh the dataset. 3. Create an analysis in QuickSight. 4. Create a QuickSight dashboard.	AWS DevOps, Cloud administrator, Cloud architect, Data engineer

Additional information

Additional considerations

Consider the following:

- After you use CloudFormation to deploy the infrastructure, you can wait to get the reports created in Amazon S3 and analyzed by Athena until Lambda and AWS Glue run as per their schedules. Alternatively, you can run Lambda manually to get the reports in Amazon S3, and then run the AWS Glue crawler to get the Athena table that's created from the data.
- QuickSight is a powerful tool for analyzing and visualizing data based on your business requirements. You can use parameters in QuickSight to control widget data based on data fields that you choose. Also, you can use a QuickSight analysis to create parameters (for example, Account, Date, and User fields such as `partition_0`, `partition_1`, and `user` respectively) from your dataset to add controls for the parameters for Account, Date, and User.
- To build your own QuickSight dashboards, see QuickSight Workshops from the AWS Workshop Studio website.
- To see sample QuickSight dashboards, see the GitHub getiamcredsreport-allaccounts-org code repository.

Targeted business outcomes

You can use this pattern to achieve the following targeted business outcomes:

- **Identify security incidents related to IAM users –** Investigate every user across every AWS account in your organization by using a single pane of glass. You can track the trend of an IAM user's most recently accessed individual AWS Regions and the services they used.
- **Track real-time migration of IAM users to SSO authentication –** By using SSO, users can sign in once with a single credential and access multiple AWS accounts and applications. If you're planning to migrate your IAM users to SSO, this pattern can help you transition to SSO and track all IAM user credential usage (such as access to the AWS Management Console or usage of access keys) across all AWS accounts.
- **Track AWS Regions accessed by IAM users –** You can control IAM user access to Regions for various purposes, such as data sovereignty and cost control. You can also track use of Regions by any IAM user.
- **Stay compliant –** By following the principle of least privilege, you can grant only the specific IAM permissions that are required to perform a specific task. Also, you can track access to AWS services, the AWS Management Console, and long-term credentials usage.
- **Share information with other stakeholders –** You can share curated dashboards with other stakeholders, without granting them access to IAM credential reports or AWS accounts.

More patterns

- Authenticate existing React application users by using Amazon Cognito and AWS Amplify UI
- Automatically extract content from PDF files using Amazon Textract
- Build a cold start forecasting model by using DeepAR for time series in Amazon SageMaker AI Studio Lab
- Create detailed cost and usage reports for Amazon EMR clusters by using AWS Cost Explorer
- Create detailed cost and usage reports for Amazon RDS and Amazon Aurora
- Create detailed cost and usage reports for AWS Glue jobs by using AWS Cost Explorer
- Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Firehose with AWS CDK
- Deploy an AWS Glue job with an AWS CodePipeline CI/CD pipeline
- Embed an Amazon QuickSight dashboard in a local Angular application
- Ensure an Amazon Redshift cluster is encrypted upon creation
- Ensure encryption for Amazon EMR data at rest is enabled at launch
- Estimate storage costs for an Amazon DynamoDB table
- Execute Amazon Redshift SQL queries by using Terraform
- Extract and query AWS IoT SiteWise metadata attributes in a data lake
- Generate data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Generate Db2 z/OS data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Give SageMaker notebook instances temporary access to a CodeCommit repository in another AWS account
- Identify and alert when Amazon Data Firehose resources are not encrypted with an AWS KMS key
- Import the psycopg2 library to AWS Lambda to interact with your PostgreSQL database
- Ingest and analyze AWS security logs in Microsoft Sentinel
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL in SSL mode by using AWS DMS
- Migrate an Oracle database to Amazon RDS for Oracle by using Oracle GoldenGate flat file adapters
- Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT
- Migrate data from an on-premises Hadoop environment to Amazon S3 using DistCp with AWS PrivateLink for Amazon S3
- Migrate from Couchbase Server to Couchbase Capella on AWS

- Migrate on-premises Cloudera workloads to Cloudera Data Platform on AWS
- Monitor Amazon EMR clusters for in-transit encryption at launch
- Manage Multi-AZ failover for EMR clusters by using Application Recovery Controller
- Automate blue/green deployments of Amazon Aurora global databases by using IaC principles
- Provision AWS Service Catalog products based on AWS CloudFormation templates by using GitHub Actions
- Run unit tests for Python ETL jobs in AWS Glue using the pytest framework
- Set up a Grafana monitoring dashboard for AWS ParallelCluster
- Unload data from an Amazon Redshift cluster across accounts to Amazon S3
- Verify that new Amazon Redshift clusters have required SSL endpoints
- Verify that new Amazon Redshift clusters launch in a VPC
- Visualize AI/ML model results using Flask and AWS Elastic Beanstalk

Compute

Topics

- Containers & microservices
- Serverless
- Networking
- Content delivery

Containers & microservices

Topics

- Access an Amazon Neptune database from an Amazon EKS container
- Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer
- Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer
- Activate mTLS in AWS App Mesh using AWS Private CA on Amazon EKS
- Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch
- Automate deployment of Node Termination Handler in Amazon EKS by using a CI/CD pipeline
- Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS
- Deploy Lambda functions with container images
- Deploy Java microservices on Amazon ECS using AWS Fargate
- Deploy Kubernetes resources and packages using Amazon EKS and a Helm chart repository in Amazon S3
- Deploy a sample Java microservice on Amazon EKS and expose the microservice using an Application Load Balancer
- Deploy a clustered application to Amazon ECS by using AWS Copilot
- Deploy a gRPC-based application on an Amazon EKS cluster and access it with an Application Load Balancer
- Deploy and debug Amazon EKS clusters
- Deploy containers by using Elastic Beanstalk
- Generate a static outbound IP address using a Lambda function, Amazon VPC, and a serverless architecture
- Identify duplicate container images automatically when migrating to an Amazon ECR repository
- Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet

- Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands
- Migrate NGINX Ingress Controllers when enabling Amazon EKS Auto Mode
- Migrate your container workloads from Azure Red Hat OpenShift (ARO) to Red Hat OpenShift Service on AWS (ROSA)
- Place Kubernetes Pods on Amazon EKS by using node affinity, taints, and tolerations
- Replicate filtered Amazon ECR container images across accounts or Regions
- Rotate database credentials without restarting containers
- Run Amazon ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere
- Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance
- Run message-driven workloads at scale by using AWS Fargate
- Run stateful workloads with persistent data storage by using Amazon EFS on Amazon EKS with AWS Fargate
- Set up event-driven auto scaling in Amazon EKS by using Amazon EKS Pod Identity and KEDA
- Streamline PostgreSQL deployments on Amazon EKS by using PGO
- Simplify application authentication with mutual TLS in Amazon ECS by using Application Load Balancer
- More patterns

Access an Amazon Neptune database from an Amazon EKS container

Ramakrishnan Palaninathan, Amazon Web Services

Summary

This pattern establishes a connection between Amazon Neptune, which is a fully managed graph database, and Amazon Elastic Kubernetes Service (Amazon EKS), a container orchestration service, to access a Neptune database. Neptune DB clusters are confined within a virtual private cloud (VPC) on AWS. For this reason, accessing Neptune requires careful configuration of the VPC to enable connectivity.

Unlike Amazon Relational Database Service (Amazon RDS) for PostgreSQL, Neptune doesn't rely on typical database access credentials. Instead, it uses AWS Identity and Access Management (IAM) roles for authentication. Therefore, connecting to Neptune from Amazon EKS involves setting up an IAM role with the necessary permissions to access Neptune.

Furthermore, Neptune endpoints are accessible only within the VPC where the cluster resides. This means that you have to configure network settings to facilitate communication between Amazon EKS and Neptune. Depending on your specific requirements and networking preferences, there are various approaches to configuring the VPC to enable seamless connectivity between Neptune and Amazon EKS. Each method offers distinct advantages and considerations, which provide flexibility in designing your database architecture to suit your application's needs.

Prerequisites and limitations

Prerequisites

- Install the latest version of **kubectl** (see instructions). To check your version, run:

```
kubectl version --short
```

- Install the latest version of **eksctl** (see instructions). To check your version, run:

```
eksctl info
```

- Install the latest version of the AWS Command Line Interface (AWS CLI) version 2 (see instructions). To check your version, run:

```
aws --version
```

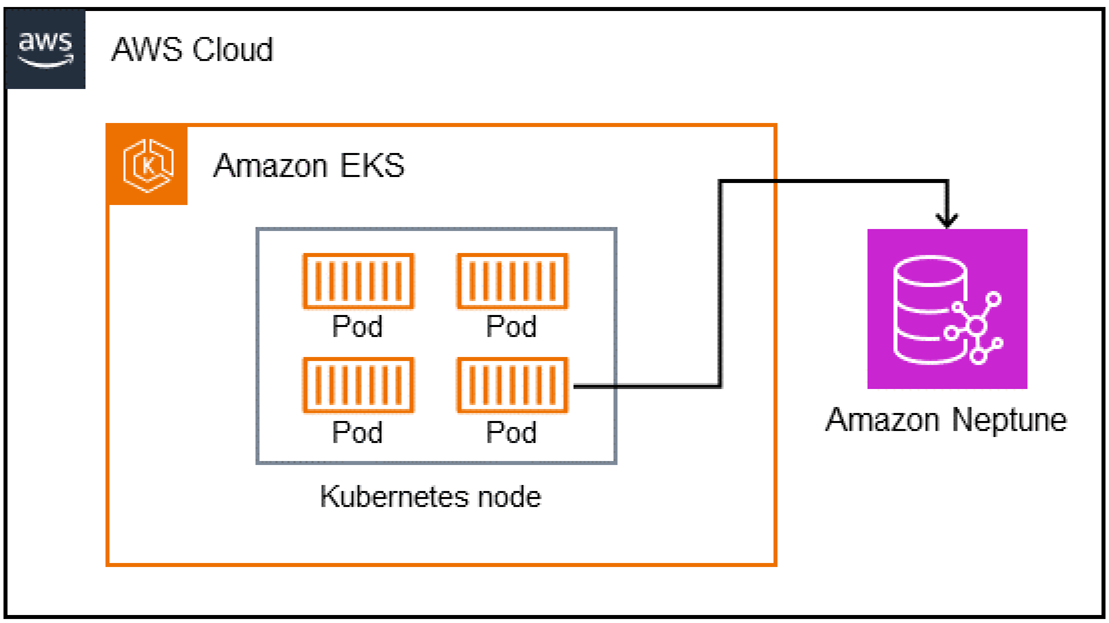
- Create a Neptune DB cluster (see instructions). Make sure to establish communications between the cluster's VPC and Amazon EKS through VPC peering, AWS Transit Gateway, or another method. Also make sure that the status of the cluster is "available" and that it has an inbound rule on port 8182 for the security group.
- Configure an IAM OpenID Connect (OIDC) provider on an existing Amazon EKS cluster (see instructions).

Product versions

- Amazon EKS 1.27
- Amazon Neptune engine version 1.3.0.0 (2023-11-15)

Architecture

The following diagram shows the connection between Kubernetes pods in an Amazon EKS cluster and Neptune to provide access to a Neptune database.



Automation and scale

You can use the Amazon EKS Horizontal Pod Autoscaler to scale this solution.

Tools

Services

- [Amazon Elastic Kubernetes Service (Amazon EKS)](#) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [Amazon Neptune](#) is a graph database service that helps you build and run applications that work with highly connected datasets.

Best practices

For best practices, see [Identity and Access Management](#) in the *Amazon EKS Best Practices Guides*.

Epics

Set environment variables

Task	Description	Skills required
Verify the cluster context.	Before you interact with your Amazon EKS cluster by using Helm or other command-line tools, you must define environment variables that encapsulate your cluster's details. These variables are used in subsequent commands to ensure that they target the correct cluster and resources. First, confirm that you are operating within the correct cluster context. This ensures that any subsequent commands are sent to the intended Kubernetes cluster. To verify the current context, run the following command.	AWS administrator, Cloud administrator

Task	Description	Skills required
	`kubectl config current-context`	
Define the `CLUSTER_NAME` variable.	Define the `CLUSTER_NAME` environment variable for your Amazon EKS cluster. In the following command, replace the sample value `us-west-2` with the correct AWS Region for your cluster. Replace the sample value `eks-workshop` with your existing cluster name. `export CLUSTER_NAME=$(aws eks describe-cluster --region us-west-2 --name eks-workshop --query "cluster.name" --output text)`	AWS administrator, Cloud administrator
Validate output.	To validate that the variables have been set properly, run the following command. `echo $CLUSTER_NAME` Verify that the output of this command matches the input you specified in the previous step.	AWS administrator, Cloud administrator

Create IAM role and associate it with Kubernetes

Task	Description	Skills required
Create a service account.	You use IAM roles for service accounts to map your Kubernetes service accounts to IAM roles, to enable fine-grained permissions management for your applications that run on Amazon EKS. You can use eksctl to create and associate an IAM role with a specific Kubernetes service account within your Amazon EKS cluster. The AWS managed policy `NeptuneFullAccess` allows write and read access to your specified Neptune cluster. **Important** You must have an OIDC endpoint associated with your cluster before you run these commands. Create a service account that you want to associate with an AWS managed policy named `NeptuneFullAccess` .	AWS administrator, Cloud administrator

Task	Description	Skills required

	<pre>eksctl create iamservic
	eaccount --name eks-
	neptune-sa --namespa
	ce default --cluster
	 $CLUSTER_NAME --
	attach-policy-arn
	 arn:aws:iam::aws:p
	olicy/NeptuneFullA
	ccess --approve --
	override-existing-
	serviceaccounts</pre>
	where `eks-neptune-sa` is the name of the service account that you want to create.
	
	Upon completion, this command displays the following response:
	<pre>2024-02-07 01:12:39 [#]
	 created serviceaccount
	 "default/eks-neptune-
	sa"</pre>	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Verify that the account is set up properly.</td><td>Make sure that the <code>eks-neptune-sa</code> service account is set up correctly in the default namespace in your cluster.
<pre>kubectl get sa eks-neptu
ne-sa -o yaml</pre>The output should look like this:
<pre>apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/
role-arn: arn:aws:i
am::123456789123:r
ole/eksctl-eks-wor
kshop-addon-iamser
viceaccount-d-Role1-
Q35yKgdQOlmM
  creationTimestamp:
 "2024-02-07T01:12:
39Z"
  labels:
    app.kubernetes.io/
managed-by: eksctl
  name: eks-neptune-sa
  namespace: default
  resourceVersion:
 "5174750"
  uid: cd6ba2f7-
a0f5-40e1-a6f4-408
1e0042316</pre></td><td>AWS administrator, Cloud administrator</td></tr>
</table>

Task	Description	Skills required

Check connectivity.	Deploy a sample pod called `pod-util` and check connectivity with Neptune.
	``
	apiVersion: v1
	kind: Pod
	metadata:
	  name: pod-util
	  namespace: default
	spec:
	  serviceAccountName: eks-neptune-sa
	  containers:
	  - name: pod-util
	    image: public.ecr.aws/patrickc/troubleshoot-util
	    command:
	      - sleep
	      - "3600"
	    imagePullPolicy: IfNotPresent
	``
	``
	kubectl apply -f pod-util.yaml
	``
	``
	kubectl exec --stdin --tty pod-util -- /bin/bash
	bash-5.1# curl -X POST -d '{"gremlin":"g.V().limit(1)"}' https://db-neptune-1.cluster-xxxxxxxxxxxx.us-west-2.neptune.amazonaws.com:8182/gremlin
	{"requestId":"a4964f2d-12b1-4ed3-8a14-e
	``
		AWS administrator, Cloud administrator

Task	Description	Skills required

	`ff511431a0e","status":{"message":"","code":200,"attributes":{"@type":"g:Map","@value":[]}},"result":{"data":{"@type":"g:List","@value":[]},"meta":{"@type":"g:Map","@value":[]}}}`
	`bash-5.1# exit`
	`exit`	

Validate connection activity

Task	Description	Skills required
Enable IAM database authentication.	By default, IAM database authentication is disabled when you create a Neptune DB cluster. You can enable or disable IAM database authentication by using the AWS Management Console. Follow the steps in the AWS documentation to enable IAM database authentication in Neptune.	AWS administrator, Cloud administrator
Verify connections.	In this step, you interact with the `pod-util` container, which is already in running status, to install **awscurl** and verify the connection.	AWS administrator, Cloud administrator

Task	Description	Skills required

	1. Run the following command to find the pod.
	`kubectl get pods`
	The output should look like this:
	`NAME READY STATUS RESTARTS AGE pod-util 1/1 Running 0 50m`
	2. Run the following command to install **awscurl**.
	`kubectl exec --stdin --tty pod-util -- /bin/bash bash-5.1# pip3 install awscurl Installing collected packages: idna, configparser, configargparse, charset-normalizer, certifi, requests, awscurl Successfully installed awscurl-0.32 certifi-2024.2.2 charset-normalizer-3.3.2 configargparse-1.7 configparser-6.0.0 idna-3.6 requests-2.31.0 bash-5.1# awscurl https://db-neptune-1.cluster-xxxxxxxxxxxx.us-west-2.neptune.amazonaws.co`	

Task	Description	Skills required
	`m:8182/status --region us-west-2 --service neptune-db {"status":"healthy","startTime":"Thu Feb 08 01:22:14 UTC 2024","dbEngineVersion":"1.3.0.0.R1","role":"writer","dfeQueryEngine":"viaQueryHint","gremlin":{"version":"tinkerpop-3.6.4"},"sparql":{"version":"sparql-1.1"},"opencypher":{"version":"Neptune-9.0.20190305-1.0"},"labMode":{"ObjectIndex":"disabled","ReadWriteConflictDetection":"enabled"},"features":{"SlowQueryLogs":"disabled","ResultCache":{"status":"disabled"},"IAMAuthentication":"enabled","Streams":"disabled","AuditLog":"disabled"},"settings":{"clusterQueryTimeoutInMs":"120000","SlowQueryLogsThreshold":"5000"}}`	

Troubleshooting

Issue	Solution
Can't access the Neptune database.	Review the IAM policy that's attached to the service account. Make sure that it allows the necessary actions (for example, `neptune:Connec,neptune:DescribeDBInstances`) for the operations you want to run.

Related resources

- Grant Kubernetes workloads access to AWS using Kubernetes Service Accounts (Amazon EKS documentation)
- IAM roles for service accounts (Amazon EKS documentation)
- Creating a new Neptune DB cluster (Amazon Neptune documentation)

Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer

Kirankumar Chandrashekar, Amazon Web Services

Summary

This pattern describes how to privately host a Docker container application on Amazon Elastic Container Service (Amazon ECS) behind a Network Load Balancer, and access the application by using AWS PrivateLink. You can then use a private network to securely access services on the Amazon Web Services (AWS) Cloud. Amazon Relational Database Service (Amazon RDS) hosts the relational database for the application running on Amazon ECS with high availability (HA). Amazon Elastic File System (Amazon EFS) is used if the application requires persistent storage.

The Amazon ECS service running the Docker applications, with a Network Load Balancer at the front end, can be associated with a virtual private cloud (VPC) endpoint for access through AWS PrivateLink. This VPC endpoint service can then be shared with other VPCs by using their VPC endpoints.

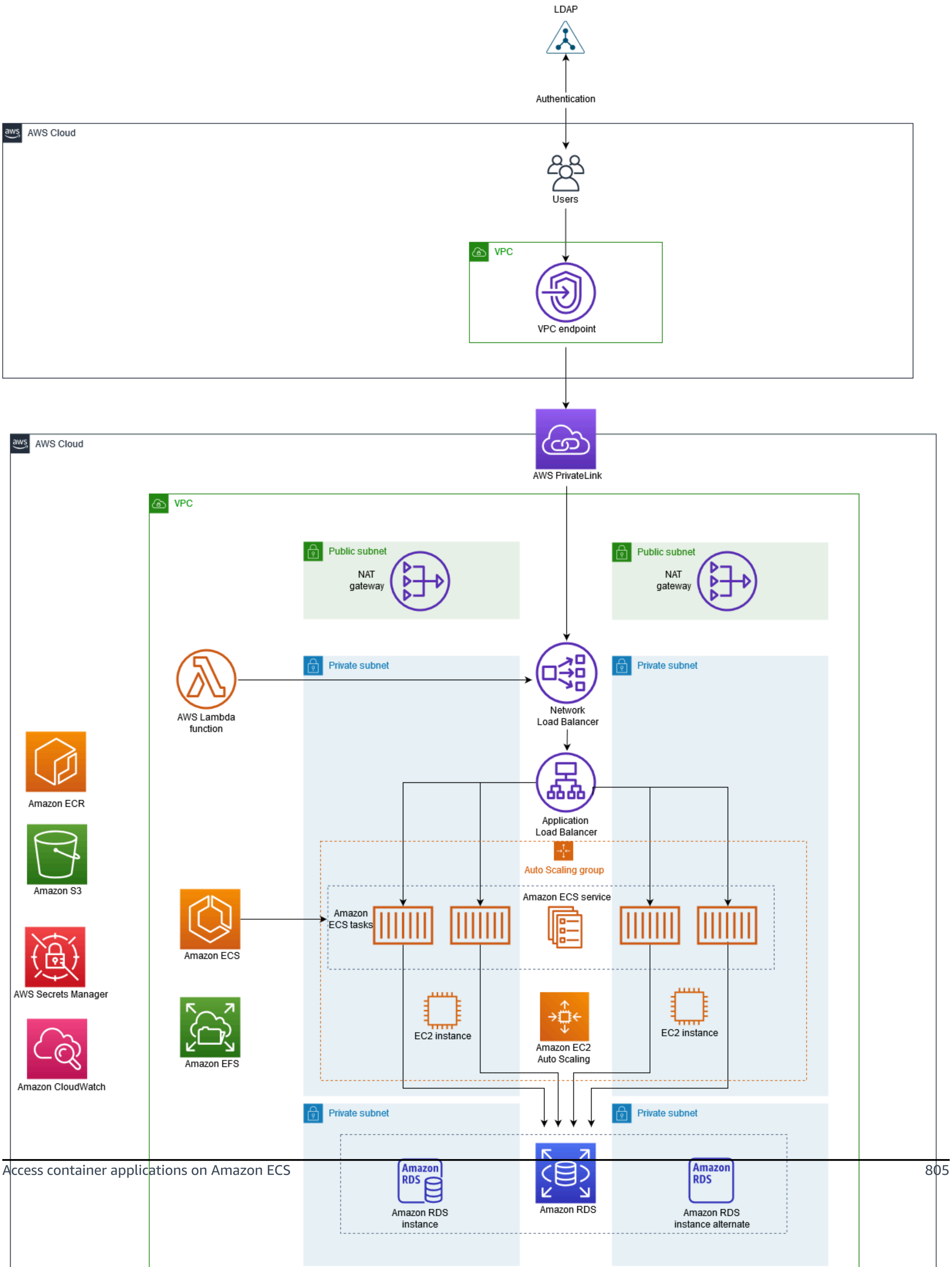
You can also use AWS Fargate instead of an Amazon EC2 Auto Scaling group. For more information, see Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows
- Docker, installed and configured on Linux, macOS, or Windows
- An application running on Docker

Architecture



Technology stack

- Amazon CloudWatch
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon EC2 Auto Scaling
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- Amazon RDS
- Amazon Simple Storage Service (Amazon S3)
- AWS Lambda
- AWS PrivateLink
- AWS Secrets Manager
- Application Load Balancer
- Network Load Balancer
- VPC

Automation and scale

- You can use AWS CloudFormation to create this pattern by using Infrastructure as Code.

Tools

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud.
- Amazon EC2 Auto Scaling – Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage containers on a cluster.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is a managed AWS container image registry service that is secure, scalable, and reliable.
- Amazon EFS – Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources.

- [AWS Lambda](#) – Lambda is a compute service for running code without provisioning or managing servers.
- [Amazon RDS](#) – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.
- [Amazon S3](#) – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers.
- [AWS Secrets Manager](#) – Secrets Manager helps you replace hardcoded credentials in your code, including passwords, by providing an API call to Secrets Manager to retrieve the secret programmatically.
- [Amazon VPC](#) – Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined.
- [Elastic Load Balancing](#) – Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon EC2 instances, containers, and IP addresses, in multiple Availability Zones.
- [Docker](#) – Docker helps developers to pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

Epics

Create networking components

Task	Description	Skills required
Create a VPC.	1. Sign in to the AWS Management Console and open the Amazon VPC console. Choose **Create VPC**, and choose **VPC and more**. 2. Enter a name for your VPC, and choose an appropriate CIDR block range. 3. Specify two Availability Zones, two public subnets, four private subnets. Two	Cloud administrator

Task	Description	Skills required
	private subnets are for Amazon ECS tasks, and two private subnets are for Amazon RDS databases. 4. Specify one NAT gateway for each Availability Zone. 5. Choose **Create VPC**.	

Create the load balancers

Task	Description	Skills required
Create a Network Load Balancer.	1. Open the Amazon EC2 console and choose the AWS Region that contains your VPC. 2. Under **Load balancing**, choose **Load balancers**, and choose **Create load balancer**. 3. Choose **Network Load Balancer**, and choose **Create**. 4. On the **Configure load balancer** page, configure your Network Load Balancer and listener. **Important:** Make sure you choose your Network Load Balancer's scheme as **Internal**. 5. Choose the applicabl e security settings,	Cloud administrator

Task	Description	Skills required
	configure a security group and a target group. Choose **Instance** or **IP** as the **Target type** in the **Configure routing** section. Make sure you do not register a target. 6. When you have configure d all the settings, choose **Next: Review**, and then choose **Create**.	

Task	Description	Skills required
Create an Application Load Balancer.	1. On the Amazon EC2 console, choose the same Region that contains your VPC. 2. Under **Load balancing**, choose **Load balancers**, and choose **Create load balancer**. 3. Choose **Application Load Balancer**, and choose **Create**. 4. **Important** Configure your Application Load Balancer and its listener. Make sure you choose your Application Load Balancer's scheme as **Internal**. 5. Choose the applicabl e security settings, configure a security group and a target group. Choose **Instance** or **IP** as the **Target type** in the **Configure routing** section. Make sure you do not register a target. 6. When you have configure d all the settings, choose	Cloud administrator

Task	Description	Skills required
	Next: Review, and then choose **Create**.	

Create an Amazon EFS file system

Task	Description	Skills required
Create an Amazon EFS file system.	1. Open the Amazon EFS console, and choose **Create file system**. 2. In the **Create file system** dialog box, enter a name for your file system, and choose your VPC. 3. Choose **Create** to create the file system. 4. Set up and configure your Amazon EFS file system.	Cloud administrator
Mount targets for the subnets.	1. Return to the Amazon EFS console, and choose **File systems**. The **File systems** page shows the Amazon EFS file systems in your account. 2. Choose the file system that you created, and choose **Manage** to display the **Availability Zones**. To add a mount target, choose **Add mount target**, and add the four private subnets that you created.	Cloud administrator

Task	Description	Skills required
Verify that the subnets are mounted as targets.	1. On the Amazon EFS console, choose **File systems**. 2. Choose **Network** to display the list of existing mount targets. Make sure that these include the four subnets that you created.	Cloud administrator

Create an S3 bucket

Task	Description	Skills required
Create an S3 bucket.	Open the Amazon S3 console and create an S3 bucket to store your application's static assets, if required.	Cloud administrator

Create a Secrets Manager secret

Task	Description	Skills required
Create an AWS KMS key to encrypt the Secrets Manager secret.	Open the AWS Key Management Service (AWS KMS) console and create a KMS key.	Cloud administrator
Create a Secrets Manager secret to store the Amazon RDS password.	1. Open the AWS Secrets Manager console, and create a new secret by choosing **Store a new secret**.	Cloud Administrator

Task	Description	Skills required
	2. Choose the KMS key that you created, and store your new secret.	

Create an Amazon RDS instance

Task	Description	Skills required
Create a DB subnet group.	1. Open the Amazon RDS console and choose **Subnet groups**. 2. Choose **Create DB subnet group**, and enter a name and description for your DB subnet group. 3. Choose the VPC that you created earlier, and choose the Availability Zones and subnets. Then choose **Create**.	Cloud administrator
Create an Amazon RDS instance.	Create and configure an Amazon RDS instance within the private subnets. Make sure that **Multi-AZ** is turned on for HA.	Cloud administrator
Load data to the Amazon RDS instance.	Load the relational data required by your applicati on into your Amazon RDS instance. This process will vary depending on your application's needs, as well as	Cloud administrator, DBA

Task	Description	Skills required
	how your database schema is defined and designed.	

Create the Amazon ECS components

Task	Description	Skills required
Create an ECS cluster.	1. Open the Amazon ECS console, and choose **Clusters**. 2. Choose **Create clusters**, and set up an ECS cluster according to your required specifications.	Cloud administrator
Create the Docker images.	Create the Docker images by following the instructions in the *Related resources* section.	Cloud administrator
Create Amazon ECR repositor ies.	1. On the Amazon ECR console, choose **Repositor ies**. 2. Choose **Create repository**, and enter a unique name for your repository. 3. Configure the repository according to your specifica tions, including AWS KMS encryption if required.	Cloud administrator, DevOps engineer
Authenticate your Docker client for the Amazon ECR repository.	To authenticate your Docker client for the Amazon ECR repository, run the "`aws ecr`	Cloud administrator

Task	Description	Skills required
	`get-login-password` command in the AWS CLI.	
Push the Docker images to the Amazon ECR repository.	1. Identify the Docker image you want to push, and run the `docker images` command in the AWS CLI. 2. Tag your image with the Amazon ECR registry, repository, and optional image tag name combinati on. 3. Push the Docker image by running the `docker push` command. 4. Repeat these steps for all required images.	Cloud administrator

Task	Description	Skills required
Create an Amazon ECS task definition.	A task definition is required to run Docker containers in Amazon ECS. 1. Return to the Amazon ECS console, choose **Task definitions**, and then choose **Create new task definition**. 2. On the **Select compatibilities** page, select the launch type that your task should use, and choose **Next step**. **Important** For help with setting up your task definition, see "Creating a task definition" in the *Related resources* section. Make sure you provide the Docker images that you pushed to Amazon ECR.	Cloud administrator

Task	Description	Skills required
Create an Amazon ECS service.	Create an Amazon ECS service by using the ECS cluster you created earlier. Make sure you choose Amazon EC2 as the launch type, and choose the task definition created in the previous step, as well as the target group of the Applicati on Load Balancer.	Cloud administrator

Create an Amazon EC2 Auto Scaling group

Task	Description	Skills required
Create a launch configuration.	Open the Amazon EC2 console, and create a launch configuration. Make sure that the user data has the code to allow the EC2 instances to join the desired ECS cluster. For an example of the code required, see the *Related resources* section.	Cloud administrator
Create an Amazon EC2 Auto Scaling group.	Return to the Amazon EC2 console and under **Auto Scaling**, choose **Auto Scaling groups**. Set up an Amazon EC2 Auto Scaling group. Make sure you choose the private subnets and launch configura tion that you created earlier.	Cloud administrator

Set up AWS PrivateLink

Task	Description	Skills required
Set up the AWS PrivateLink endpoint.	1. On the Amazon VPC console, create an AWS PrivateLink endpoint. 2. Associate this endpoint with the Network Load Balancer, which makes the application hosted on Amazon ECS available privately to customers. For more information, see the *Related resources* section.	Cloud administrator

Create a VPC endpoint

Task	Description	Skills required
Create a VPC endpoint.	Create a VPC endpoint for the AWS PrivateLink endpoint that you created earlier. The VPC endpoint Fully Qualified Domain Name (FQDN) will point to the AWS PrivateLink endpoint FQDN. This creates an elastic network interface to the VPC endpoint service that the DNS endpoints can access.	Cloud administrator

Create the Lambda function

Task	Description	Skills required
Create the Lambda function.	On the AWS Lambda console, create a Lambda function to update the Application Load Balancer IP addresses as targets for the Network Load Balancer. For more informati on on this, see the Using AWS Lambda to enable static IP addresses for Application Load Balancers blog post.	App developer

Related resources

Create the load balancers:

- Use a Network Load Balancer for Amazon ECS
- Create a Network Load Balancer
- Use an Application Load Balancer for Amazon ECS
- Create an Application Load Balancer

Create an Amazon EFS file system:

- Create an Amazon EFS file system
- Create mount targets in Amazon EFS

Create an S3 bucket:

- Create an S3 bucket

Create a Secrets Manager secret:

- Create keys in AWS KMS

- Create a secret in AWS Secrets Manager

Create an Amazon RDS instance:

- Create an Amazon RDS DB instance

Create the Amazon ECS components:

- Create an Amazon ECS cluster
- Create a Docker image
- Create an Amazon ECR repository
- Authenticate Docker with Amazon ECR repository
- Push an image to an Amazon ECR repository
- Create Amazon ECS task definition
- Create an Amazon ECS service

Create an Amazon EC2 Auto Scaling group:

- Create a launch configuration
- Create an Auto Scaling group using a launch configuration
- Bootstrap container instances with Amazon EC2 user data

Set up AWS PrivateLink:

- VPC endpoint services (AWS PrivateLink)

Create a VPC endpoint:

- Interface VPC endpoints (AWS PrivateLink)

Create the Lambda function:

- Create a Lambda function

Other resources:

- Using static IP addresses for Application Load Balancers
- Securely accessing services over AWS PrivateLink

Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer

Kirankumar Chandrashekar, Amazon Web Services

Summary

This pattern describes how to privately host a Docker container application on the Amazon Web Services (AWS) Cloud by using Amazon Elastic Container Service (Amazon ECS) with an AWS Fargate launch type, behind a Network Load Balancer, and access the application by using AWS PrivateLink. Amazon Relational Database Service (Amazon RDS) hosts the relational database for the application running on Amazon ECS with high availability (HA). You can use Amazon Elastic File System (Amazon EFS) if the application requires persistent storage.

This pattern uses a Fargate launch type for the Amazon ECS service running the Docker applications, with a Network Load Balancer at the front end. It can then be associated with a virtual private cloud (VPC) endpoint for access through AWS PrivateLink. This VPC endpoint service can then be shared with other VPCs by using their VPC endpoints.

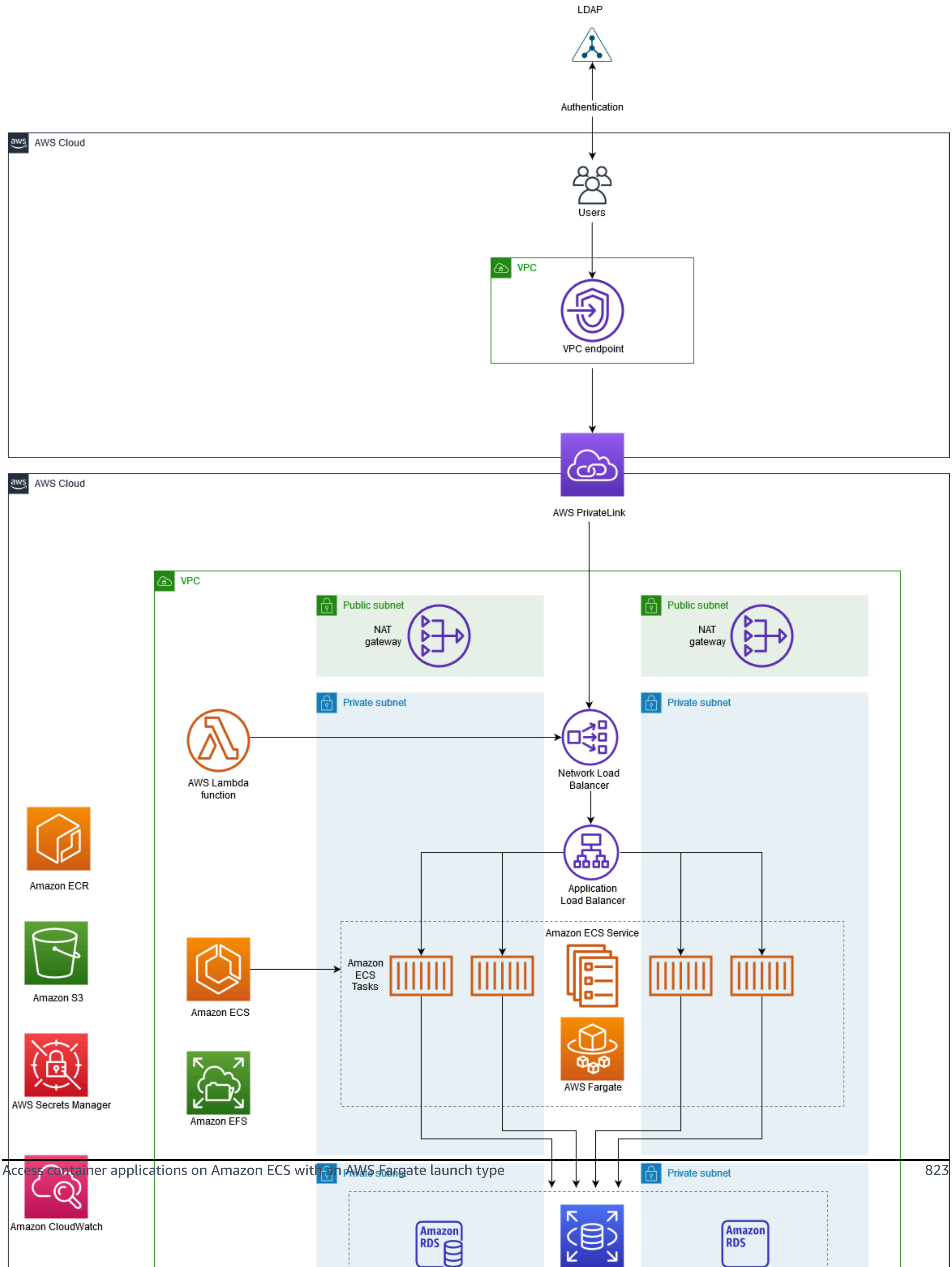
You can use Fargate with Amazon ECS to run containers without having to manage servers or clusters of Amazon Elastic Compute Cloud (Amazon EC2) instances. You can also use an Amazon EC2 Auto Scaling group instead of Fargate. For more information, see Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows
- Docker, installed and configured on Linux, macOS, or Windows
- An application running on Docker

Architecture



Technology stack

- Amazon CloudWatch
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- Amazon EFS
- Amazon RDS
- Amazon Simple Storage Service (Amazon S3)
- AWS Fargate
- AWS PrivateLink
- AWS Secrets Manager
- Application Load Balancer
- Network Load Balancer
- VPC

Automation and scale

- You can use AWS CloudFormation to create this pattern by using Infrastructure as Code.

Tools

AWS services

- Amazon Elastic Container Registry (Amazon ECR) is a managed AWS container image registry service that is secure, scalable, and reliable.
- Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage containers on a cluster.
- Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources.
- AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances.
- Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.

- Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in multiple Availability Zones.

Other tools

- Docker helps developers to easily pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

Epics

Create networking components

Task	Description	Skills required
Create a VPC.	1. Sign in to the AWS Management Console, and open the Amazon VPC console. Choose **Create VPC**, and choose **VPC and more**. 2. Enter a name for your VPC, and choose an appropriate CIDR block range. 3. Specify two Availability Zones, two public subnets, four private subnets. Two private subnets are for Amazon ECS tasks, and two private subnets are for Amazon RDS databases.	Cloud administrator

Task	Description	Skills required
	4. Specify one NAT gateway for each Availability Zone. 5. Choose Create **VPC**.	

Create the load balancers

Task	Description	Skills required
Create a Network Load Balancer.	1. Open the Amazon EC2 console, and choose the AWS Region that contains your VPC. 2. Under **Load balancing**, choose **Load balancers**, and choose **Create load balancer**. 3. Choose **Network Load Balancer**, and choose **Create**. 4. On the **Configure load balancer** page, configure your Network Load Balancer and listener. **Important:** Make sure you choose your Network Load Balancer's scheme as **Internal**. 5. Choose the applicable security settings, configure a security group and a target group. Choose **IP** as the **Target type** in the **Configure routing** section.	Cloud administrator

Task	Description	Skills required
	Make sure you do not register a target. 6. When you have configure d all the settings, choose **Next: Review**, and then choose **Create**. For help with this and other stories, see the *Related resources* section.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create an Application Load Balancer.</td><td>1. On the Amazon EC2 console, choose the same Region that contains your VPC.
2. Under Load balancing, choose Load balancers, and choose Create load balancer.
3. Choose Application Load Balancer, and choose Create.
4. Important
Configure your Application Load Balancer and its listener. Make sure you choose your Application Load Balancer's scheme as Internal.
5. Choose the applicable security settings, configure a security group and a target group. Choose IP as the Target type in the Configure routing section. Make sure you do not register a target.
6. When you have configure d all the settings, choose</td><td>Cloud administrator</td></tr>
</table>

Task	Description	Skills required
	Next: Review, and then choose **Create**.	

Create an Amazon EFS file system

Task	Description	Skills required
Create an Amazon EFS file system.	1. Open the Amazon EFS console, and choose **Create file system**. 2. In the **Create file system** dialog box, enter a name for your file system, and choose your VPC. 3. Choose **Create** to create the file system. 4. Set up and configure your Amazon EFS file system.	Cloud administrator
Mount targets for the subnets.	1. Return to the Amazon EFS console, and choose **File systems**. The **File systems** page shows the Amazon EFS file systems in your account. 2. Choose the file system that you created, and choose **Manage** to display the **Availability Zone**. 3. To add a mount target, choose **Add mount target**, and add the four private subnets that you created.	Cloud administrator

Task	Description	Skills required
Verify that the subnets are mounted as targets.	1. On the Amazon EFS console, choose **File systems**. 2. Choose **Network** to display the list of existing mount targets. Make sure that these include the four subnets that you created.	Cloud administrator

Create an S3 bucket

Task	Description	Skills required
Create an S3 bucket.	Open the Amazon S3 console and create an S3 bucket to store your application's static assets, if required.	Cloud administrator

Create a Secrets Manager secret

Task	Description	Skills required
Create an AWS KMS key to encrypt the Secrets Manager secret.	Open the AWS Key Management Service (AWS KMS) console and create a KMS key.	Cloud administrator
Create a Secrets Manager secret to store the Amazon RDS password.	1. Open the AWS Secrets Manager console, and create a new secret by choosing **Store a new secret**.	Cloud administrator

Task	Description	Skills required
	2. Choose the KMS key that you created, and store your new secret.	

Create an Amazon RDS instance

Task	Description	Skills required
Create a DB subnet group.	1. Open the Amazon RDS console, and choose **Subnet groups**. 2. Choose **Create DB subnet group**, and enter a name and description for your DB subnet group. 3. Choose the VPC that you created earlier, and choose the Availability Zones and subnets. Then choose **Create**.	Cloud administrator
Create an Amazon RDS instance.	Create and configure an Amazon RDS instance within the private subnets. Make sure that **Multi-AZ** is turned on for high availability (HA).	Cloud administrator
Load data to the Amazon RDS instance.	Load the relational data required by your applicati on into your Amazon RDS instance. This process will vary depending on your application's needs, as well as	DBA

Task	Description	Skills required
	how your database schema is defined and designed.	

Create the Amazon ECS components

Task	Description	Skills required
Create an ECS cluster.	1. Open the Amazon ECS console, and choose **Clusters**. 2. Choose **Create clusters**, and set up an ECS cluster according to your required specifications.	Cloud administrator
Create the Docker images.	Create the Docker images by following the instructions in the AWS documentation.	Cloud administrator
Create an Amazon ECR repository.	1. Open the Amazon ECR console, and choose **Repositories**. 2. Choose **Create repository**, and enter a unique name for your repository. 3. Configure the repository according to your specifica tions, including AWS KMS encryption if required.	Cloud administrator, DevOps engineer
Push the Docker images to the Amazon ECR repository.	1. Identify the Docker image you want to push, and run the `docker images` command in AWS CLI.	Cloud administrator

Task	Description	Skills required
	2. Tag your image with the Amazon ECR registry, repository, and optional image tag name combinati on. 3. Push the Docker image by running the `docker push` command. 4. Repeat these steps for all required images.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create an Amazon ECS task definition.</td><td>A task definition is required to run Docker containers in Amazon ECS.

1. Return to the Amazon ECS console, choose Task definitions, and then choose Create new task definition.
2. On the Select compatibi lities page, select the launch type that your task should use, and choose Next step.

Important
For help with setting up your task definitio n, see "Creating a task definition" in the Related resources section. Make sure you provide the Docker images that you pushed to Amazon ECR.</td><td>Cloud administrator</td></tr>
</table>

Task	Description	Skills required
Create an ECS service and choose Fargate as the launch type.	1. Create an Amazon ECS service by using the ECS cluster you created earlier. Make sure you choose **Fargate** as the launch type. 2. Choose the task definitio n created in the previous step, and choose the target group of the Application Load Balancer.	Cloud administrator

Set up AWS PrivateLink

Task	Description	Skills required
Set up the AWS PrivateLink endpoint.	1. Open the Amazon VPC console, and create an AWS PrivateLink endpoint. 2. Associate this endpoint with the Network Load Balancer, which makes the application hosted on Amazon ECS available privately to customers.	Cloud administrator

Create a VPC endpoint

Task	Description	Skills required
Create a VPC endpoint.	Create a VPC endpoint for the AWS PrivateLink endpoint that you created earlier. The VPC endpoint Fully Qualified	Cloud administrator

Task	Description	Skills required
	Domain Name (FQDN) will point to the AWS PrivateLink endpoint FQDN. This creates an elastic network interface to the VPC endpoint service that the Domain Name Service endpoints can access.	

Set the target

Task	Description	Skills required
Add the Application Load Balancer as a target.	To add the Application Load Balancer as a target for the Network Load Balancer, follow the instructions in the AWS documentation.	App developer

Related resources

Create the load balancers:

- Use a Network Load Balancer for Amazon ECS
- Create a Network Load Balancer
- Use an Application Load Balancer for Amazon ECS
- Create an Application Load Balancer

Create an Amazon EFS file system:

- Create an Amazon EFS file system
- Create mount targets in Amazon EFS

Create a Secrets Manager secret:

- Create keys in AWS KMS
- Create a secret in AWS Secrets Manager

Create an Amazon RDS instance:

- Create an Amazon RDS DB instance

Create the Amazon ECS components

- Create an Amazon ECR repository
- Authenticate Docker with Amazon ECR repository
- Push an image to an Amazon ECR repository
- Create Amazon ECS task definition
- Create an Amazon ECS service

Other resources:

- Securely accessing services over AWS PrivateLink

Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer

Kirankumar Chandrashekar, Amazon Web Services

Summary

This pattern describes how to privately host a Docker container application on Amazon Elastic Kubernetes Service (Amazon EKS) behind a Network Load Balancer, and access the application by using AWS PrivateLink. You can then use a private network to securely access services on the Amazon Web Services (AWS) Cloud.

The Amazon EKS cluster running the Docker applications, with a Network Load Balancer at the front end, can be associated with a virtual private cloud (VPC) endpoint for access through AWS PrivateLink. This VPC endpoint service can then be shared with other VPCs by using their VPC endpoints.

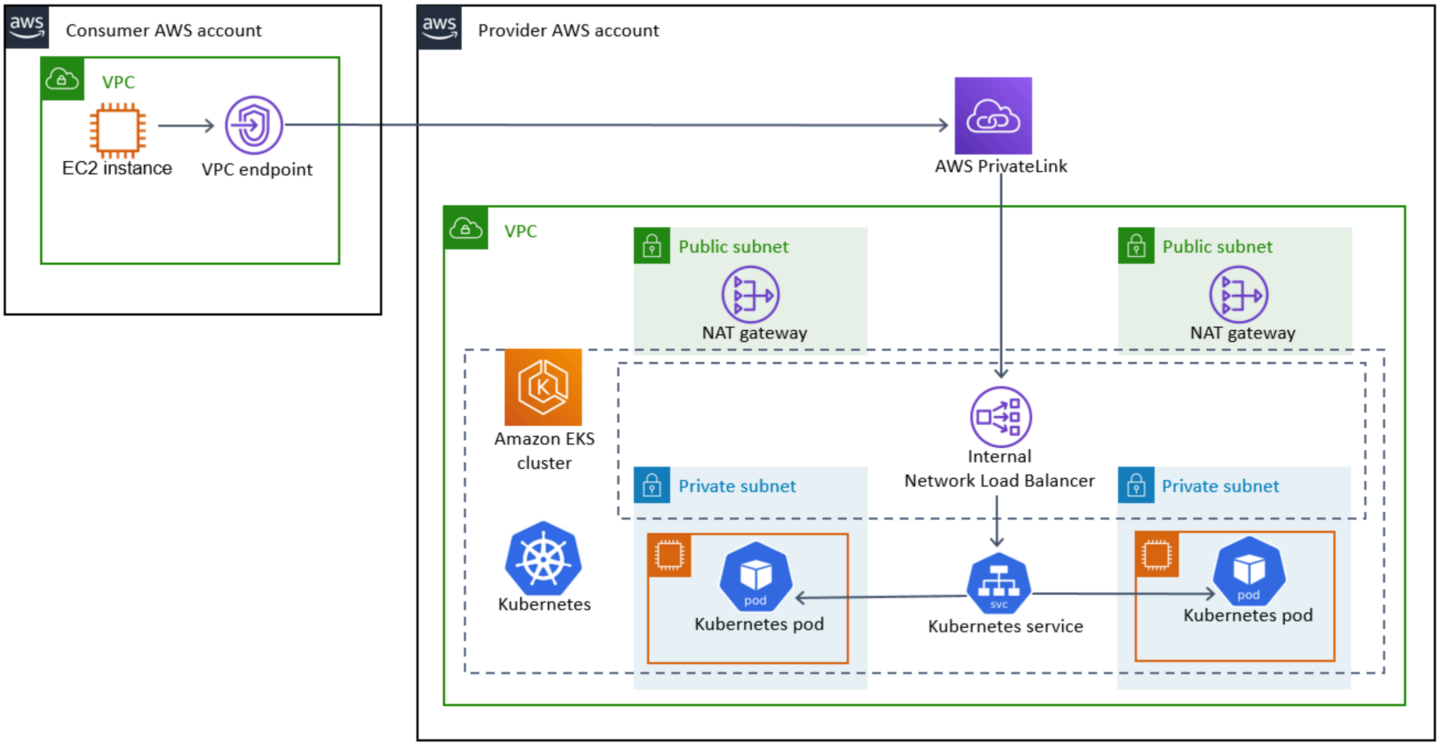
The setup described by this pattern is a secure way to share application access among VPCs and AWS accounts. It requires no special connectivity or routing configurations, because the connection between the consumer and provider accounts is on the global AWS backbone and doesn't traverse the public internet.

Prerequisites and limitations

Prerequisites

- Docker, installed and configured on Linux, macOS, or Windows.
- An application running on Docker.
- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows.
- An existing Amazon EKS cluster with tagged private subnets and configured to host applications. For more information, see Subnet tagging in the Amazon EKS documentation.
- Kubectl, installed and configured to access resources on your Amazon EKS cluster. For more information, see Installing kubectl in the Amazon EKS documentation.

Architecture



Technology stack

- Amazon EKS
- AWS PrivateLink
- Network Load Balancer

Automation and scale

- Kubernetes manifests can be tracked and managed on a Git-based repository, and deployed by using continuous integration and continuous delivery (CI/CD) in AWS CodePipeline.
- You can use AWS CloudFormation to create this pattern by using infrastructure as code (IaC).

Tools

- AWS CLI – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.

- [Elastic Load Balancing](#) – Elastic Load Balancing distributes incoming application or network traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses, in one or more Availability Zones.

- [Amazon EKS](#) – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.
- [Amazon VPC](#) – Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined.
- [Kubectl](#) – Kubectl is a command line utility for running commands against Kubernetes clusters.

Epics

Deploy the Kubernetes deployment and service manifest files

Task	Description	Skills required

Create the Kubernetes deployment manifest file.	Create a deployment manifest file by modifying the following sample file according to your requireme nts.
	```apiVersion: apps/v1```
	```kind: Deployment```
	```metadata:```
	``` name: sample-app```
	```spec:```
	``` replicas: 3```
	``` selector:```
	``` matchLabels:```
	``` app: nginx```
	``` template:```
	``` metadata:```
	``` labels:```
	``` app: nginx```
	``` spec:```
	``` containers:```
	``` - name: nginx```
		DevOps engineer

Task	Description	Skills required

	``
	            image:
	 public.ecr.aws/z9d
	2n7e1/nginx:1.19.5
	            ports:
	              - name: http
	                container
	Port: 80
	``
	
	Note
	
	This is a NGINX sample configuration file that is deployed by using the NGINX Docker image. For more information, see [How to use the official NGINX Docker image](#) in the Docker documentation.	

Deploy the Kubernetes deployment manifest file.	Run the following command to apply the deployment manifest file to your Amazon EKS cluster: `kubectl apply -f <your_deployment_f ile_name>`	DevOps engineer

Task	Description	Skills required

Create the Kubernetes service manifest file.	Create a service manifest file by modifying the following sample file according to your requirements.
	
	`apiVersion: v1`
	`kind: Service`
	`metadata:`
	`  name: sample-service`
	`  annotations:`
	`    service.beta.kubernetes.io/aws-load-balancer-type: nlb`
	`    service.beta.kubernetes.io/aws-load-balancer-internal: "true"`
	`spec:`
	`  ports:`
	`    - port: 80`
	`      targetPort: 80`
	`      protocol: TCP`
	`  type: LoadBalancer`
	`  selector:`
	`    app: nginx`
	
	Important
	Make sure that you included the following `annotations` to define an internal Network Load Balancer:
		DevOps engineer

Task	Description	Skills required
	`service.beta.kubernetes.io/aws-load-balancer-type: nlb service.beta.kubernetes.io/aws-load-balancer-internal: "true"`	
Deploy the Kubernetes service manifest file.	Run the following command to apply the service manifest file to your Amazon EKS cluster: `kubectl apply -f <your_service_file_name>`	DevOps engineer

Create the endpoints

Task	Description	Skills required
Record the Network Load Balancer's name.	Run the following command to retrieve the name of the Network Load Balancer: `kubectl get svc sample-service -o wide` Record the Network Load Balancer's name, which is required to create an AWS PrivateLink endpoint.	DevOps engineer
Create an AWS PrivateLink endpoint.	Sign in to the AWS Management Console, open	Cloud administrator

Task	Description	Skills required
	the Amazon VPC console, and then create an AWS PrivateLink endpoint. Associate this endpoint with the Network Load Balancer, this makes the application privately available to customers. For more information, see [VPC endpoint services (AWS PrivateLink)]() in the Amazon VPC documentation. **Important** If the consumer account requires access to the application, the consumer account's [AWS account ID]() must be added to the allowed principals list for the AWS PrivateLink endpoint configuration. For more information, see [Adding and removing permissions for your endpoint service]() in the Amazon VPC documentation.	

Task	Description	Skills required
Create a VPC endpoint.	On the Amazon VPC console, choose **Endpoint Services**, and then choose **Create Endpoint Service**. Create a VPC endpoint for the AWS PrivateLink endpoint. The VPC endpoint's fully qualified domain name (FQDN) points to the FQDN for the AWS PrivateLink endpoint. This creates an elastic network interface to the VPC endpoint service that the DNS endpoints can access.	Cloud administrator

Related resources

- Using the official NGINX Docker image
- Network load balancing on Amazon EKS
- Creating VPC endpoint services (AWS PrivateLink)
- Adding and removing permissions for your endpoint service

Activate mTLS in AWS App Mesh using AWS Private CA on Amazon EKS

Omar Kahil, Emmanuel Saliu, Muhammad Shahzad, and Andy Wong, Amazon Web Services

Summary

This pattern shows how to implement Mutual Transport Layer Security (mTLS) on Amazon Web Services (AWS) using certificates from AWS Private Certificate Authority (AWS Private CA) in AWS App Mesh. It uses the Envoy secret discovery service (SDS) API through the Secure Production Identity Framework for Everyone (SPIFFE). SPIFFE is a Cloud Native Computing Foundation (CNCF) open-source project with wide community support that provides fine-grained and dynamic workload identity management. To implement SPIFFE standards, use the SPIRE SPIFFE runtime environment.

Using mTLS in App Mesh offers two-way peer authentication, because it adds a layer of security over TLS and allows services in the mesh to verify the client that's making the connection. The client in the client-server relationship also provides an X.509 certificate during the session negotiation process. The server uses this certificate to identify and authenticate the client. This helps to verify if the certificate is issued by a trusted certificate authority (CA) and if the certificate is a valid one.

Prerequisites and limitations

Prerequisites

- An Amazon Elastic Kubernetes Service (Amazon EKS) cluster with self-managed or managed node groups
- App Mesh controller deployed on the cluster with SDS activated
- A private certificate from AWS Certificate Manager (ACM) that is issued by AWS Private CA

Limitations

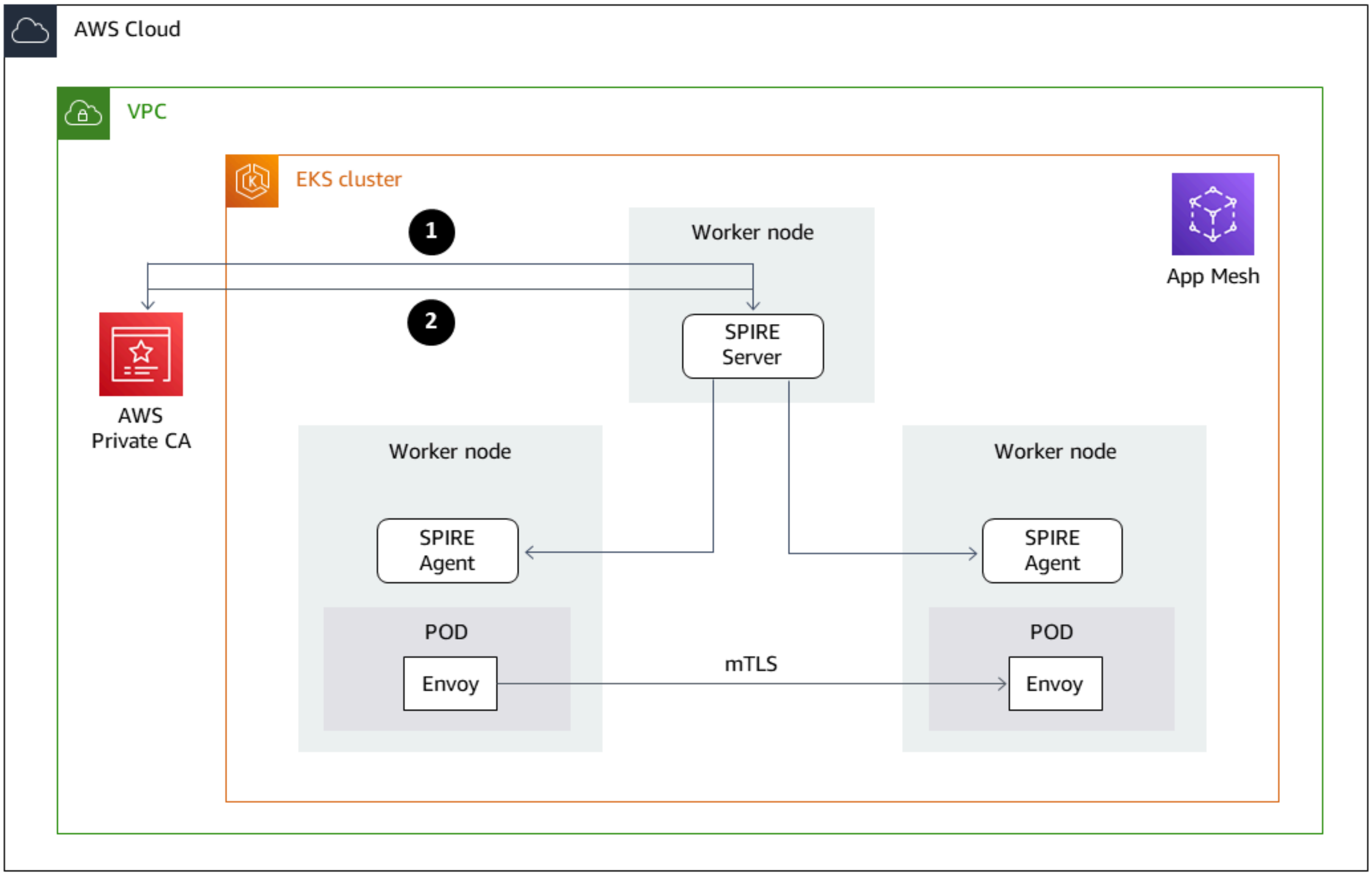
- SPIRE cannot be installed on AWS Fargate because the SPIRE Agent must be run as a Kubernetes DaemonSet.

Product versions

- AWS App Mesh Controller chart 1.3.0 or later

Architecture

The following diagram shows the EKS cluster with App Mesh in the VPC. The SPIRE server in one worker node communicates with the SPIRE Agents in other worker nodes, and with AWS Private CA. Envoy is used for mTLS communication between the SPIRE Agent worker nodes.



The diagram illustrates the following steps:

1. Certificate is issued.
2. Request cert signing and certificate.

Tools

AWS services

- [AWS Private CA](#) – AWS Private Certificate Authority (AWS Private CA) enables creation of private certificate authority (CA) hierarchies, including root and subordinate CAs, without the investment and maintenance costs of operating an on-premises CA.

- AWS App Mesh – AWS App Mesh is a service mesh that makes it easier to monitor and control services. App Mesh standardizes how your services communicate, giving you consistent visibility and network traffic controls for every service in an application.
- Amazon EKS – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

Other tools

- Helm – Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster. This pattern uses Helm to deploy AWS App Mesh Controller.
- AWS App Mesh Controller chart – AWS App Mesh Controller chart is used by this pattern to enable AWS App Mesh on Amazon EKS.

Epics

Set up the environment

Task	Description	Skills required
Set up App Mesh with Amazon EKS.	Follow base deployment steps that are provided in the repository.	DevOps engineer
Install SPIRE.	Install SPIRE on the EKS cluster by using spire_set up.yaml.	DevOps engineer
Install the AWS Private CA certificate.	Create and install a certifica te for your private root CA by following the instructions in the AWS documentation.	DevOps engineer
Grant permissions to the cluster node instance role.	To attach policies to the cluster node instance role, use the code that's in the	DevOps engineer

Task	Description	Skills required
	Additional information section.	
Add the SPIRE plugin for AWS Private CA.	To add the plugin to the SPIRE server configuration, use the code that's in the Additional information section. Replace the `certificate_authority_arn` Amazon Resource Name (ARN) to your private CA ARN. The signing algorithm used must be the same as the signing algorithm on the private CA. Replace `your_region` with your AWS Region. For more information about the plugin, see Server plugin: UpstreamAuthority "aws_pca".	DevOps engineer
Update bundle.cert.	After you create the SPIRE server, a `spire-bundle.yaml` file will be created. Change the `bundle.crt` value in the `spire-bundle.yaml` file from the private CA to the public certificate.	DevOps engineer

Deploy and register the workloads

Task	Description	Skills required
Register node and workload entries with SPIRE.	To register node and workload (services) with SPIRE Server, use the code in the [repository]().	DevOps engineer
Create a mesh in App Mesh with mTLS activated.	Create a new mesh in App Mesh with all the component s for your microservices application (for example, virtual service, virtual router, and virtual nodes).	DevOps engineer
Inspect the registered entries.	You can inspect the registere d entries for your nodes and workloads by running the following command. `kubectl exec -n spire spire-server-0 -- /opt/spire/bin/spire-server entry show` This will show the entries for the SPIRE Agents.	DevOps engineer

Verify mTLS traffic

Task	Description	Skills required
Verify mTLS traffic.	1. From the frontend service, send an HTTP header to the backend service, and verify a successful response	DevOps engineer

Task	Description	Skills required

	with the services that are registered in SPIRE.
	2. For mutual TLS authentication, you can inspect the `ssl.handshake` statistic by running the following command.
	`kubectl exec -it $POD -n $NAMESPACE -c envoy -- curl http://localhost:9901/stats | grep ssl.handshake`
	After running the previous command, you should see the listener `ssl.handshake` count, which will look similar to the following example:
	`listener.0.0.0.0_15000.ssl.handshake: 2`	

Task	Description	Skills required
Verify that certificates are being issued from AWS Private CA.	You can check that the plugins have been configure d correctly and certificates are being issued from your upstream private CA by viewing the logs in your SPIRE server. Run the following command. `kubectl logs spire-server-0 -n spire` Then view the logs that are produced. This code assumes that your server is named `spire-server-0` and is hosted in your spire namespace. You should see successful loading of the plugins and a connection being made to your upstream private CA.	DevOps engineer

Related resources

- Using mTLS with SPIFFE/SPIRE in AWS App Mesh on Amazon EKS
- Enabling mTLS in AWS App Mesh using SPIFFE/SPIRE in a multi-account Amazon EKS environment
- Walkthrough used in this pattern
- Server plugin: UpstreamAuthority "aws_pca"
- Quickstart for Kubernetes

Additional information

Attach permissions to the cluster node instance role

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ACMPCASigning",
            "Effect": "Allow",
            "Action": [
                "acm-pca:DescribeCertificateAuthority",
                "acm-pca:IssueCertificate",
                "acm-pca:GetCertificate",
                "acm:ExportCertificate"
            ],
            "Resource": "*"
        }
    ]
}
AWS Managed Policy: "AWSAppMeshEnvoyAccess"
```

Add the SPIRE plugin for ACM

```
Add the SPIRE plugin for ACM
Change certificate_authority_arn to your PCA ARN. The signing algorithm used must be
 the same as the signing algorithm on the PCA. Change your_region to the appropriate
 AWS Region.
UpstreamAuthority "aws_pca" {
    plugin_data {
        region = "your_region"
        certificate_authority_arn = "arn:aws:acm-pca:...."
        signing_algorithm = "your_signing_algorithm"
    }
}
```

Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch

Kirankumar Chandrashekar, Amazon Web Services

Summary

Backing up your PostgreSQL databases is an important task and can typically be completed with the pg_dump utility, which uses the COPY command by default to create a schema and data dump of a PostgreSQL database. However, this process can become repetitive if you require regular backups for multiple PostgreSQL databases. If your PostgreSQL databases are hosted in the cloud, you can also take advantage of the automated backup feature provided by Amazon Relational Database Service (Amazon RDS) for PostgreSQL as well. This pattern describes how to automate regular backups for Amazon RDS for PostgreSQL DB instances using the pg_dump utility.

Note: The instructions assume that you're using Amazon RDS. However, you can also use this approach for PostgreSQL databases that are hosted outside Amazon RDS. To take backups, the AWS Lambda function must be able to access your databases.

A time-based Amazon CloudWatch Events event initiates a Lambda function that searches for specific backup tags applied to the metadata of the PostgreSQL DB instances on Amazon RDS. If the PostgreSQL DB instances have the **bkp:AutomatedDBDump = Active** tag and other required backup tags, the Lambda function submits individual jobs for each database backup to AWS Batch.

AWS Batch processes these jobs and uploads the backup data to an Amazon Simple Storage Service (Amazon S3) bucket. This pattern uses a Dockerfile and an entrypoint.sh file to build a Docker container image that is used to make backups in the AWS Batch job. After the backup process is complete, AWS Batch records the backup details to an inventory table on Amazon DynamoDB. As an additional safeguard, a CloudWatch Events event initiates an Amazon Simple Notification Service (Amazon SNS) notification if a job fails in AWS Batch.

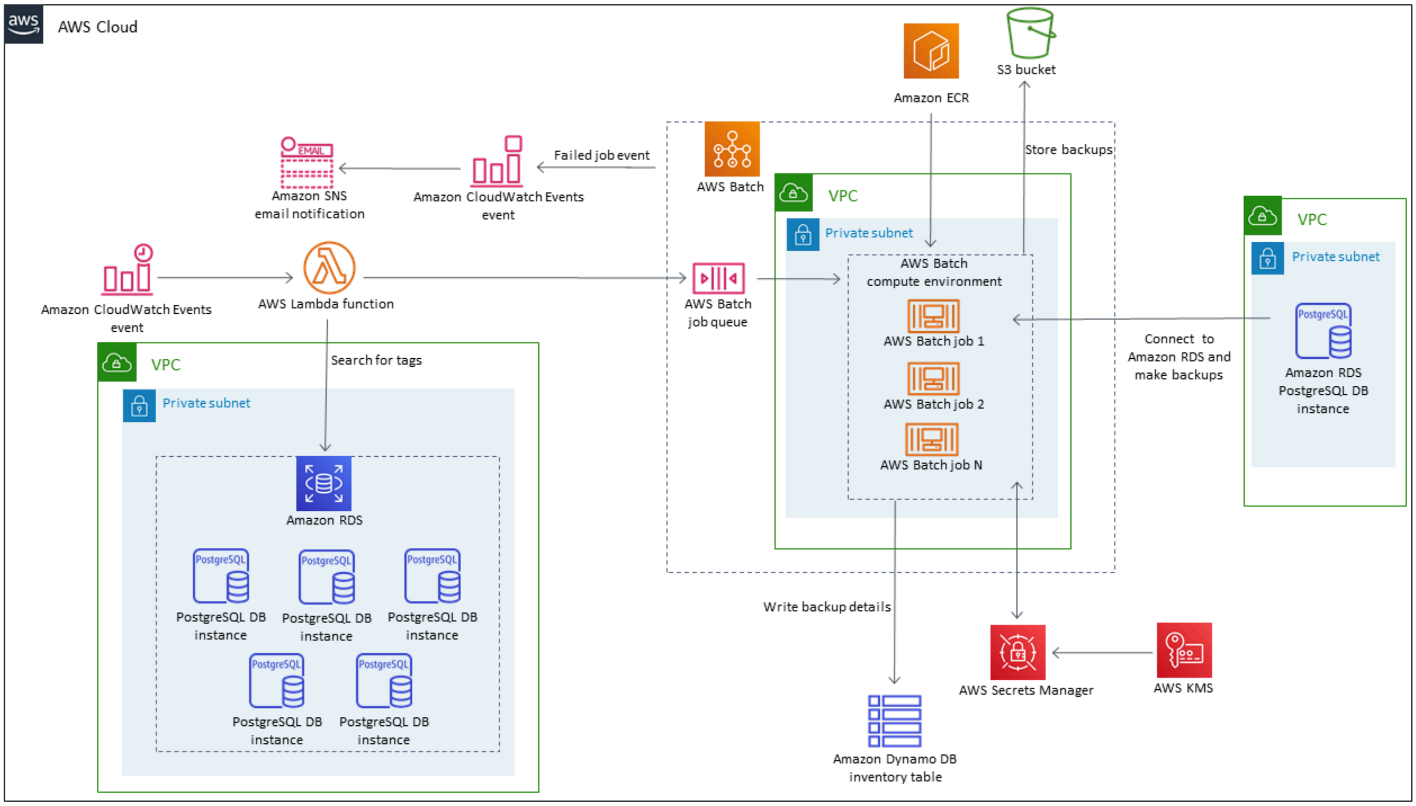
Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing managed or unmanaged compute environment. For more information, see Managed and unmanaged compute environments in the AWS Batch documentation.

- [AWS Command Line Interface (CLI) version 2 Docker image](), installed and configured.
- Existing Amazon RDS for PostgreSQL DB instances.
- An existing S3 bucket.
- [Docker](), installed and configured on Linux, macOS, or Windows.
- Familiarity with coding in Lambda.

Architecture



Technology stack

- Amazon CloudWatch Events
- Amazon DynamoDB
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon RDS
- Amazon SNS
- Amazon S3

- AWS Batch
- AWS Key Management Service (AWS KMS)
- AWS Lambda
- AWS Secrets Manager
- Docker

Tools

- Amazon CloudWatch Events – CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- Amazon DynamoDB – DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is a managed AWS container image registry service that is secure, scalable, and reliable.
- Amazon RDS – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- AWS Batch – AWS Batch helps you run batch computing workloads on the AWS Cloud.
- AWS KMS – AWS Key Management Service (AWS KMS) is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data.
- AWS Lambda – Lambda is a compute service that helps you run code without provisioning or managing servers.
- AWS Secrets Manager – Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Docker – Docker helps developers easily pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

Your PostgreSQL DB instances on Amazon RDS must have tags applied to their metadata. The Lambda function searches for tags to identify DB instances that should be backed up, and the following tags are typically used.

Tag	Description
bkp:AutomatedDBDump = Active	Identifies an Amazon RDS DB instance as a candidate for backups.
bkp:AutomatedBackupSecret = <secret_name >	Identifies the Secrets Manager secret that contains the Amazon RDS login credentials.
bkp:AutomatedDBDumpS3Bucket = <s3_bucket_name>	Identifies the S3 bucket to send backups to.
bkp:AutomatedDBDumpFrequency bkp:AutomatedDBDumpTime	Identify the frequency and times when databases should be backed up.
bkp:pgdumpcommand = <pgdump_command>	Identifies the databases for which the backups need to be taken.

Epics

Create an inventory table in DynamoDB

Task	Description	Skills required
Create a table in DynamoDB.	Sign in to the AWS Management Console, open the Amazon DynamoDB console, and create a table. For help with this and other stories, see the *Related resources* section.	Cloud administrator, Database administrator
Confirm that the table was created.	Run the `aws dynamodb describe-table --table-name <table-name> | grep TableStatus` command. If the table exists, the command will	Cloud administrator, Database administrator

Task	Description	Skills required
	return the `"TableSta tus": "ACTIVE",` result.	

Create an SNS topic for failed job events in AWS Batch

Task	Description	Skills required
Create an SNS topic.	Open the Amazon SNS console, choose **Topics**, and create an SNS topic with the name `JobFailedAlert` . Subscribe an active email address to the topic, and check your email inbox to confirm the SNS subscription email from AWS Notifications.	Cloud administrator
Create a failed job event rule for AWS Batch.	Open the Amazon CloudWatc h console, choose **Events**, and then choose **Create rule**. Choose **Show advanced options**, and choose **Edit**. For **Build a pattern that selects events for processin g by your targets**, replace the existing text with the "Failed job event" code from the *Additional information* section. This code defines a CloudWatch Events rule that initiates when AWS Batch has a `Failed` event.	Cloud administrator
Add event rule target.	In **Targets**, choose **Add targets**, and choose the	Cloud administrator

Task	Description	Skills required
	`JobFailedAlert` SNS topic. Configure the remaining details and create the Cloudwatch Events rule.	

Build a Docker image and push it to an Amazon ECR repository

Task	Description	Skills required
Create an Amazon ECR repository.	Open the Amazon ECR console and choose the AWS Region in which you want to create your repository. Choose **Repositories**, and then choose **Create repository**. Configure the repository according to your requirements.	Cloud administrator
Write a Dockerfile.	Sign in to Docker and use the "Sample Dockerfile" and "Sample entrypoint.sh file" from the *Additional information* section to build a Dockerfile.	DevOps engineer
Create a Docker image and push it to the Amazon ECR repository.	Build the Dockerfile into a Docker image and push it to the Amazon ECR repository. For help with this story, see the *Related resources* section.	DevOps engineer

Create the AWS Batch components

Task	Description	Skills required
Create an AWS Batch job definition.	Open the AWS Batch console and create a job definition that includes the Amazon ECR repository's Uniform Resource Identifier (URI) as the property `Image`.	Cloud administrator
Configure the AWS Batch job queue.	On the AWS Batch console, choose **Job queues**, and then choose **Create queue**. Create a job queue that will store jobs until AWS Batch runs them on the resources within your compute environment. Important: Make sure you write logic for AWS Batch to record the backup details to the DynamoDB inventory table.	Cloud administrator

Create and schedule a Lambda function

Task	Description	Skills required
Create a Lambda function to search for tags.	Create a Lambda function that searches for tags on your PostgreSQL DB instances and identifies backup candidate s. Make sure your Lambda function can identify the `bkp:AutomatedDBDump = Active` tag and all other	DevOps engineer

Task	Description	Skills required
	required tags. Important: The Lambda function must also be able to add jobs to the AWS Batch job queue.	
Create a time-based CloudWatch Events event.	Open the Amazon CloudWatch console and create a CloudWatch Events event that uses a cron expression to run your Lambda function on a regular schedule. Important: All scheduled events use the UTC time zone.	Cloud administrator

Test the backup automation

Task	Description	Skills required
Create an Amazon KMS key.	Open the Amazon KMS console and create a KMS key that can be used to encrypt the Amazon RDS credentials stored in AWS Secrets Manager.	Cloud administrator
Create an AWS Secrets Manager secret.	Open the AWS Secrets Manager console and store your Amazon RDS for PostgreSQL database credentials as a secret.	Cloud administrator
Add the required tags to the PostgreSQL DB instances.	**Important** Open the Amazon RDS console and	Cloud administrator

Task	Description	Skills required
	add tags to the PostgreSQL DB instances that you want to automatically back up. You can use the tags from the table in the *Tools* section. If you require backups from multiple PostgreSQL databases within the same Amazon RDS instance, then use `-d test:-d test1` as the value for the `bkp:pgdumpcommand` tag. `test` and `test1` are database names. Make sure that there is no space after the colon (:).	

Task	Description	Skills required
Verify the backup automation.	To verify the backup automation, you can either invoke the Lambda function or wait for the backup schedule to begin. After the backup process is complete, check that the DynamoDB inventory table has a valid backup entry for your PostgreSQL DB instances. If they match, then the backup automation process is successful.	Cloud administrator

Related resources

Create an inventory table in DynamoDB

- Create an Amazon DynamoDB table

Create an SNS topic for failed job events in AWS Batch

- Create an Amazon SNS topic
- Send SNS alerts for failed job events in AWS Batch

Build a Docker image and push it to an Amazon ECR repository

- Create an Amazon ECR repository
- Write a Dockerfile, create a Docker image, and push it to Amazon ECR

Create the AWS Batch components

- Create an AWS Batch job definition
- Configure your compute environment and AWS Batch job queue
- Create a job queue in AWS Batch

Create a Lambda function

- Create a Lambda function and write code
- Use Lambda with DynamoDB

Create a CloudWatch Events event

- Create a time-based CloudWatch Events event
- Use cron expressions in Cloudwatch Events

Test the backup automation

- Create an Amazon KMS key
- Create a Secrets Manager secret
- Add tags to an Amazon RDS instance

Additional information

Failed job event:

```
{
  "detail-type": [
    "Batch Job State Change"
  ],
  "source": [
    "aws.batch"
```

```
  ],
  "detail": {
    "status": [
      "FAILED"
    ]
  }
}
```

Sample Dockerfile:

```
FROM alpine:latest
RUN apk --update add py-pip postgresql-client jq bash && \
pip install awscli && \
rm -rf /var/cache/apk/*
ADD entrypoint.sh /usr/bin/
RUN chmod +x /usr/bin/entrypoint.sh
ENTRYPOINT ["entrypoint.sh"]
```

Sample entrypoint.sh file:

```
 #!/bin/bash
set -e
DATETIME=`date +"%Y-%m-%d_%H_%M"`
FILENAME=RDS_PostGres_dump_${RDS_INSTANCE_NAME}
FILE=${FILENAME}_${DATETIME}

aws configure --profile new-profile set role_arn arn:aws:iam::${TargetAccountId}:role/
${TargetAccountRoleName}
aws configure --profile new-profile set credential_source EcsContainer

echo "Central Account access provider IAM role is: "
aws sts get-caller-identity

echo "Target Customer Account access provider IAM role is: "
aws sts get-caller-identity --profile new-profile

securestring=$(aws secretsmanager get-secret-value --secret-id $SECRETID --output json
 --query 'SecretString' --region=$REGION --profile new-profile)

if [[ ${securestring} ]]; then
    echo "successfully accessed secrets manager and got the credentials"
    export PGPASSWORD=$(echo $securestring | jq --raw-output | jq -r '.DB_PASSWORD')
    PGSQL_USER=$(echo $securestring | jq --raw-output | jq -r '.DB_USERNAME')
```

```
    echo "Executing pg_dump for the PostGres endpoint ${PGSQL_HOST}"
    # pg_dump -h $PGSQL_HOST -U $PGSQL_USER -n dms_sample | gzip -9 -c  | aws s3 cp -
 --region=$REGION  --profile new-profile s3://$BUCKET/$FILE
    # in="-n public:-n private"
    IFS=':' list=($EXECUTE_COMMAND);
    for command in "${list[@]}";
      do
        echo $command;
        pg_dump -h $PGSQL_HOST -U $PGSQL_USER ${command} | gzip -9 -c  | aws s3 cp - --
region=$REGION --profile new-profile s3://${BUCKET}/${FILE}-${command}".sql.gz"
        echo $?;
        if  [[ $? -ne 0 ]]; then
            echo "Error occurred in database backup process. Exiting now....."
            exit 1
        else
            echo "Postgresql dump was successfully taken for the RDS endpoint
 ${PGSQL_HOST} and is uploaded to the following S3 location s3://${BUCKET}/${FILE}-
${command}.sql.gz"
            #write the details into the inventory table in central account
            echo "Writing to DynamoDB inventory table"
            aws dynamodb put-item --table-name ${RDS_POSTGRES_DUMP_INVENTORY_TABLE} --
region=$REGION --item '{ "accountId": { "S": "'"${TargetAccountId}"'" }, "dumpFileUrl":
 {"S": "'"s3://${BUCKET}/${FILE}-${command}.sql.gz"'" }, "DumpAvailableTime": {"S":
 "'"`date +"%Y-%m-%d::%H::%M::%S"` UTC"'"}}'
            echo $?
            if  [[ $? -ne 0 ]]; then
                echo "Error occurred while putting item to DynamoDb Inventory Table.
 Exiting now....."
                exit 1
            else
                echo "Successfully written to DynamoDb Inventory Table
 ${RDS_POSTGRES_DUMP_INVENTORY_TABLE}"
            fi
        fi
      done;
else
    echo "Something went wrong {$?}"
    exit 1
fi

exec "$@"
```

Automate deployment of Node Termination Handler in Amazon EKS by using a CI/CD pipeline

Sandip Gangapadhyay, Sandeep Gawande, Viyoma Sachdeva, Pragtideep Singh, and John Vargas, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

On the Amazon Web Services (AWS) Cloud, you can use AWS Node Termination Handler, an open-source project, to handle Amazon Elastic Compute Cloud (Amazon EC2) instance shutdown within Kubernetes gracefully. AWS Node Termination Handler helps to ensure that the Kubernetes control plane responds appropriately to events that can cause your EC2 instance to become unavailable. Such events include the following:

- EC2 instance scheduled maintenance
- Amazon EC2 Spot Instance interruptions
- Auto Scaling group scale in
- Auto Scaling group rebalancing across Availability Zones
- EC2 instance termination through the API or the AWS Management Console

If an event isn't handled, your application code might not stop gracefully. It also might take longer to recover full availability, or it might accidentally schedule work to nodes that are going down. The `aws-node-termination-handler` (NTH) can operate in two different modes: Instance Metadata Service (IMDS) or Queue Processor. For more information about the two modes, see the Readme file.

This pattern uses AWS CodeCommit, and it automates the deployment of NTH by using Queue Processor through a continuous integration and continuous delivery (CI/CD) pipeline.

> **Note**
>
> If you're using EKS managed node groups, you don't need the `aws-node-termination-handler`.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A web browser that is supported for use with the AWS Management Console. See the list of supported browsers.
- AWS Cloud Development Kit (AWS CDK) installed.
- `kubectl`, the Kubernetes command line tool, installed.
- `eksctl`, the AWS Command Line Interface (AWS CLI) for Amazon Elastic Kubernetes Service (Amazon EKS), installed.
- A running EKS cluster with version 1.20 or later.
- A self-managed node group attached to the EKS cluster. To create an Amazon EKS cluster with a self-managed node group, run the following command.

```
eksctl create cluster --managed=false --region <region> --name <cluster_name>
```

  For more information on `eksctl`, see the eksctl documentation.
- AWS Identity and Access Management (IAM) OpenID Connect (OIDC) provider for your cluster. For more information, see Creating an IAM OIDC provider for your cluster.

Limitations

- You must use an AWS Region that supports the Amazon EKS service.

Product versions

- Kubernetes version 1.20 or later
- `eksctl` version 0.107.0 or later
- AWS CDK version 2.27.0 or later

Architecture

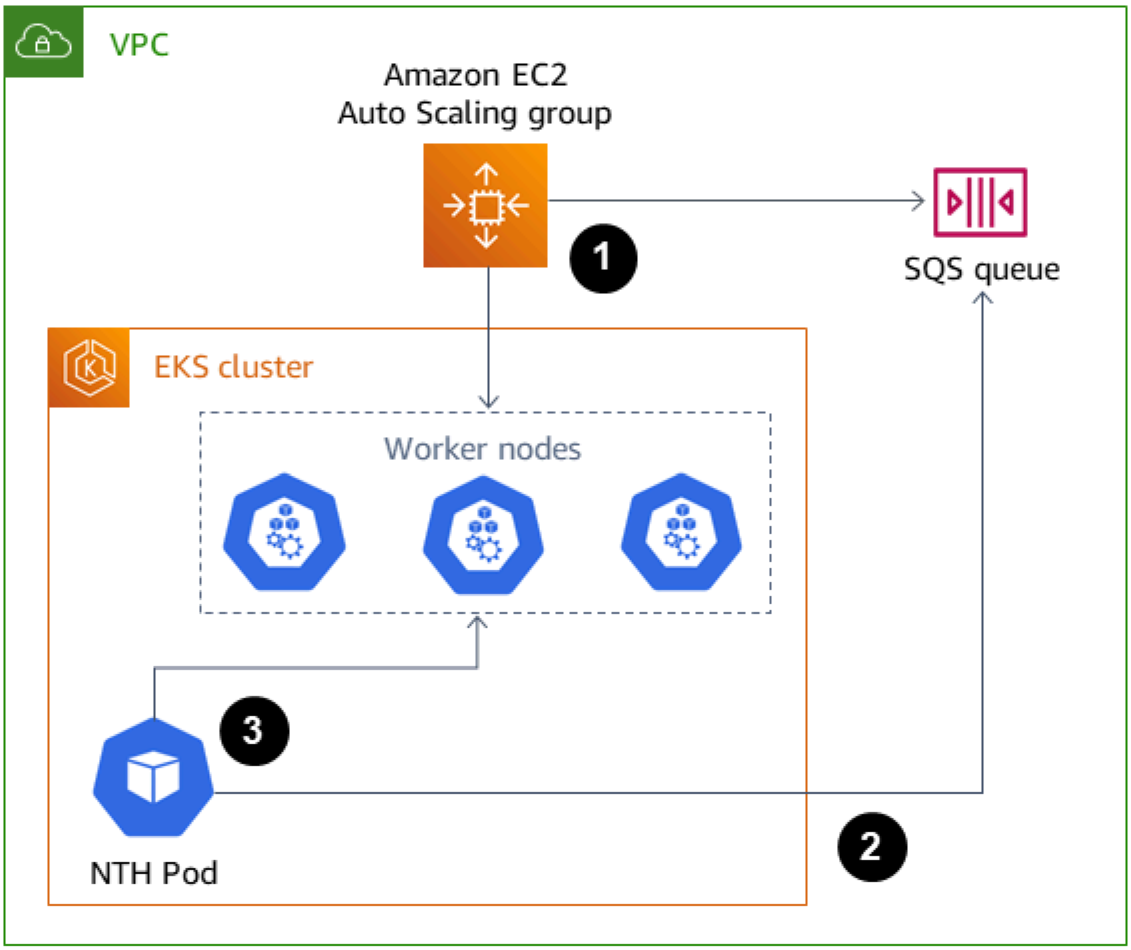
Target technology stack

- A virtual private cloud (VPC)

- An EKS cluster
- Amazon Simple Queue Service (Amazon SQS)
- IAM
- Kubernetes

Target architecture

The following diagram shows the high-level view of the end-to-end steps when the node termination is started.



The workflow shown in the diagram consists of the following high-level steps:

1. The automatic scaling EC2 instance terminate event is sent to the SQS queue.
2. The NTH Pod monitors for new messages in the SQS queue.
3. The NTH Pod receives the new message and does the following:

- Cordons the node so that new pod does not run on the node.
- Drains the node, so that the existing pod is evacuated
- Sends a lifecycle hook signal to the Auto Scaling group so that the node can be terminated.

Automation and scale

- Code is managed and deployed by AWS CDK, backed by AWS CloudFormation nested stacks.
- The Amazon EKS control plane runs across multiple Availability Zones to ensure high availability.
- For automatic scaling, Amazon EKS supports the Kubernetes Cluster Autoscaler and Karpenter.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- Amazon EC2 Auto Scaling helps you maintain application availability and allows you to automatically add or remove Amazon EC2 instances according to conditions you define.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.

Other tools

- kubectl is a Kubernetes command line tool for running commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.

Code

The code for this pattern is available in the deploy-nth-to-eks repo on GitHub.com. The code repo contains the following files and folders.

- `nth folder` – The Helm chart, values files, and the scripts to scan and deploy the AWS CloudFormation template for Node Termination Handler.
- `config/config.json` – The configuration parameter file for the application. This file contains all the parameters needed for CDK to be deployed.
- `cdk` – AWS CDK source code.
- `setup.sh` – The script used to deploy the AWS CDK application to create the required CI/CD pipeline and other required resources.
- `uninstall.sh` – The script used to clean up the resources.

To use the example code, follow the instructions in the *Epics* section.

Best practices

For best practices when automating AWS Node Termination Handler, see the following:

- EKS Best Practices Guides
- Node Termination Handler - Configuration

Epics

Set up your environment

Task	Description	Skills required
Clone the repo.	To clone the repo by using SSH (Secure Shell), run the following the command. `git clone git@github.com:aws-samples/deploy-nth-to-eks.git`	App developer, AWS DevOps, DevOps engineer

Task	Description	Skills required

	To clone the repo by using HTTPS, run the following the command.
	
	`git clone https://github.com/aws-samples/deploy-nth-to-eks.git`
	
	Cloning the repo creates a folder named `deploy-nth-to-eks`.
	
	Change to that directory.
	
	`cd deploy-nth-to-eks`	

Set the kubeconfig file.	Set your AWS credentials in your terminal and confirm that you have rights to assume the cluster role. You can use the following example code. `aws eks update-kubeconfig --name <Cluster_Name> --region <region>--role-arn <Role_ARN>`	AWS DevOps, DevOps engineer, App developer

Deploy the CI/CD pipeline

Task	Description	Skills required
Set up the parameters.	In the `config/config.json` file, set up the following required parameters. • `pipelineName` : The name of the CI/CD pipeline to be created by AWS CDK (for example, `deploy-nth-to-eks-pipeline`). AWS CodePipeline will create a pipeline that has this name. • `repositoryName` : The AWS CodeCommit repo to be created (for example, `deploy-nth-to-eks-repo`). AWS CDK will create this repo and set it as the source for the CI/CD pipeline. **Note** This solution will create this CodeCommit repo and the branch (provided in the following branch parameter).	App developer, AWS DevOps, DevOps engineer

Task	Description	Skills required
	• `branch`: The branch name in the repo (for example, `main`). A commit to this branch will initiate the CI/CD pipeline. • `cfn_scan_script` : The path of the script that will be used to scan the AWS CloudFormation template for NTH (`scan.sh`). This script exists in `nth` folder that will be part of the AWS CodeCommit repo. • `cfn_deploy_scrip t`: The path of the script that will be used to deploy the AWS CloudFormation template for NTH (`installApp.sh`). • `stackName` : The name of the CloudFormation stack to be deployed. • `eksClusterName` : The name of the existing EKS cluster. • `eksClusterRole` : The IAM role that will be used to access the EKS cluster for all Kubernetes API calls (for example, `clusteradmin`). Usually, this role is added in `aws-auth` `ConfigMap` .	

Task	Description	Skills required
	• `create_cluster_role` : To create the `eksClusterRole` IAM role, enter **yes**. If you want to provide an existing cluster role in the `eksClusterRole` parameter, enter **no**. • `create_iam_oidc_provider` : To create the IAM OIDC provider for your cluster, enter **yes**. If an IAM OIDC provider already exists, enter **no**. For more information, see [Creating an IAM OIDC provider for your cluster](#). • `AsgGroupName` : A comma-separated list of Auto Scaling group names that are part of the EKS cluster (for example, `ASG_Group_1,ASG_Group_2`). • `region`: The name of the AWS Region where the cluster is located (for example, `us-east-2`). • `install_cdk` : If AWS CDK isn't currently installed on the machine, enter **yes**. Run the `cdk --version` command to check whether the installed AWS CDK	

Task	Description	Skills required
	version is 2.27.0 or later. In that case, enter **no**. If you enter **yes**, the setup.sh script will run the `sudo npm install -g cdk@2.27.0` command to install AWS CDK on the machine. The script requires sudo permissio ns, so provide the account password when prompted.	
Create the CI/CD pipeline to deploy NTH.	Run the setup.sh script. `./setup.sh` The script will deploy the AWS CDK application that will create the CodeCommi t repo with example code, the pipeline, and CodeBuild projects based on the user input parameters in `config/config.json` file. This script will ask for the password as it installs npm packages with the sudo command.	App developer, AWS DevOps, DevOps engineer

Task	Description	Skills required
Review the CI/CD pipeline.	Open the AWS Managemen t Console, and review the following resources created in the stack. • CodeCommit repo with the contents of the `nth` folder • AWS CodeBuild project `cfn-scan`, which will scan the CloudFormation template for vulnerabilities. • CodeBuild project `Nth-Deploy` , which will deploy the AWS CloudFormation template and the correspon ding NTH Helm charts through the AWS CodePipel ine pipeline. • A CodePipeline pipeline to deploy NTH. After the pipeline runs successfully, Helm release `aws-node-terminati on-handler` is installed in the EKS cluster. Also, a Pod named `aws-node-termination-handler` is running in the `kube-syst em` namespace in the cluster.	App developer, AWS DevOps, DevOps engineer

Test NTH deployment

Task	Description	Skills required
Simulate an Auto Scaling group scale-in event.	To simulate an automatic scaling scale-in event, do the following: 1. On the AWS console, open the **EC2 console, and choose Auto Scaling Groups.** 2. Select the Auto Scaling group that has same name as the one provided in `config/config.json` , and choose **Edit**. 3. Decrease **Desired and Minimum Capacity** by 1. 4. Choose **Update**.	
Review the logs.	During the scale-in event, the NTH Pod will cordon and drain the correspon ding worker node (the EC2 instance that will be terminated as part of the scale-in event). To check the logs, use the code in the *Additional information* section.	App developer, AWS DevOps, DevOps engineer

Clean up

Task	Description	Skills required
Clean up all AWS resources.	To clean up the resources created by this pattern, run the following command. `./uninstall.sh` This will clean up all the resources created in this pattern by deleting the CloudFormation stack.	DevOps engineer

Troubleshooting

Issue	Solution

The npm registry isn't set correctly.	During the installation of this solution, the script installs npm install to download all the required packages. If, during the installation, you see a message that says "Cannot find module," the npm registry might not be set correctly. To see the current registry setting, run the following command.
	`npm config get registry`
	To set the registry with `https://registry.npmjs.org/` , run the following command.
	`npm config set registry https://registry.npmjs.org`

Issue	Solution
Delay SQS message delivery.	As part of your troubleshooting, if you want to delay the SQS message delivery to NTH Pod, you can adjust the SQS delivery delay parameter. For more information, see Amazon SQS delay queues.

Related resources

- AWS Node Termination Handler source code
- EC2 workshop
- AWS CodePipeline
- Amazon Elastic Kubernetes Service (Amazon EKS)
- AWS Cloud Development Kit
- AWS CloudFormation

Additional information

1. Find the NTH Pod name.

```
kubectl get pods -n kube-system |grep aws-node-termination-handler
aws-node-termination-handler-65445555-kbqc7   1/1     Running   0          26m
kubectl get pods -n kube-system |grep aws-node-termination-handler
aws-node-termination-handler-65445555-kbqc7   1/1     Running   0          26m
```

2. Check the logs. An example log looks like the following. It shows that the node has been cordoned and drained before sending the Auto Scaling group lifecycle hook completion signal.

```
kubectl -n kube-system logs aws-node-termination-handler-65445555-kbqc7
022/07/17 20:20:43 INF Adding new event to the event store
 event={"AutoScalingGroupName":"eksctl-my-cluster-target-nodegroup-
ng-10d99c89-NodeGroup-ZME36IGAP7O1","Description":"ASG Lifecycle Termination
 event received. Instance will be interrupted at 2022-07-17 20:20:42.702
 +0000 UTC \n","EndTime":"0001-01-01T00:00:00Z","EventID":"asg-lifecycle-
term-33383831316538382d353564362d343332362d613931352d383430666165636334333564","InProgress":fal
```

```
east-2.compute.internal","NodeProcessed":false,"Pods":null,"ProviderID":"aws:///us-
east-2c/i-0409f2a9d3085b80e","StartTime":"2022-07-17T20:20:42.702Z","State":""}
2022/07/17 20:20:44 INF Requesting instance drain event-id=asg-lifecycle-
term-33383831316538382d353564362d343332362d613931352d383430666165636334333564
 instance-id=i-0409f2a9d3085b80e kind=SQS_TERMINATE node-name=ip-192-168-75-60.us-
east-2.compute.internal provider-id=aws:///us-east-2c/i-0409f2a9d3085b80e
2022/07/17 20:20:44 INF Pods on node node_name=ip-192-168-75-60.us-
east-2.compute.internal pod_names=["aws-node-qchsw","aws-node-termination-
handler-65445555-kbqc7","kube-proxy-mz5x5"]
2022/07/17 20:20:44 INF Draining the node
2022/07/17 20:20:44 ??? WARNING: ignoring DaemonSet-managed Pods: kube-system/aws-node-
qchsw, kube-system/kube-proxy-mz5x5
2022/07/17 20:20:44 INF Node successfully cordoned and drained
 node_name=ip-192-168-75-60.us-east-2.compute.internal reason="ASG Lifecycle
 Termination event received. Instance will be interrupted at 2022-07-17 20:20:42.702
 +0000 UTC \n"
2022/07/17 20:20:44 INF Completed ASG Lifecycle Hook (NTH-K8S-TERM-HOOK) for instance
 i-0409f2a9d3085b80e
```

Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline

MAHESH RAGHUNANDANAN, Jomcy Pappachen, and James Radtke, Amazon Web Services

Summary

This pattern describes how to create a continuous integration and continuous delivery (CI/CD) pipeline that automatically builds and deploys a Java application with recommended DevSecOps practices to an Amazon Elastic Kubernetes Service (Amazon EKS) cluster on the AWS Cloud. This pattern uses a greeting application developed with a Spring Boot Java framework and that uses Apache Maven.

You can use this pattern's approach to build the code for a Java application, package the application artifacts as a Docker image, security scan the image, and upload the image as a workload container on Amazon EKS. This pattern's approach is useful if you want to migrate from a tightly coupled monolithic architecture to a microservices architecture. The approach also helps you to monitor and manage the entire lifecycle of a Java application, which ensures a higher level of automation and helps avoid errors or bugs.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2, installed and configured. For more information about this, see [Installing or updating to the latest version of the AWS CLI] in the AWS CLI documentation.

  AWS CLI version 2 must be configured with the same AWS Identity and Access Management (IAM) role that creates the Amazon EKS cluster, because only that role is authorized to add other IAM roles to the `aws-auth` `ConfigMap`. For information and steps to configure AWS CLI, see [Configuring settings] in the AWS CLI documentation.
- IAM roles and permissions with full access to AWS CloudFormation. For more information about this, see [Controlling access with IAM] in the AWS CloudFormation documentation.
- An existing Amazon EKS cluster, with details of the IAM role name and the Amazon Resource Name (ARN) of the IAM role for worker nodes in the EKS cluster.

- Kubernetes Cluster Autoscaler, installed and configured in your Amazon EKS cluster. For more information, see Scale cluster compute with Karpenter and Cluster Autoscaler in the Amazon EKS documentation.
- Access to code in the GitHub repository.

> **Important**
>
> AWS Security Hub is enabled as part of the AWS CloudFormation templates that are included in the code for this pattern. By default, after Security Hub is enabled, it comes with a 30–day free trial. After the trial, there is a cost associated with this AWS service. For more information about pricing, see AWS Security Hub pricing.

Product versions

- Helm version 3.4.2 or later
- Apache Maven version 3.6.3 or later
- BridgeCrew Checkov version 2.2 or later
- Aqua Security Trivy version 0.37 or later

Architecture

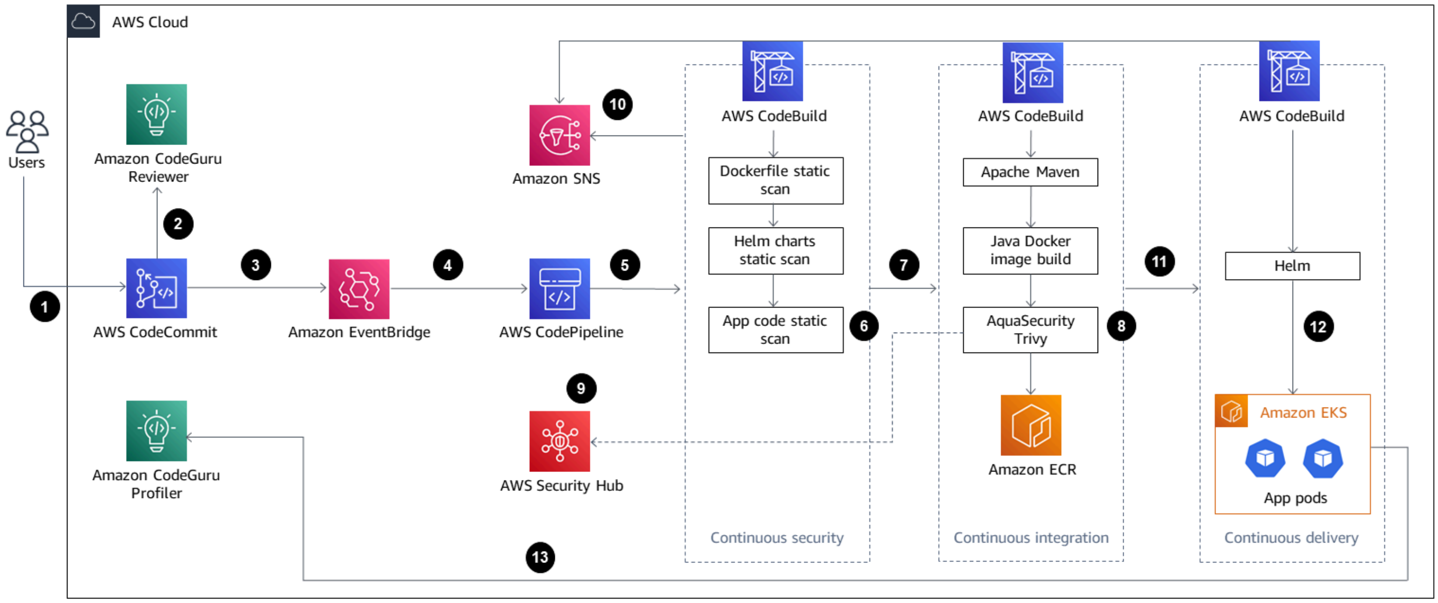
Technology stack

- AWS CodeBuild
- AWS CodeCommit

  Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more. However, this solution will work with any version control system (VCS) Git provider such as GitHub or GitLab with minimal changes.
- Amazon CodeGuru
- AWS CodePipeline
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon EKS

- Amazon EventBridge
- AWS Security Hub
- Amazon Simple Notification Service (Amazon SNS)

Target architecture



The diagram shows the following workflow:

1. The developer updates the Java application code in the base branch of the CodeCommit repository, which creates a pull request (PR).
2. As soon as the PR is submitted, Amazon CodeGuru Reviewer automatically reviews the code, analyzes it based on best practices for Java, and gives recommendations to the developer.
3. After the PR is merged to the base branch, an Amazon EventBridge event is created.
4. The EventBridge event initiates the CodePipeline pipeline, which starts.
5. CodePipeline runs the CodeSecurity Scan stage (continuous security).
6. AWS CodeBuild starts the security scan process in which the Dockerfile and Kubernetes deployment Helm files are scanned by using Checkov, and application source code is scanned based on incremental code changes. The application source code scan is performed by the CodeGuru Reviewer Command Line Interface (CLI) wrapper.
7. If the security scan stage is successful, the Build stage (continuous integration) is initiated.

8. In the Build stage, CodeBuild builds the artifact, packages the artifact to a Docker image, scans the image for security vulnerabilities by using Aqua Security Trivy, and stores the image in Amazon ECR.
9. The vulnerabilities detected from step 8 are uploaded to Security Hub for further analysis by developers or engineers. Security Hub provides an overview and recommendations for remediating the vulnerabilities.
10. Email notifications of sequential phases within the CodePipeline pipeline are sent through Amazon SNS.
11. After the continuous integration phases are complete, CodePipeline enters the Deploy stage (continuous delivery).
12. The Docker image is deployed to Amazon EKS as a container workload (pod) by using Helm charts.
13. The application pod is configured with Amazon CodeGuru Profiler agent, which sends the profiling data of the application (CPU, heap usage, and latency) to CodeGuru Profiler, which helps developers understand the behavior of the application.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.

  Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more
- Amazon CodeGuru Profiler collects runtime performance data from your live applications, and provides recommendations that can help you fine-tune your application performance.
- Amazon CodeGuru Reviewer uses program analysis and machine learning to detect potential defects that are difficult for developers to find and offers suggestions for improving your Java and Python code.

- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources, including AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Security Hub provides a comprehensive view of your security state on AWS. It also helps you check your AWS environment against security industry standards and best practices.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other services

- Helm is an open-source package manager for Kubernetes.
- Apache Maven is a software project management and comprehension tool.
- BridgeCrew Checkov is a static code analysis tool for scanning infrastructure as code (IaC) files for misconfigurations that might lead to security or compliance problems.
- Aqua Security Trivy is a comprehensive scanner for vulnerabilities in container images, file systems, and Git repositories, in addition to configuration issues.

Code

The code for this pattern is available in the GitHub aws-codepipeline-devsecops-amazoneks repository.

Best practices

- This pattern follows IAM security best practices to apply the principle of least privilege for IAM entities across all phases of the solution. If you want to extend the solution with additional AWS services or third-party tools, we recommend that you review the section on applying least-privilege permissions in the IAM documentation.
- If you have multiple Java applications, we recommend that you create separate CI/CD pipelines for each application.
- If you have a monolith application, we recommend that you break the application into microservices where possible. Microservices are more flexible, they make it easier to deploy applications as containers, and they provide better visibility into the overall build and deployment of the application.

Epics

Set up the environment

Task	Description	Skills required
Clone the GitHub repository.	To clone the repository, run the following command. `git clone https://github.com/aws-samples/aws-codepipeline-devsecops-amazoneks`	App developer, DevOps engineer
Create an S3 bucket and upload the code.	1. Sign in to the AWS Management Console, open the Amazon S3 console, and then create an S3 bucket in the AWS Region where you plan to deploy this solution. For more information, see Creating a bucket in the	AWS DevOps, Cloud administrator, DevOps engineer

Task	Description	Skills required

	Amazon S3 documentation.
	2. In the S3 bucket, create a folder named `code`.
	3. Navigate to where you cloned the repository. To create a compressed version of the entire code with the .zip extension (`cicdstack.zip`) and validate the .zip file, run the following commands in order.
	`cd aws-codepipeline-devsecops-amazoneks`
	`python -m zipfile -c cicdstack.zip *`
	`python -m zipfile -t cicdstack.zip`
	Note
	If the `python` command fails and states that Python was not found, use `python3` instead.
	4. Upload the `cicdstack.zip` file to the code folder that you previously created in the S3 bucket.	

Task	Description	Skills required
Create an AWS CloudFormation stack.	1. Open the [AWS CloudFormation console](#) and choose **Create stack**. 2. In **Specify template**, choose **Upload a template file**, upload the `cf_templates/codecommit_ecr.yaml` file, and then choose **Next**. 3. In **Specify stack details**, enter the stack name, and then provide the following input parameter values: • `CodeCommitRepositoryBranchName` : The name of the branch where your code will reside (the default is `main`) • `CodeCommitRepositoryName` : The name of the CodeCommitrepository to create • `CodeCommitRepositoryS3Bucket` : The name of the S3 bucket where you created the code folder • `CodeCommitRepositoryS3BucketObjKey` : `code/cicdstack.zip`	AWS DevOps, DevOps engineer

Task	Description	Skills required
	• `ECRRepositoryName` : The name of the Amazon ECR repository to create 4. Choose **Next**, use the default settings for the **Configure stack options**, and then choose **Next**. 5. In the **Review** section, verify the template and stack details, and then choose **Create stack**. The stack is then created, including the CodeCommit and Amazon ECR repositories. 6. Note the names of the CodeCommit and Amazon ECR repositories, which will be required to set up the Java CI/CD pipeline.	
Validate the CloudFormation stack deployment.	1. Under **Stacks** on the CloudFormation console, verify the status of the CloudFormation stack that you deployed. The status of the stack should be **CREATE COMPLETE**. 2. From the console, validate that the CloudFormation and Amazon ECR repositories have been provisioned and are ready.	AWS DevOps, DevOps engineer

Task	Description	Skills required
Delete the S3 bucket.	Empty and delete the S3 bucket that you created earlier. For more information, see [Deleting a bucket](#) in the Amazon S3 documentation.	AWS DevOps, DevOps engineer

Configure the Helm charts

Task	Description	Skills required

Configure the Helm charts of your Java application.	1. In the location where you cloned the GitHub repository, navigate to the folder `helm_charts/aws-proserve-java-greeting` . In this folder, the `values.dev.yaml` file contains information about Kubernetes resources configuration that you can modify for your container deployments to Amazon EKS. Update the Docker repository parameter by providing your AWS account ID, AWS Region, and Amazon ECR repository name.
	`image:`
	`  repository: <account-id>.dkr.ecr.<region>.amazon`
		DevOps engineer

Task	Description	Skills required

	`aws.com/<app-ecr-repo-name>`
	2. The Java pod's service type is set to `LoadBalancer` .
	`service:`
	`  type: LoadBalancer`
	`  port: 80`
	`  targetPort: 8080`
	`  path: /hello`
	`  initialDelaySeconds: 60`
	`  periodSeconds: 30`
	To use a different service (for example, `NodePort`), you can change this parameter. For more information, see the [Kubernetes documentation](#).
	3. You can activate the [Kubernetes Horizontal Pod Autoscaler](#) by changing the `autoscaling` parameter to `enabled: true`.
	`autoscaling:`
	`  enabled: true`
	`  minReplicas: 1`
	`  maxReplicas: 100`
	`  targetCPUUtilizationPercentage: 80`
	`  # targetMemoryUtilizationPercentage: 80`	

Task	Description	Skills required
	4. You can enable different features for the Kubernetes workloads by changing the values in the `values.<ENV>.yaml` file, where <ENV> is your development, production, UAT, or QA environment.	

Validate Helm charts for syntax errors.	1. From the terminal, verify that Helm v3 is installed in your local workstation by running the following command.
	`helm --version`
	If Helm v3 isn't installed, install it.
	2. In the terminal, navigate to the Helm charts directory (`helm_charts/aws-proserve-java-greeting`), and run the following command.
	`helm lint . -f values.dev.yaml`
	This will check the Helm charts for any syntax errors.
		DevOps engineer

Set up the Java CI/CD pipeline

Task	Description	Skills required
Create the CI/CD pipeline.	1. Open the [AWS CloudFormation console](), and choose **Create stack**. 2. In **Specify template**, choose **Upload a template file**, upload the `cf_templates/build_deployment.yaml` template, and then choose **Next**. 3. In **Specify stack details**, specify the **Stack name**, and then provide the following values for the input parameters: • `CodeBranchName` : Branch name of the CodeCommit repository where your code resides • `EKSClusterName` : Name of your EKS cluster (not the `EKSCluster` ID) • `EKSCodeBuildAppName` : Name of the app Helm chart (`aws-proserve-java-greeting`) • `EKSWorkerNodeRoleARN` : ARN of the IAM role	AWS DevOps

Task	Description	Skills required
	assigned to the Amazon EKS worker nodes • `EKSWorkerNodeRoleName` : Name of the IAM role assigned to the Amazon EKS worker nodes • `EcrDockerRepository` : Name of the Amazon ECR repository where the Docker images of your code will be stored • `EmailRecipient` : Email address where build notifications should be sent • `EnvType`: Environment (for example, dev, test, or prod) • `SourceRepoName` : Name of the CodeCommit repository where your code resides 4. Choose **Next**. Use the default settings in **Configure stack options**, and then choose **Next**. 5. In the **Review** section, verify the CloudFormation template and stack details, and then choose **Next**. 6. Choose **Create stack**.	

Task	Description	Skills required
	7. During the CloudFormation stack deployment, the owner of the email address that you provided in the parameters will receive a message to subscribe to an SNS topic. To subscribe to Amazon SNS, the owner must choose the link in the message. 8. After the stack is created, open the **Outputs** tab of the stack, and then record the ARN value for the `EksCodeBuildkubeRoleARN` output key. This IAM ARN value will be required later when you provide permissions for the CodeBuild IAM role to deploy workloads in the Amazon EKS cluster.	

Activate integration between Security Hub and Aqua Security

Task	Description	Skills required
Turn on Aqua Security integration.	This step is required for uploading the Docker image vulnerability findings reported by Trivy to Security Hub. Because AWS CloudFormation doesn't support Security	AWS administrator, DevOps engineer

Task	Description	Skills required
	Hub integrations, this process must be done manually. 1. Open the [AWS Security Hub console](), and navigate to **Integrations**. 2. Search for Aqua Security, and select **Aqua Security: Aqua Security**. 3. Choose **Accept findings**.	

Configure CodeBuild to run Helm or kubectl commands

Task	Description	Skills required
Allow CodeBuild to run Helm or kubectl commands in the Amazon EKS cluster.	For CodeBuild to be authenticated to use Helm or `kubectl` commands with the Amazon EKS cluster, you must add the IAM roles to the `aws-auth ConfigMap` . In this case, add the ARN of the IAM role `EksCodeBuildkubeRoleARN` , which is the IAM role created for the CodeBuild service to access the Amazon EKS cluster and deploy workloads on it. This is a one-time activity. **Important** The following procedure must be	DevOps

Task	Description	Skills required

	completed before the deployment approval stage in CodePipeline.
	
	1. Open the `cf_templates/kube_aws_auth_configmap_patch.sh` shell script in your Amazon Linux or macOS environment.
	2. Authenticate to the Amazon EKS cluster by running the following command.
	`aws eks --region <aws-region> update-kubeconfig --name <eks-cluster-name>`
	3. Run the shell script by using the following command, replacing `<rolearn-eks-codebuild-kubectl>` with the ARN value of `EksCodeBuildkubeRoleARN` that you recorded earlier.
	`bash cf_templates/kube_aws_auth_configmap_patch.sh <rolearn-eks-codebuild-kubectl>`	

Task	Description	Skills required
	The `aws_auth ConfigMap` is configured, and access is granted.	

Validate the CI/CD pipeline

Task	Description	Skills required

Verify that the CI/CD pipeline automatically initiates.	1. The CodeSecurity Scan stage in the pipeline will usually fail if Checkov detects vulnerabilities in the Dockerfile or Helm charts. However, the purpose of this example is to establish a process of identifying potential security vulnerabilities rather than fixing it through the CI/CD pipeline, typically a DevSecOps process. In the file `buildspec/buildspec_secscan.yaml`, the `checkov` command uses the `--soft-fail` flag to avoid pipeline failure.
	`- echo -e "\n Running Dockerfile Scan"`
	`- checkov -f code/app/Dockerfile --framework dockerfile --soft-fail --summary-position bottom`
		DevOps

Task	Description	Skills required
	``` - echo -e "\n Running Scan of Helm Chart files" - cp -pv helm_charts/$EKS_CODEBUILD_APP_NAME/values.dev.yaml helm_charts/$EKS_CODEBUILD_APP_NAME/values.yaml - checkov -d helm_charts/$EKS_CODEBUILD_APP_NAME --framework helm --soft-fail --summary-position bottom - rm -rfv helm_charts/$EKS_CODEBUILD_APP_NAME/values.yaml ``` For the pipeline to fail when vulnerabilities are reported for the Dockerfile and Helm charts, the `--soft-fail` option must be removed from the `checkov` command. Developers or engineers can then fix the vulnerabilities and commit the changes to the CodeCommit source code repository. 2. Similar to CodeSecurity Scan, the Build stage uses Aqua Security Trivy to identify `HIGH` and `CRITICAL` Docker image	

Task	Description	Skills required
	vulnerabilities before pushing the application to Amazon ECR. `- AWS_REGION=$AWS_DEFAULT_REGION AWS_ACCOUNT_ID=$AWS_ACCOUNT_ID trivy -d image --no-progress --ignore-unfixed --exit-code 0 --severity HIGH,CRITICAL --format template --template "@securityhub/asff.tpl" -o securityhub/report.asff $AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/$IMAGE_REPO_NAME:$CODEBUILD_RESOLVED_SOURCE_VERSION` In this example, the pipeline doesn't fail when Docker image vulnerabilities are reported, because the `trivy` command in the `buildspec/buildspec.yml` file includes the flag `--exit-code` with a value of`0`. For the pipeline to fail when `HIGH` and `CRTICAL` vulnerabilities are reported, change the value of `--exit-code` to 1. Developers or engineers	

Task	Description	Skills required
	can then fix the vulnerabilities and commit the changes to the CodeCommit source code repository. 3. Docker image vulnerabilities reported by Aqua Security Trivy are uploaded to Security Hub. On the Security Hub console, navigate to **Findings**. Filter the findings with **Record State = Active** and **Product = Aqua Security**. This lists the Docker image vulnerabilities in Security Hub. It can take 15 minutes to an hour for vulnerabilities to appear in Security Hub. For more information about starting the pipeline by using CodePipeline, see [Start a pipeline in]CodePipeline, [Start a pipeline manually], and [Start a pipeline on a schedule] in the CodePipeline documentation.	

Task	Description	Skills required
Approve the deployment.	1. After the build phase is complete, there is a deployment approval gate. The reviewer or a release manager should inspect the build, and, if all requirements are met, approve it. This is the recommended approach for teams that use continuous delivery for application deployment. 2. After approval, the pipeline initiates the Deploy stage. 3. After the Deploy stage is successful, the CodeBuild log for this stage provides the URL of the application. Use the URL to validate the readiness of the applicati on.	DevOps

Task	Description	Skills required
Validate application profiling.	After the deployment is complete and the application pod is deployed in Amazon EKS, the Amazon CodeGuru Profiler agent that is configured in the application will try to send profiling data of the application (CPU, heap summary, latency, and bottlenecks) to CodeGuru Profiler. For the initial deployment of an application, CodeGuru Profiler takes about 15 minutes to visualize the profiling data.	AWS DevOps

Related resources

- AWS CodePipeline documentation
- Scanning images with Trivy in an AWS CodePipeline (AWS blog post)
- Improving your Java applications using Amazon CodeGuru Profiler (AWS blog post)
- AWS Security Finding Format (ASFF) syntax
- Amazon EventBridge event patterns
- Helm upgrade

Additional information

- AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more. This solution will also work with any version control system (VCS) Git provider such as GitHub or GitLab with minimal changes.

- CodeGuru Profiler should not be confused with the AWS X-Ray service in terms of functionality. We recommend that you use CodeGuru Profiler to identify the most expensive lines of codes that might cause bottlenecks or security issues, and fix them before they become a potential risk. The X-Ray service is for application performance monitoring.
- In this pattern, event rules are associated with the default event bus. If needed, you can extend the pattern to use a custom event bus.
- This pattern uses CodeGuru Reviewer as a static application security testing (SAST) tool for application code. You can also use this pipeline for other tools, such as SonarQube or Checkmarx. You can add the scan setup instructions for any of these tools to `buildspec/buildspec_secscan.yaml` to replace the CodeGuru scan instructions.

Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS

Durga Prasad Cheepuri, Amazon Web Services

Summary

This pattern provides code samples and steps to create an Amazon Elastic Container Service (Amazon ECS) task definition that runs on Amazon Elastic Compute Cloud (Amazon EC2) instances in the Amazon Web Services (AWS) Cloud, while using Amazon Elastic File System (Amazon EFS) to mount a file system on those EC2 instances. Amazon ECS tasks that use Amazon EFS automatically mount the file systems that you specify in the task definition and make these file systems available to the task's containers across all Availability Zones in an AWS Region.

To meet your persistent storage and shared storage requirements, you can use Amazon ECS and Amazon EFS together. For example, you can use Amazon EFS to store persistent user data and application data for your applications with active and standby ECS container pairs running in different Availability Zones for high availability. You can also use Amazon EFS to store shared data that can be accessed in parallel by ECS containers and distributed job workloads.

To use Amazon EFS with Amazon ECS, you can add one or more volume definitions to a task definition. A volume definition includes an Amazon EFS file system ID, access point ID, and a configuration for AWS Identity and Access Management (IAM) authorization or Transport Layer Security (TLS) encryption in transit. You can use container definitions within task definitions to specify the task definition volumes that get mounted when the container runs. When a task that uses an Amazon EFS file system runs, Amazon ECS ensures that the file system is mounted and available to the containers that need access to it.

Prerequisites and limitations

Prerequisites

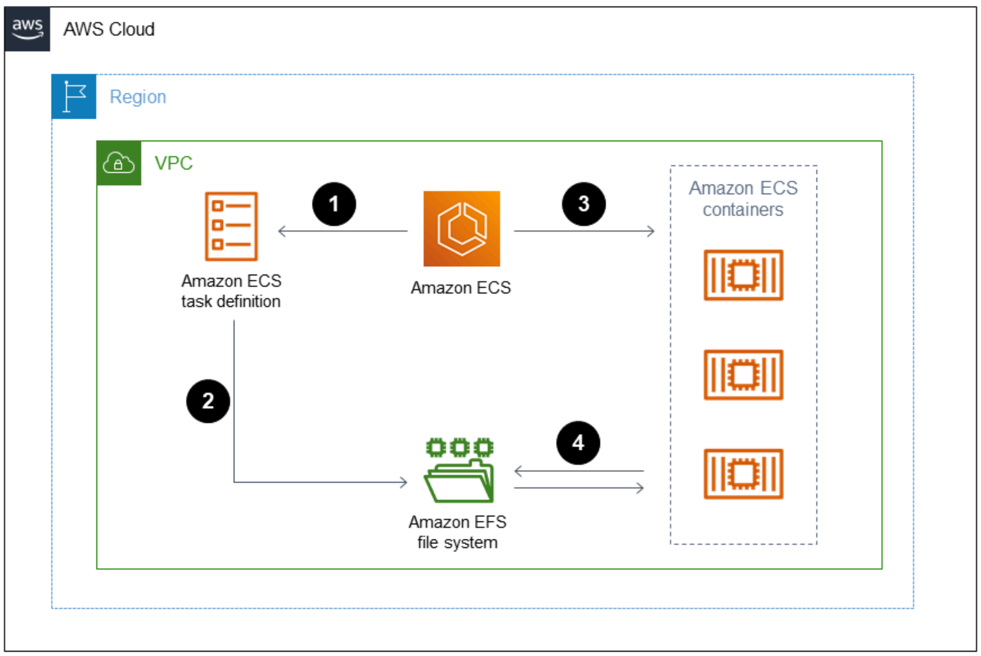
- An active AWS account
- A virtual private cloud (VPC) with a virtual private network (VPN) endpoint or a router
- (Recommended) Amazon ECS container agent 1.38.0 or later for compatibility with Amazon EFS access points and IAM authorization features (For more information, see the AWS blog post New for Amazon EFS – IAM Authorization and Access Points.)

Limitations

- Amazon ECS container agent versions earlier than 1.35.0 don't support Amazon EFS file systems for tasks that use the EC2 launch type.

Architecture

The following diagram shows an example of an application that uses Amazon ECS to create a task definition and mount an Amazon EFS file system on EC2 instances in ECS containers.



The diagram shows the following workflow:

1. Create an Amazon EFS file system.
2. Create a task definition with a container.
3. Configure the container instances to mount the Amazon EFS file system. The task definition references the volume mounts, so the container instance can use the Amazon EFS file system. ECS tasks have access to the same Amazon EFS file system, regardless of which container instance those tasks are created on.
4. Create an Amazon ECS service with three instances of the task definition.

Technology stack

- Amazon EC2
- Amazon ECS
- Amazon EFS

Tools

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of EC2 instances that you manage.
- Amazon EFS – Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources.
- AWS CLI – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

Epics

Create an Amazon EFS file system

Task	Description	Skills required
Create an Amazon EFS file system by using the AWS Management Console.	1. **Note** Create an Amazon EFS file system and choose the VPC that includes	AWS DevOps

Task	Description	Skills required
	your containers. : If you use a different VPC, set up a VPC peering connection. 2. Note the file system ID.	

Create an Amazon ECS task definition by using either an Amazon EFS file system or the AWS CLI

Task	Description	Skills required
Create a task definition using an Amazon EFS file system.	Create a task definition by using the new Amazon ECS console or classic Amazon ECS console with the following configurations: • If you use the new console, choose **Amazon EC2 instances** for **App environment**. If you use the classic console, choose **EC2** as the launch type. • Add a volume. Enter a name for the volume, choose **EFS** for volume type, and then choose the file system ID that you noted earlier. For the root directory, choose the Amazon EFS file system path that you want to host on the Amazon ECS container host.	AWS DevOps

Task	Description	Skills required

Create a task definition using the AWS CLI.	1. To create a JSON template with input parameter placeholders for your task definition, run the following command:
	`aws ecs register-task-definition --generate-cli-skeleton`
	2. To create the task definition with the JSON template, run the following command:
	`aws ecs register-task-definition --cli-input-json file://<path_to_your_json_file>`
	3. **Note** Enter the input parameters in your JSON template based on the `task_definition_parameters.json` file (attached). : For more information on input parameters, see [Task definition parameters](#) (Amazon ECS
		AWS DevOps

Task	Description	Skills required
	documentation) and register-task-definition (AWS CLI Command Reference).	

Related resources

- Amazon ECS task definitions
- Amazon EFS volumes

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Deploy Lambda functions with container images

Ram Kandaswamy, Amazon Web Services

Summary

AWS Lambda supports containers images as a deployment model. This pattern shows how to deploy Lambda functions through container images.

Lambda is a serverless, event-driven compute service that you can use to run code for virtually any type of application or backend service without provisioning or managing servers. With container image support for Lambda functions, you get the benefits of up to 10 GB of storage for your application artifact and the ability to use familiar container image development tools.

The example in this pattern uses Python as the underlying programming language, but you can use other languages, such as Java, Node.js, or Go. For the source, consider a Git-based system such as GitHub, GitLab, or Bitbucket, or use Amazon Simple Storage Service (Amazon S3).

Prerequisites and limitations

Prerequisites

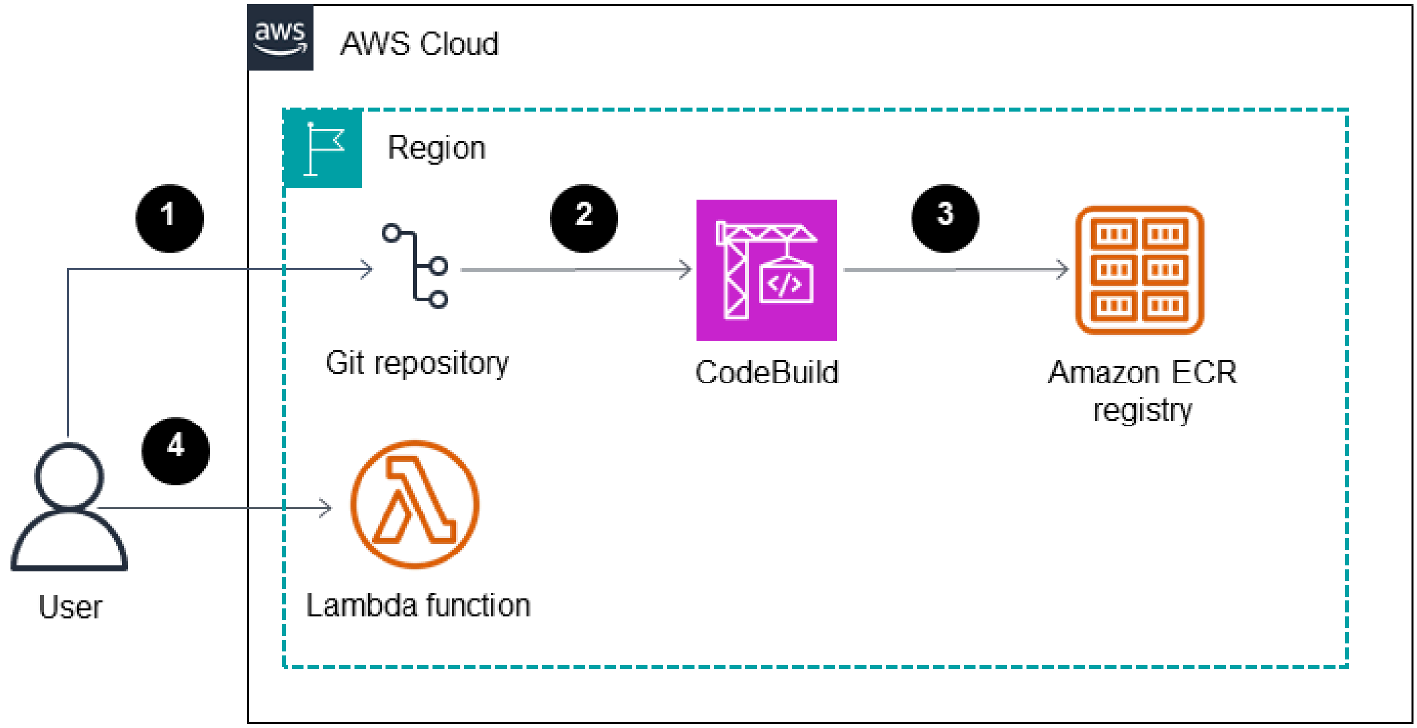
- Amazon Elastic Container Registry (Amazon ECR) activated
- Application code
- Docker images with the runtime interface client and the latest version of Python
- Working knowledge of Git

Limitations

- Maximum image size supported is 10 GB.
- Maximum runtime for a Lambda based container deployment is 15 minutes.

Architecture

Target architecture



1. You create a Git repository and commit the application code to the repository.
2. The AWS CodeBuild project is triggered by commit changes.
3. The CodeBuild project creates the Docker image and publishes the built image to Amazon ECR.
4. You create the Lambda function using the image in Amazon ECR.

Automation and scale

This pattern can be automated by using AWS CloudFormation, AWS Cloud Development Kit (AWS CDK), or API operations from an SDK. Lambda can automatically scale based on the number of requests, and you can tune it by using the concurrency parameters. For more information, see the Lambda documentation.

Tools

AWS services

- AWS CloudFormation AWS CloudFormationhelps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions. This pattern uses AWS CloudFormation Application Composer, which helps you visually view and edit AWS CloudFormation templates.

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- GitHub, GitLab, and Bitbucket are some of the commonly used Git-based source control system to keep track of source code changes.

Best practices

- Make your function as efficient and small as possible to avoid loading unnecessary files.
- Strive to have static layers higher up in your Docker file list, and place layers that change more often lower down. This improves caching, which improves performance.
- The image owner is responsible for updating and patching the image. Add that update cadence to your operational processes. For more information, see the AWS Lambda documentation.

Epics

Create a project in CodeBuild

Task	Description	Skills required
Create a Git repository.	Create a Git repository that will contain the application source code, the Dockerfile, and the `buildspec.yaml` file.	Developer

Task	Description	Skills required
Create a CodeBuild project.	To use a CodeBuild project to create the custom Lambda image, do the following: 1. Sign in to the AWS Management Console, and open the CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/. 2. Create a new project. For source, choose the Git repository that you created. For information about different kinds of Git repository integration, see the [Working with connections](#) documentation. 3. Confirm that privileged mode is enabled. To build Docker images, this is necessary. Otherwise, the image will not build successfully. 4. Provide values for the project name and description.	Developer

Task	Description	Skills required
Edit the Dockerfile.	The Dockerfile should be located in the top-level directory where you're developing the application. The Python code should be in the `src` folder. When you create the image, use the official Lambda supported images. Otherwise , a bootstrap error will occur, making the packing process more difficult. For details, see the Additional information section.	Developer
Create a repository in Amazon ECR.	Create a container repositor y in Amazon ECR. In the following example command, the name of the repository created is `cf-demo`: `aws ecr create-re pository --cf-demo` The repository will be referenced in the `buildspec .yaml` file.	AWS administrator, Developer

Task	Description	Skills required
Push the image to Amazon ECR.	You can use CodeBuild to perform the image-build process. CodeBuild needs permission to interact with Amazon ECR and to work with S3. As part of the process, the Docker image is built and pushed to the Amazon ECR registry. For details on the template and the code, see the Additional information section.	Developer
Verify that the image is in the repository.	To verify that the image is in the repository, on the Amazon ECR console, choose **Repositories**. The image should be listed, with tags and with the results of a vulnerability scan report if that feature was turned on in the Amazon ECR settings. For more information, see the AWS documentation.	Developer

Create the Lambda function to run the image

Task	Description	Skills required
Create the Lambda function.	On the Lambda console, choose **Create function**, and then choose **Container image**. Enter the function name and the URI for the	App developer

Task	Description	Skills required
	image that is in the Amazon ECR repository, and then choose **Create function**. For more information, see the [AWS Lambda documentation]().	
Test the Lambda function.	To invoke and test the function, choose **Test**. For more information, see the [AWS Lambda documentation]().	App developer

Troubleshooting

Issue	Solution
Build is not succeeding.	1. Check if the privileged mode is turned on for the CodeBuild project. 2. Ensure that the Docker related commands have the necessary permissions. Trying adding `sudo` to the commands. 3. Verify that the IAM role associated with CodeBuild has a policy with appropria te actions to interact with Amazon ECR, Amazon S3, and CloudWatch logs.

Related resources

- [Base images for Lambda]()
- [Docker sample for CodeBuild]()
- [Pass temporary credentials]()

Additional information

Edit the Dockerfile

The following code shows the commands that you edit in the Dockerfile:

```
FROM public.ecr.aws/lambda/python:3.xx

# Copy function code
COPY app.py ${LAMBDA_TASK_ROOT}
COPY requirements.txt  ${LAMBDA_TASK_ROOT}

# install dependencies
RUN pip3 install --user -r requirements.txt

# Set the CMD to your handler (could also be done as a parameter override outside of
 the Dockerfile)
CMD [ "app.lambda_handler" ]
```

In the `FROM` command, use appropriate value for the Python version that is supported by Lambda (for example, `3.12`). This will be the base image that is available in the public Amazon ECR image repository.

The `COPY app.py ${LAMBDA_TASK_ROOT}` command copies the code to the task root directory, which the Lambda function will use. This command uses the environment variable so we don't have to worry about the actual path. The function to be run is passed as an argument to the `CMD [ "app.lambda_handler" ]` command.

The `COPY requirements.txt` command captures the dependencies necessary for the code.

The `RUN pip install --user -r requirements.txt` command installs the dependencies to the local user directory.

To build your image, run the following command.

```
docker build -t <image name> .
```

Add the image in Amazon ECR

In the following code, replace `aws_account_id` with the account number, and replace `us-east-1` if you are using a different Region. The `buildspec` file uses the CodeBuild build number to uniquely identify image versions as a tag value. You can change this to fit your requirements.

The buildspec custom code

```
phases:
  install:
    runtime-versions:
       python: 3.xx
  pre_build:
    commands:
      - python3 --version
      - pip3 install --upgrade pip
      - pip3 install --upgrade awscli
      - sudo docker info
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - ls
      - cd app
      - docker build -t cf-demo:$CODEBUILD_BUILD_NUMBER .
      - docker container ls
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker image...
      - aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin aws_account_id.dkr.ecr.us-east-1.amazonaws.com
      - docker tag cf-demo:$CODEBUILD_BUILD_NUMBER aws_account_id.dkr.ecr.us-
east-1.amazonaws.com/cf-demo:$CODEBUILD_BUILD_NUMBER
      - docker push aws_account_id.dkr.ecr.us-east-1.amazonaws.com/cf-demo:
$CODEBUILD_BUILD_NUMBER
```

Deploy Java microservices on Amazon ECS using AWS Fargate

Vijay Thompson and Sandeep Bondugula, Amazon Web Services

Summary

This pattern provides guidance for deploying containerized Java microservices on Amazon Elastic Container Service (Amazon ECS) by using AWS Fargate. The pattern doesn't use Amazon Elastic Container Registry (Amazon ECR) for container management; instead, Docker images are pulled in from a Docker hub.

Prerequisites and limitations

Prerequisites

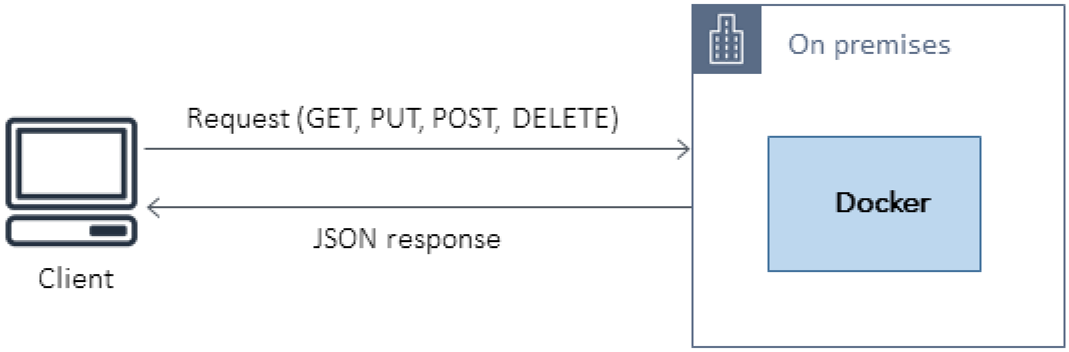
- An existing Java microservices application on a Docker hub
- A public Docker repository
- An active AWS account
- Familiarity with AWS services, including Amazon ECS and Fargate
- Docker, Java, and Spring Boot framework
- Amazon Relational Database Service (Amazon RDS) up and running (optional)
- A virtual private cloud (VPC) if the application requires Amazon RDS (optional)

Architecture

Source technology stack

- Java microservices (for example, implemented in Spring Boot) and deployed on Docker

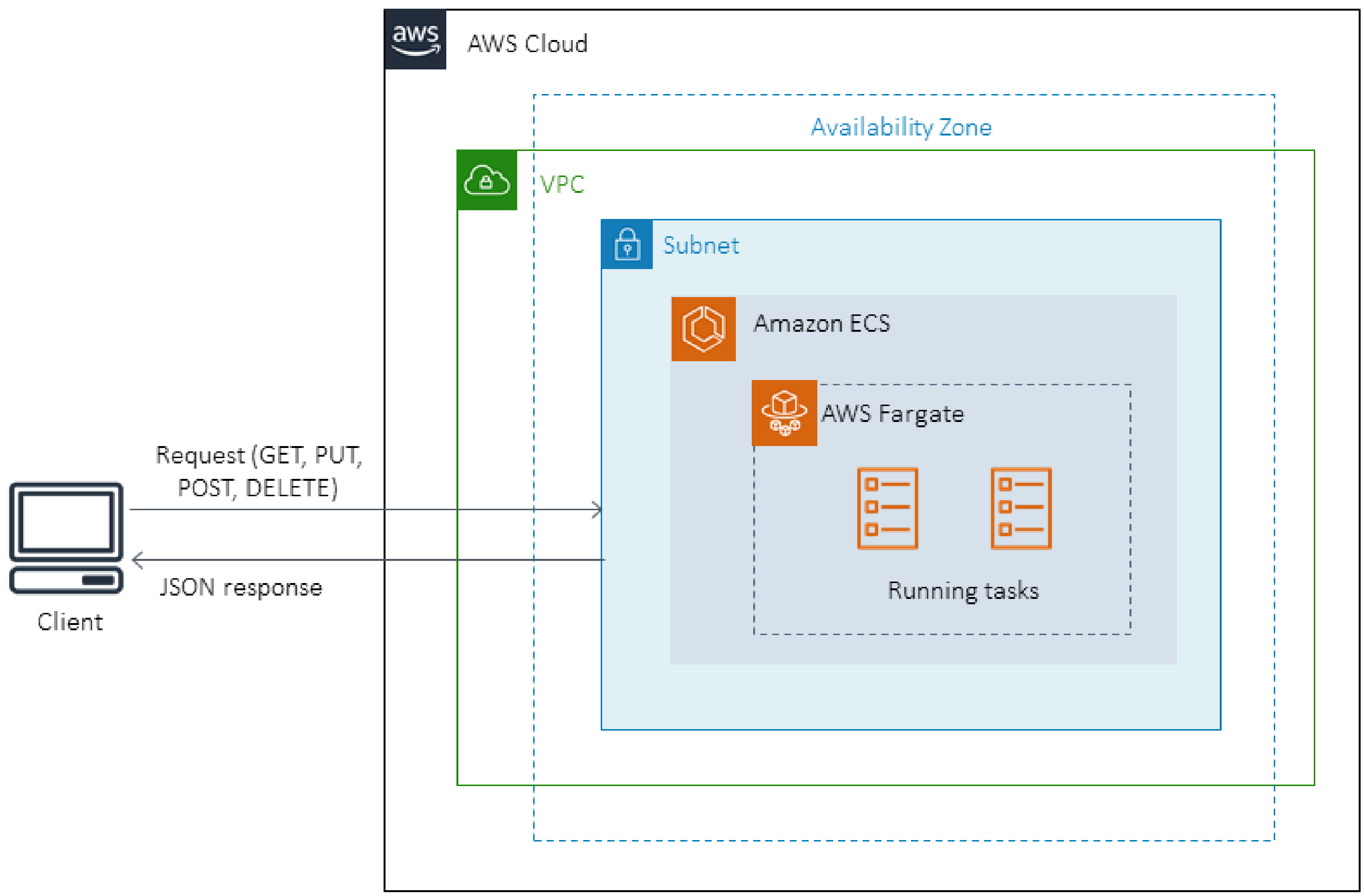
Source architecture



Target technology stack

- An Amazon ECS cluster that hosts each microservice by using Fargate
- A VPC network to host the Amazon ECS cluster and associated security groups
- A cluster/task definition for each microservice that spins up containers by using Fargate

Target architecture



Tools

Tools

- [Amazon ECS](#) eliminates the need to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.
- [AWS Fargate](#) helps you run containers without needing to manage servers or Amazon Elastic Compute Cloud (Amazon EC2) instances. It’s used in conjunction with Amazon Elastic Container Service (Amazon ECS).
- [Docker](#) is a software platform that allows you to build, test, and deploy applications quickly. Docker packages software into standardized units called *containers* that have everything the software needs to run, including libraries, system tools, code, and runtime.

Docker code

The following Dockerfile specifies the Java Development Kit (JDK) version that is used, where the Java archive (JAR) file exists, the port number that is exposed, and the entry point for the application.

```
FROM openjdk:11
ADD target/Spring-docker.jar Spring-docker.jar
EXPOSE 8080
ENTRYPOINT ["java","-jar","Spring-docker.jar"]
```

Epics

Create new task definitions

Task	Description	Skills required
Create a task definition.	Running a Docker container in Amazon ECS requires a task definition. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/, choose **Task definitions**, and then create a new	AWS systems administrator, App developer

Task	Description	Skills required
	task definition. For more information, see the [Amazon ECS documentation]().	
Choose launch type.	Choose **Fargate** as the launch type.	AWS systems administrator, App developer
Configure the task.	Define a task name and configure the application with the appropriate amount of task memory and CPU.	AWS systems administrator, App developer
Define the container.	Specify the container name. For the image, enter the Docker site name, the repository name, and the tag name of the Docker image (`docker.io/sample-repo/sample-application:sample-tag-name`). Set memory limits for the application, and set port mappings (`8080, 80`) for the allowed ports.	AWS systems administrator, App developer
Create the task.	When the task and container configurations are in place, create the task. For detailed instructions, see the links in the *Related resources* section.	AWS systems administrator, App developer

Configure the cluster

Task	Description	Skills required
Create and configure a cluster.	Choose **Networking only** as the cluster type, configure the name, and then create the cluster or use an existing cluster if available. For more information, see the Amazon ECS documentation.	AWS systems administrator, App developer

Configure Task

Task	Description	Skills required
Create a task.	Inside the cluster, choose **Run new task**.	AWS systems administrator, App developer
Choose launch type.	Choose **Fargate** as the launch type.	AWS systems administrator, App developer
Choose task definition, revision, and platform version.	Choose the task that you want to run, the revision of the task definition, and the platform version.	AWS systems administrator, App developer
Select the cluster.	Choose the cluster where you want to run the task from.	AWS systems administrator, App developer
Specify the number of tasks.	Configure the number of tasks that should run. If you're launching with two or more tasks, a load balancer is required to distribute the traffic among the tasks.	AWS systems administrator, App developer

Task	Description	Skills required
Specify the task group.	(Optional) Specify a task group name to identify a set of related tasks as a task group.	AWS systems administrator, App developer
Configure the cluster VPC, subnets, and security groups.	Configure the cluster VPC and the subnets on which you want to deploy the applicati on. Create or update security groups (HTTP, HTTPS, and port 8080) to provide access to inbound and outbound connections.	AWS systems administrator, App developer
Configure public IP settings.	Enable or disable the public IP, depending on whether you want to use a public IP address for Fargate tasks. The default, recommended option is **Enabled**.	AWS systems administrator, App developer
Review settings and create the task	Review your settings, and then choose **Run Task**.	AWS systems administrator, App developer

Cut over

Task	Description	Skills required
Copy the application URL.	When the task status has been updated to *Running*, select the task. In the Networking section, copy the public IP.	AWS systems administrator, App developer

Task	Description	Skills required
Test your application.	In your browser, enter the public IP to test the applicati on.	AWS systems administrator, App developer

Related resources

- Docker Basics for Amazon ECS (Amazon ECS documentation)
- Amazon ECS on AWS Fargate (Amazon ECS documentation)
- Creating a Task Definition (Amazon ECS documentation)
- Creating a Cluster (Amazon ECS documentation)
- Configuring Basic Service Parameters (Amazon ECS documentation)
- Configuring a Network (Amazon ECS documentation)
- Deploying Java Microservices on Amazon ECS (blog post)

Deploy Kubernetes resources and packages using Amazon EKS and a Helm chart repository in Amazon S3

Sagar Panigrahi, Amazon Web Services

Summary

This pattern helps you to manage Kubernetes applications efficiently, regardless of their complexity. The pattern integrates Helm into your existing continuous integration and continuous delivery (CI/CD) pipelines to deploy applications into a Kubernetes cluster. Helm is a Kubernetes package manager that helps you manage Kubernetes applications. Helm charts help to define, install, and upgrade complex Kubernetes applications. Charts can be versioned and stored in Helm repositories, which improves mean time to restore (MTTR) during outages.

This pattern uses Amazon Elastic Kubernetes Service (Amazon EKS) for the Kubernetes cluster. It uses Amazon Simple Storage Service (Amazon S3) as a Helm chart repository, so that the charts can be centrally managed and accessed by developers across the organization.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account with a virtual private cloud (VPC)
- An Amazon EKS cluster
- Worker nodes set up within the Amazon EKS cluster and ready to take workloads
- Kubectl for configuring the Amazon EKS kubeconfig file for the target cluster in the client machine
- AWS Identity and Access Management (IAM) access to create the S3 bucket
- IAM (programmatic or role) access to Amazon S3 from the client machine
- Source code management and a CI/CD pipeline

Limitations

- There is no support at this time for upgrading, deleting, or managing custom resource definitions (CRDs).
- If you are using a resource that refers to a CRD, the CRD must be installed separately (outside of the chart).

Product versions

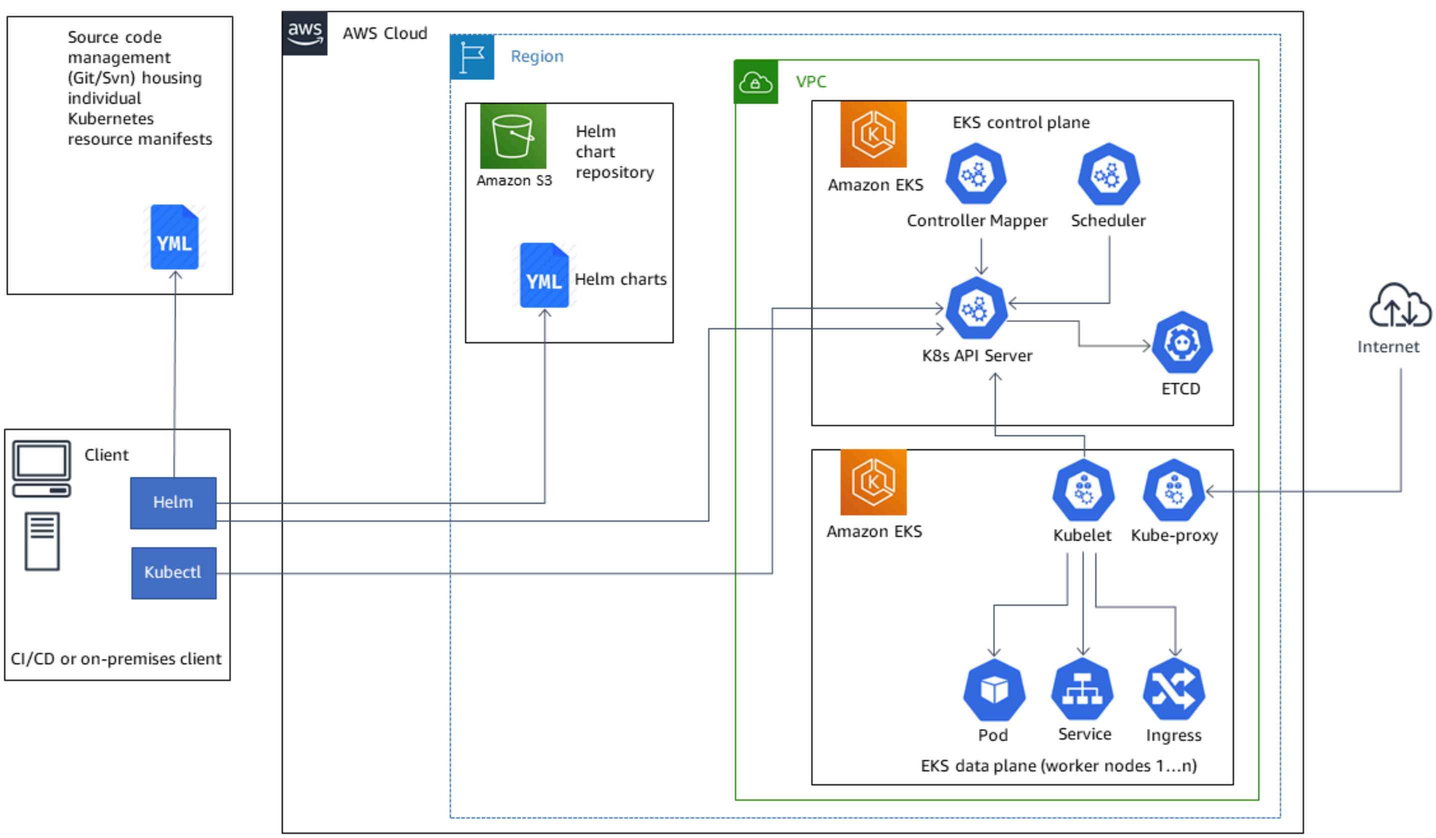
- Helm v3.6.3

Architecture

Target technology stack

- Amazon EKS
- Amazon VPC
- Amazon S3
- Source code management
- Helm
- Kubectl

Target architecture



Automation and scale

- AWS CloudFormation can be used to automate the infrastructure creation. For more information, see Creating Amazon EKS resources with AWS CloudFormation in the Amazon EKS documentation.
- Helm is to be incorporated into your existing CI/CD automation tool to automate the packaging and versioning of Helm charts (out of scope for this pattern).
- GitVersion or Jenkins build numbers can be used to automate the versioning of charts.

Tools

Tools

- Amazon EKS – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service for running Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.
- Helm – Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- Kubectl – Kubectl is a command line utility for running commands against Kubernetes clusters.

Code

The example code is attached.

Epics

Configure and initialize Helm

Task	Description	Skills required
Install the Helm client.	To download and install the Helm client on your local system, use the following command.	DevOps engineer

Task	Description	Skills required
	`sudo curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash`	
Validate the Helm installation.	To validate that Helm is able to communicate with the Kubernetes API server within the Amazon EKS cluster, run `helm version`.	DevOps engineer

Create and install a Helm chart in the Amazon EKS cluster

Task	Description	Skills required
Create a Helm chart for NGINX.	To create a helm chart named `my-nginx` on the client machine, run `helm create my-nginx`.	DevOps engineer
Review the structure of the chart.	To review the structure of the chart, run the tree command `tree my-nginx/` .	DevOps engineer
Deactivate service account creation in the chart.	In `values.yaml` , under the `serviceAccount` section, set the `create` key to `false`. This is turned off because there is no requirement to create a service account for this pattern.	DevOps engineer
Validate (lint) the modified chart for syntactical errors.	To validate the chart for any syntactical error before	DevOps engineer

Task	Description	Skills required
	installing it in the target cluster, run `helm lint my-nginx/` .	
Install the chart to deploy Kubernetes resources.	To run the Helm chart installation, use the following command. `helm install --name my-nginx-release --debug my-nginx/ --namespace helm-space` The optional `debug` flag outputs all debug messages during the installation. The `namespace` flag specifies the namespace in which the resources part of this chart will be created.	DevOps engineer
Review the resources in the Amazon EKS cluster.	To review the resources that were created as part of the Helm chart in the `helm-space` namespace, use the following command. `kubectl get all -n helm-space`	DevOps engineer

Roll back to a previous version of a Kubernetes application

Task	Description	Skills required
Modify and upgrade the release.	To modify the chart, in `values.yaml` , change the `replicaCount` value to 2. Then upgrade the already installed release by running the following command. `helm upgrade my-nginx-release my-nginx/ --namespace helm-space`	DevOps engineer
Review the history of the Helm release.	To list all the revisions for a specific release that have been installed using Helm, run the following command. `helm history my-nginx-release`	DevOps engineer
Review the details for a specific revision.	Before switching or rolling back to a working version, and for an additional layer of validation before installin g a revision, view which values were passed to each of the revisions by using the following command. `helm get --revision=2 my-nginx-release`	DevOps engineer

Task	Description	Skills required
Roll back to a previous version.	To roll back to a previous revision, use the following command. `helm rollback my-nginx-release 1` This example is rolling back to revision number 1.	DevOps engineer

Initialize an S3 bucket as a Helm repository

Task	Description	Skills required
Create an S3 bucket for Helm charts.	Create a unique S3 bucket. In the bucket, create a folder called `charts`. The example in this pattern uses `s3://my-helm-charts/charts` as the target chart repository.	Cloud administrator
Install the Helm plugin for Amazon S3.	To install the helm-s3 plugin on your client machine, use the following command. `helm plugin install https://github.com/hypnoglow/helm-s3.git --version 0.10.0` Note: Helm V3 support is available with plugin version 0.9.0 and above.	DevOps engineer

Task	Description	Skills required
Initialize the Amazon S3 Helm repository.	To initialize the target folder as a Helm repository, use the following command. `helm S3 init s3://my-helm-charts/charts` The command creates an `index.yaml` file in the target to track all the chart information that is stored at that location.	DevOps engineer
Add the Amazon S3 repository to Helm.	To add the repository in the client machine, use the following command. `helm repo add my-helm-charts s3://my-helm-charts/charts` This command adds an alias to the target repository in the Helm client machine.	DevOps engineer
Review the repository list.	To view the list of repositories in the Helm client machine, run `helm repo list`.	DevOps engineer

Package and store charts in the Amazon S3 Helm repository

Task	Description	Skills required
Package the chart.	To package the `my-nginx` chart that you created, run	DevOps engineer

Task	Description	Skills required
	`helm package ./my-nginx/` . The command packages all the contents of the `my-nginx` chart folder into an archive file, which is named using the version number that is mentioned in the `Chart.yaml` file.	
Store the package in the Amazon S3 Helm repository.	To upload the package to the Helm repository in Amazon S3, run the following command, using the correct name of the `.tgz` file. `helm s3 push ./my-nginx-0.1.0.tgz my-helm-charts`	DevOps engineer
Search for the Helm chart.	To confirm that the chart appears both locally and in the Helm repository in Amazon S3, run the following command. `helm search repo my-nginx`	DevOps engineer

Modify, version, and package a chart

Task	Description	Skills required
Modify and package the chart.	In `values.yaml` , set the `replicaCount` value to 1. Then package the chart by	DevOps engineer

Task	Description	Skills required
	running `helm package ./my-nginx/`, this time changing the version in `Chart.yaml` to `0.1.1`. The versioning is ideally updated through automation using tools such as GitVersion or Jenkins build numbers in a CI/CD pipeline. Automating the version number is out of scope for this pattern.	
Push the new version to the Helm repository in Amazon S3.	To push the new package with version of 0.1.1 to the `my-helm-charts` Helm repository in Amazon S3, run the following command. `helm s3 push ./my-nginx-0.1.1.tgz my-helm-charts`	DevOps engineer

Search for and install a chart from the Amazon S3 Helm repository

Task	Description	Skills required
Search for all versions of the my-nginx chart.	To view all the available versions of a chart, run the following command with the `--versions` flag. `helm search repo my-nginx --versions`	DevOps engineer

Task	Description	Skills required
	Without the flag, Helm by default displays the latest uploaded version of a chart.	
Install a chart from the Amazon S3 Helm repository.	The search results from the previous task show the multiple versions of the `my-nginx` chart. To install the new version (0.1.1) from the Amazon S3 Helm repository, use the following command. `helm upgrade my-nginx-release my-helm-charts/my-nginx --version 0.1.1 --namespace helm-space`	DevOps engineer

Related resources

- HELM documentation
- helm-s3 plugin (MIT License)
- HELM client binary
- Amazon EKS documentation

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Deploy a sample Java microservice on Amazon EKS and expose the microservice using an Application Load Balancer

Vijay Thompson and Akkamahadevi Hiremath, Amazon Web Services

Summary

This pattern describes how to deploy a sample Java microservice as a containerized application on Amazon Elastic Kubernetes Service (Amazon EKS) by using the `eksctl` command line utility and Amazon Elastic Container Registry (Amazon ECR). You can use an Application Load Balancer to load balance the application traffic.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows
- A running Docker daemon
- The `eksctl` command line utility, installed and configured on macOS, Linux, or Windows (For more information, see Getting started with Amazon EKS – eksctl in the Amazon EKS documentation.)
- The `kubectl` command line utility, installed and configured on macOS, Linux, or Windows (For more information, see Installing or updating kubectl in the Amazon EKS documentation.)

Limitations

- This pattern doesn't cover the installation of an SSL certificate for the Application Load Balancer.

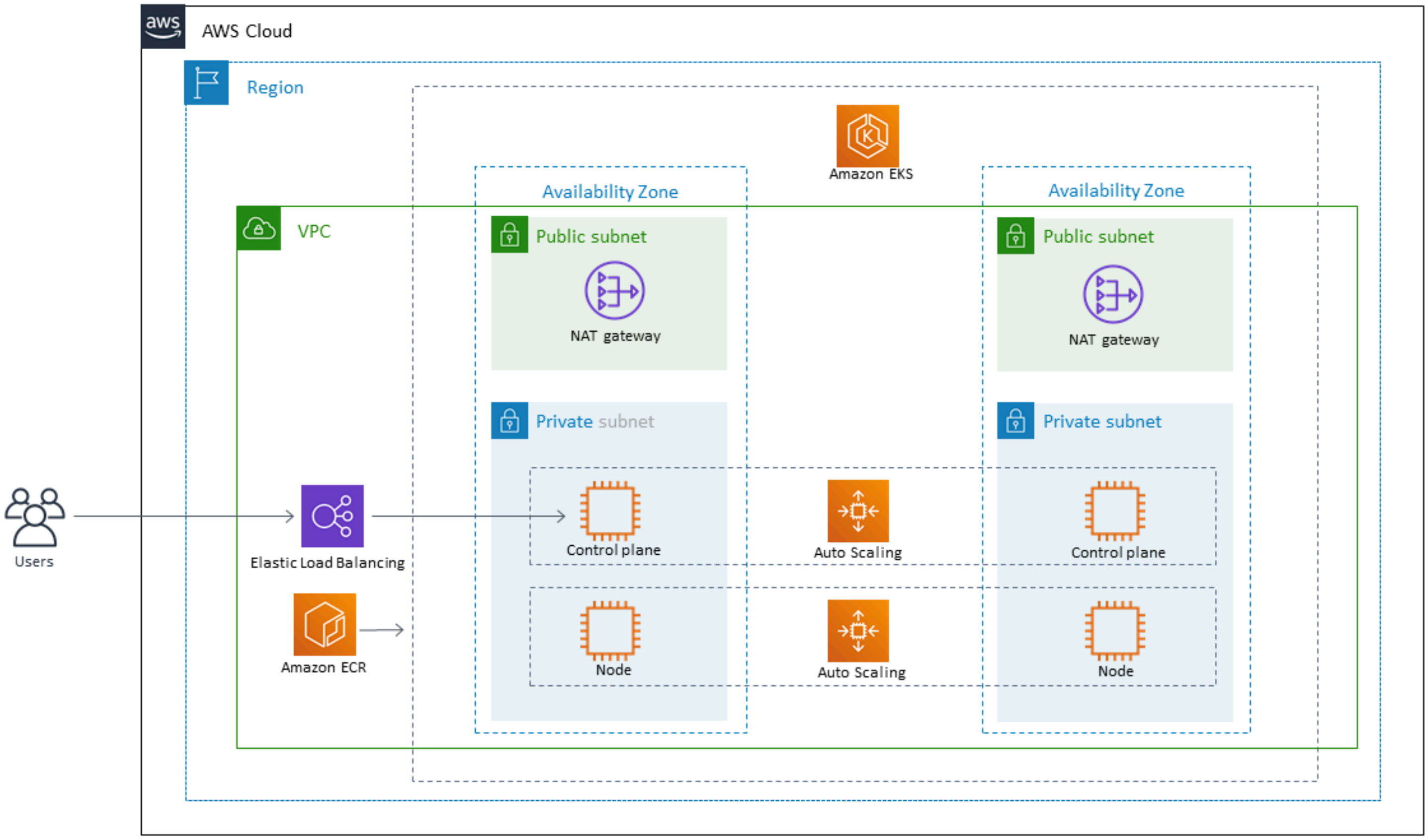
Architecture

Target technology stack

- Amazon ECR
- Amazon EKS
- Elastic Load Balancing

Target architecture

The following diagram shows an architecture for containerizing a Java microservice on Amazon EKS.



Tools

- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses, in one or more Availability Zones.
- eksctl helps you create clusters on Amazon EKS.
- kubectl makes it possible to run commands against Kubernetes clusters.

- Docker helps you build, test, and deliver applications in packages called containers.

Epics

Create an Amazon EKS cluster by using eksctl

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create an Amazon EKS cluster.</td><td>To create an Amazon EKS cluster that uses two t2.small Amazon EC2 instances as nodes, run the following command:
<pre>eksctl create cluster --name <your-cluster-name> --version <version-number> --nodes=1 --node-type=t2.small</pre>
Note
The process can take between 15 to 20 minutes. After the cluster is created, the appropriate Kubernetes configuration is added to your kubeconfig file. You can use the <code>kubeconfig</code> file with <code>kubectl</code> to deploy the application in later steps.</td><td>Developer, System Admin</td></tr>
</table>

Task	Description	Skills required
Verify the Amazon EKS cluster.	To verify that the cluster is created and that you can connect to it, run the `kubectl get nodes` command.	Developer, System Admin

Create an Amazon ECR repository and push the Docker image.

Task	Description	Skills required
Create an Amazon ECR repository.	Follow the instructions from [Creating a private repository]() in the Amazon ECR documentation.	Developer, System Admin
Create a POM XML file.	Create a `pom.xml` file based on the *Example POM file* code in the [Additional information]() section of this pattern.	Developer, System Admin

Create a source file.	Create a source file called `HelloWorld.java` in the `src/main/java/eksExample` path based on the following example:
	`package eksExample;`
	`import static spark.Spark.get;`
	
	`public class HelloWorld {`
	`    public static void main(String[] args) {`
	`        get("/", (req, res) -> {`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>
<pre>
            return
 "Hello World!";
        });
    }
}
</pre>
Be sure to use the following directory structure:
<pre>
Dockerfile
deployment.yaml
ingress.yaml
pom.xml
service.yaml
src
    ### main
        ### java
            ###
 eksExample
                ###
 HelloWorld.java
</pre>
</td><td></td></tr>
<tr><td>Create a Dockerfile.</td><td>Create a <code>Dockerfile</code> based on the Example Dockerfile code in the Additional information section of this pattern.</td><td>Developer, System Admin</td></tr>
</table>

Task	Description	Skills required

Build and push the Docker image.	In the directory where you want your `Dockerfile` to build, tag, and push the image to Amazon ECR, run the following commands:
	`aws ecr get-login -password --region <region>| docker login --username <username > --password-stdin <account_number>.dkr.ecr.<region>.amazonaws.com`
	`docker buildx build --platform linux/amd64 -t hello-world-java:v1 .`
	`docker tag hello-world-java:v1 <account_number>.dkr.ecr.<region>.amazonaws.com/<repository_name>:v1`
	`docker push <account_number>.dkr.ecr.<region>.amazonaws.com/<repository_name>:v1`
	
	Note
	Modify the AWS Region, account number, and repository details in the preceding commands. Be sure to note the	

Task	Description	Skills required
	image URL for later use. **Important** A macOS system with an M1 chip has a problem building an image that's compatible with Amazon EKS running on an AMD64 platform. To resolve this issue, use [docker buildx](https://docs.docker.com/engine/reference/commandline/buildx/) to build a Docker image that works on Amazon EKS.	

Deploy the Java microservices

Task	Description	Skills required
Create a deployment file.	Create a YAML file called `deployment.yaml` based on the *Example deployment file* code in the [Additional information](#additional-information) section of this pattern.	Developer, System Admin

Task	Description	Skills required
	Note Use the image URL that you copied earlier as the path of the image file for the Amazon ECR repository.	
Deploy the Java microservices on the Amazon EKS cluster.	To create a deployment in your Amazon EKS cluster, run the `kubectl apply -f deployment.yaml` command.	Developer, System Admin
Verify the status of the pods.	1. To verify the status of the pods, run the `kubectl get pods` command. 2. Wait for the status to change to **Ready**.	Developer, System Admin
Create a service.	1. Create a file called `service.yaml` based on the *Example service file* code in the [Additional information](#) section of this pattern. 2. Run the `kubectl apply -f service.yaml` command.	Developer, System Admin

Task	Description	Skills required
Install the AWS Load Balancer Controller add-on.	Follow the instructions from [Installing the AWS Load Balancer Controller add-on]() in the Amazon EKS documenta tion. **Note** You must have the add-on installed to create an Applicati on Load Balancer or Network Load Balancer for a Kubernetes service.	Devloper, System Admin
Create an ingress resource.	Create a YAML file called `ingress.yaml` based on the *Example ingress resource file* code in the [Additional information]() section of this pattern.	Developer, System Admin
Create an Application Load Balancer.	To deploy the ingress resource and create an Application Load Balancer, run the `kubectl apply -f ingress.yaml` command.	Developer, System Admin

Test the application

Task	Description	Skills required
Test and verify the application.	1. To get the load balancer's DNS name from the ADDRESS field, run the `kubectl get ingress.networking.k8s.io/java-microservice-ingress` command. 2. On an EC2 instance in the same VPC as your Amazon EKS nodes, run the `curl -v <DNS address from previous command>` command.	Developer, System Admin

Related resources

- Creating a private repository (Amazon ECR documentation)
- Pushing a Docker image (Amazon ECR documentation)
- Ingress Controllers (Amazon EKS Workshop)
- Docker buildx (Docker docs)

Additional information

Example POM file

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

```
  <groupId>helloWorld</groupId>
  <artifactId>helloWorld</artifactId>
  <version>1.0-SNAPSHOT</version>


  <dependencies>
    <dependency>
      <groupId>com.sparkjava</groupId><artifactId>spark-core</
artifactId><version>2.0.0</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId><artifactId>maven-jar-plugin</
artifactId><version>2.4</version>
        <configuration><finalName>eksExample</finalName><archive><manifest>
              <addClasspath>true</addClasspath><mainClass>eksExample.HelloWorld</
mainClass><classpathPrefix>dependency-jars/</classpathPrefix>
            </manifest></archive>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId><artifactId>maven-compiler-plugin</
artifactId><version>3.1</version>
        <configuration><source>1.8</source><target>1.8</target></configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId><artifactId>maven-assembly-plugin</
artifactId>
        <executions>
          <execution>
            <goals><goal>attached</goal></goals><phase>package</phase>
            <configuration>
              <finalName>eksExample</finalName>
              <descriptorRefs><descriptorRef>jar-with-dependencies</descriptorRef></
descriptorRefs>
              <archive><manifest><mainClass>eksExample.HelloWorld</mainClass></
manifest></archive>
            </configuration>
          </execution>
        </executions>
```

```
      </plugin>
    </plugins>
  </build>
</project>
```

Example Dockerfile

```
FROM bellsoft/liberica-openjdk-alpine-musl:17

RUN apk add maven
WORKDIR /code

# Prepare by downloading dependencies
ADD pom.xml /code/pom.xml
RUN ["mvn", "dependency:resolve"]
RUN ["mvn", "verify"]

# Adding source, compile and package into a fat jar
ADD src /code/src
RUN ["mvn", "package"]

EXPOSE 4567
CMD ["java", "-jar", "target/eksExample-jar-with-dependencies.jar"]
```

Example deployment file

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: microservice-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app.kubernetes.io/name: java-microservice
  template:
    metadata:
      labels:
        app.kubernetes.io/name: java-microservice
    spec:
      containers:
      - name: java-microservice-container
        image: .dkr.ecr.amazonaws.com/:
```

```
        ports:
        - containerPort: 4567
```

Example service file

```
apiVersion: v1
kind: Service
metadata:
  name: "service-java-microservice"
spec:
  ports:
    - port: 80
      targetPort: 4567
      protocol: TCP
  type: NodePort
  selector:
    app.kubernetes.io/name: java-microservice
```

Example ingress resource file

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: "java-microservice-ingress"
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/load-balancer-name: apg2
    alb.ingress.kubernetes.io/target-type: ip
  labels:
    app: java-microservice
spec:
  rules:
    - http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: "service-java-microservice"
                port:
                  number: 80
```

Deploy a clustered application to Amazon ECS by using AWS Copilot

Jean-Baptiste Guillois, Mathew George, and Thomas Scott, Amazon Web Services

Summary

This pattern shows how to deploy containers in an Amazon Elastic Container Service (Amazon ECS) cluster in two ways—by using the Amazon Web Services (AWS) Management Console, and by using AWS Copilot—to demonstrate how AWS Copilot simplifies deployment tasks.

Amazon ECS is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage.

The AWS Copilot command line interface (CLI) commands simplify building, releasing, and operating production-ready containerized applications on Amazon ECS from a local development environment. The AWS Copilot CLI aligns with developer workflows that support modern application best practices: from using infrastructure as code to creating a continuous integration and continuous delivery (CI/CD) pipeline provisioned on behalf of a user. You can use the AWS Copilot CLI as part of your everyday development and testing cycle as an alternative to the AWS Management Console.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) locally installed and configured to use your AWS account (see the installation instructions and the configuration instructions in the AWS CLI documentation)
- AWS Copilot locally installed (see the installation instructions in the Amazon ECS documentation)
- Docker installed on your local machine (see the Docker documentation)

Limitations

- Docker enforces pull limits of 100 container images per 6 hours per IP address on the free plan.

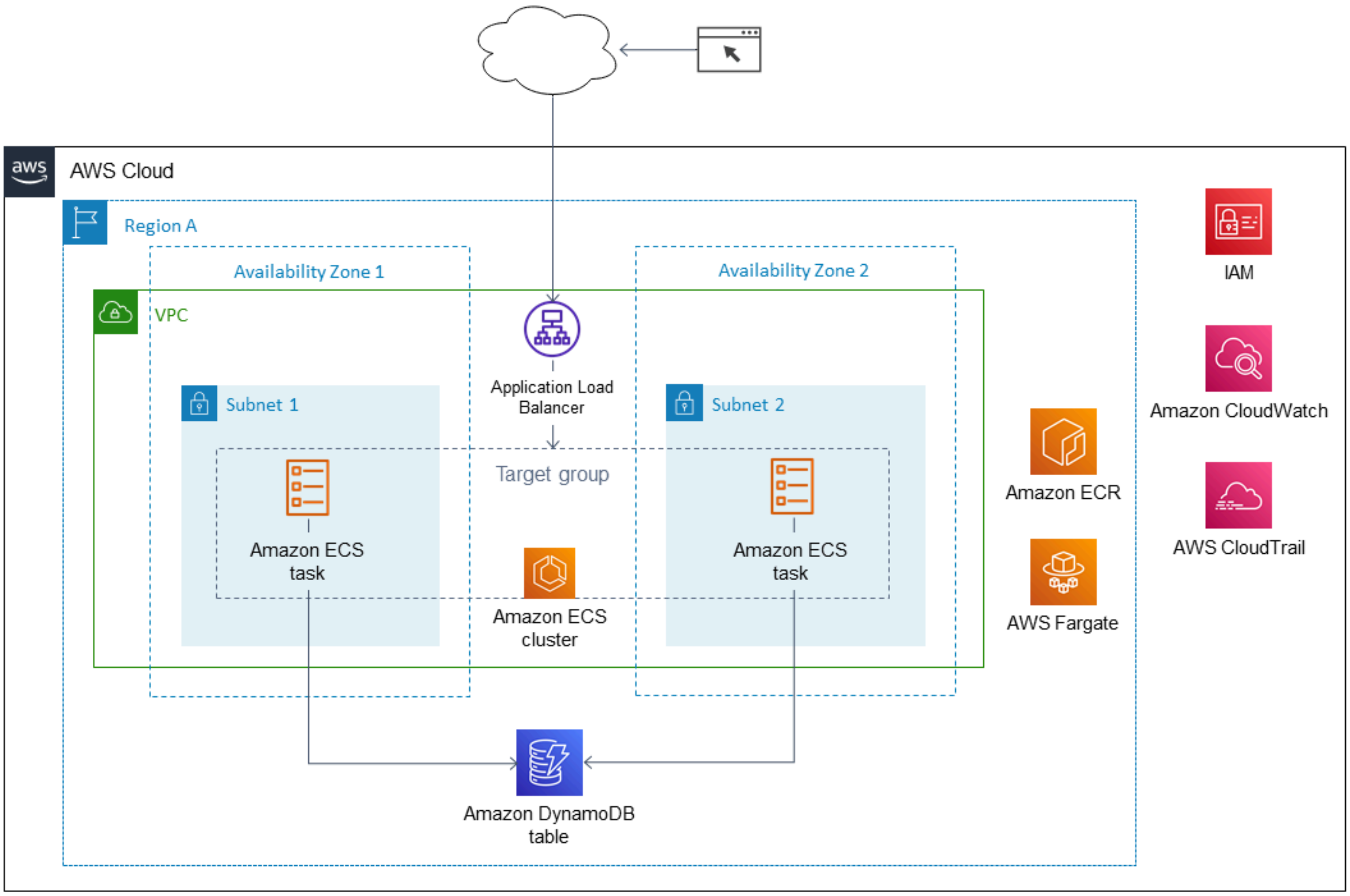
Architecture

Target technology stack

- AWS environment set up with a virtual private cloud (VPC), public and private subnets, and security groups
- Amazon ECS cluster
- Amazon ECS service and task definition
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon DynamoDB
- Application Load Balancer
- AWS Fargate
- Amazon Identity and Access Management (IAM)
- Amazon CloudWatch
- AWS CloudTrail

Target architecture

When you deploy the sample application for this pattern, multiple tasks are created and deployed in separate Availability Zones. Each task stores data in Amazon DynamoDB. When you access the webpage for a task, you can view the data from all other tasks.



Tools

AWS services

- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable. Amazon ECR supports private repositories with resource-based permissions using IAM.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage.
- AWS Copilot – AWS Copilot provides a command line interface that helps you launch and manage containerized applications on AWS, including pushing to a registry, creating a task definition, and creating a cluster.

- AWS Fargate – AWS Fargate is a serverless, pay-as-you-go compute engine that lets you focus on building applications without managing servers. AWS Fargate is compatible with both Amazon ECS and Amazon Elastic Kubernetes Service (Amazon EKS). When you run your Amazon ECS tasks and services with the Fargate launch type or a Fargate capacity provider, you package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application. Each Fargate task has its own isolation boundary and doesn't share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.
- Amazon DynamoDB – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- Elastic Load Balancing (ELB) – Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, such as EC2 instances, containers, and IP addresses, in one or more Availability Zones. It monitors the health of its registered targets, and routes traffic only to the healthy targets. Elastic Load Balancing scales your load balancer as your incoming traffic changes over time. It can automatically scale to the vast majority of workloads.

Tools

- Docker Command Line Interface
- AWS Command Line Interface (AWS CLI)
- AWS Copilot command line interface

Code

The code for the sample application used in this pattern is available on GitHub, in the Cluster Sample Application repository. Follow the instructions in the next section to use the sample files.

Epics

Deploy the application stack - option 1 (AWS Management Console)

Task	Description	Skills required
Clone the GitHub repository.	Clone the sample code repository by using the command:	App developer, AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>git clone https://github.com/aws-samples/cluster-sample-app cluster-sample-app
 && cd cluster-sample-app</code></pre></td><td></td></tr>
<tr><td>Create your Amazon ECR repository.</td><td>1. Sign in to the AWS Management Console and open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. Choose Create repository.
3. For the repository name, enter cluster-sample-app.
4. For all other settings, keep the default values.
5. Choose Create repository.

For more information, see Creating a private repository in the Amazon ECR documentation.</td><td>App developer, AWS DevOps</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Build, tag, and push your Docker image to your Amazon ECR repository.</td><td>

1. Select the repository you just created and choose **View push commands**.
2. Copy the commands that are displayed and run them locally to build, tag, and push your docker image. These commands will be similar to the following.

To authenticate your Docker client to the registry:

```
aws ecr get-login
-password --region
 <YOUR_AWS_REGION> |
 docker login --usernam
e AWS --password-stdin
 <YOUR_AWS_ACCOUNT>
.dkr.ecr.<YOUR_AWS
_REGION>.amazonaws
.com
```

To build your Docker image:

```
docker build -t cluster-
sample-app .
```

To tag your Docker image:

```
docker tag cluster-
sample-app:latest
 <YOUR_AWS_ACCOUNT>
.dkr.ecr.<YOUR_AWS
_REGION>.amazonaws
```

</td><td>App developer, AWS DevOps</td></tr>
</table>

Task	Description	Skills required

	`.com/cluster-sample-app:latest`
	
	To push the Docker image to your repository:
	
	`docker push <YOUR_AWS_ACCOUNT>.dkr.ecr.<YOUR_AWS_REGION>.amazonaws.com/cluster-sample-app:latest`	

Task	Description	Skills required
Deploy the application stack.	1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/. 2. Choose **Create stack**. 3. In the **Prepare template** section, choose **Template is ready**. 4. In the **Specify template** section, choose **Upload a template file**. 5. Choose the local file `cluster-sample-app-stack.yml` that you cloned from the GitHub repository as the CloudFormation template, and then choose **Next**. 6. Enter a name for your stack, and then choose **Next**. 7. Keep all default options, and then choose **Next**. 8. Review all options, acknowledge the creation of IAM resources, and then choose **Create stack**. 9. When your application stack has been deployed, choose the **Output** tab, copy the URL, and open it in your browser to access the application.	AWS DevOps, App developer

Task	Description	Skills required
	For more information about deploying CloudFormation templates, see Creating a stack in the AWS CloudFormation documentation.	

Deploy the application stack – option 2 (AWS Copilot CLI)

Task	Description	Skills required
Clone the GitHub repository.	Clone the sample code repository by using the command: `git clone https://github.com/aws-samples/cluster-sample-app cluster-sample-app && cd cluster-sample-app`	App developer, AWS DevOps
Deploy your container image to AWS by using the AWS Copilot CLI.	Deploy the application in one step by using the following command in the root directory of your project: `copilot init --app cluster-sample-app --name demo --type "Load Balanced Web Service" --dockerfile ./Dockerfile --port 8080 --deploy` You should then be able to access the application by	App developer, AWS DevOps

Task	Description	Skills required
	using the DNS name provided as output.	

Delete the created resources

Task	Description	Skills required
Delete the resources created through the AWS Managemen t Console.	If you used option 1 (the AWS Management Console) to deploy the application stack, follow these steps when you're ready to delete the resources you created: 1. Open the CloudForm ation console at https://console.aws.amazon.com/cloudformation/. 2. Select the stack you created, and then choose **Delete**. 3. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories. 4. Select the repository you created, and then choose **Delete**.	App developer, AWS DevOps
Delete the resources created by AWS Copilot.	If you used option 2 (the AWS Copilot CLI) to deploy the application stack, run the following command from the root directory of your project	App developer, AWS DevOps

Task	Description	Skills required
	when you're ready to delete the resources you created: `copilot app delete`	

Related resources

- Installing or updating the latest version of the AWS CLI (AWS CLI documentation)
- Using the AWS Copilot command line interface (Amazon ECS documentation)
- Amazon ECS on AWS Fargate (Amazon ECR documentation)
- Amazon ECS documentation
- Amazon ECR documentation
- Amazon CloudFormation documentation
- Docker Desktop (Docker documentation)

Deploy a gRPC-based application on an Amazon EKS cluster and access it with an Application Load Balancer

Kirankumar Chandrashekar and Huy Nguyen, Amazon Web Services

Summary

This pattern describes how to host a gRPC-based application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster and securely access it through an Application Load Balancer.

gRPC is an open-source remote procedure call (RPC) framework that can run in any environment. You can use it for microservice integrations and client-server communications. For more information about gRPC, see the AWS blog post Application Load Balancer support for end-to-end HTTP/2 and gRPC.

This pattern shows you how to host a gRPC-based application that runs on Kubernetes pods on Amazon EKS. The gRPC client connects to an Application Load Balancer through the HTTP/2 protocol with an SSL/TLS encrypted connection. The Application Load Balancer forwards traffic to the gRPC application that runs on Amazon EKS pods. The number of gRPC pods can be automatically scaled based on traffic by using the Kubernetes Horizontal Pod Autoscaler. The Application Load Balancer's target group performs health checks on the Amazon EKS nodes, evaluates if the target is healthy, and forwards traffic only to healthy nodes.

Prerequisites and limitations

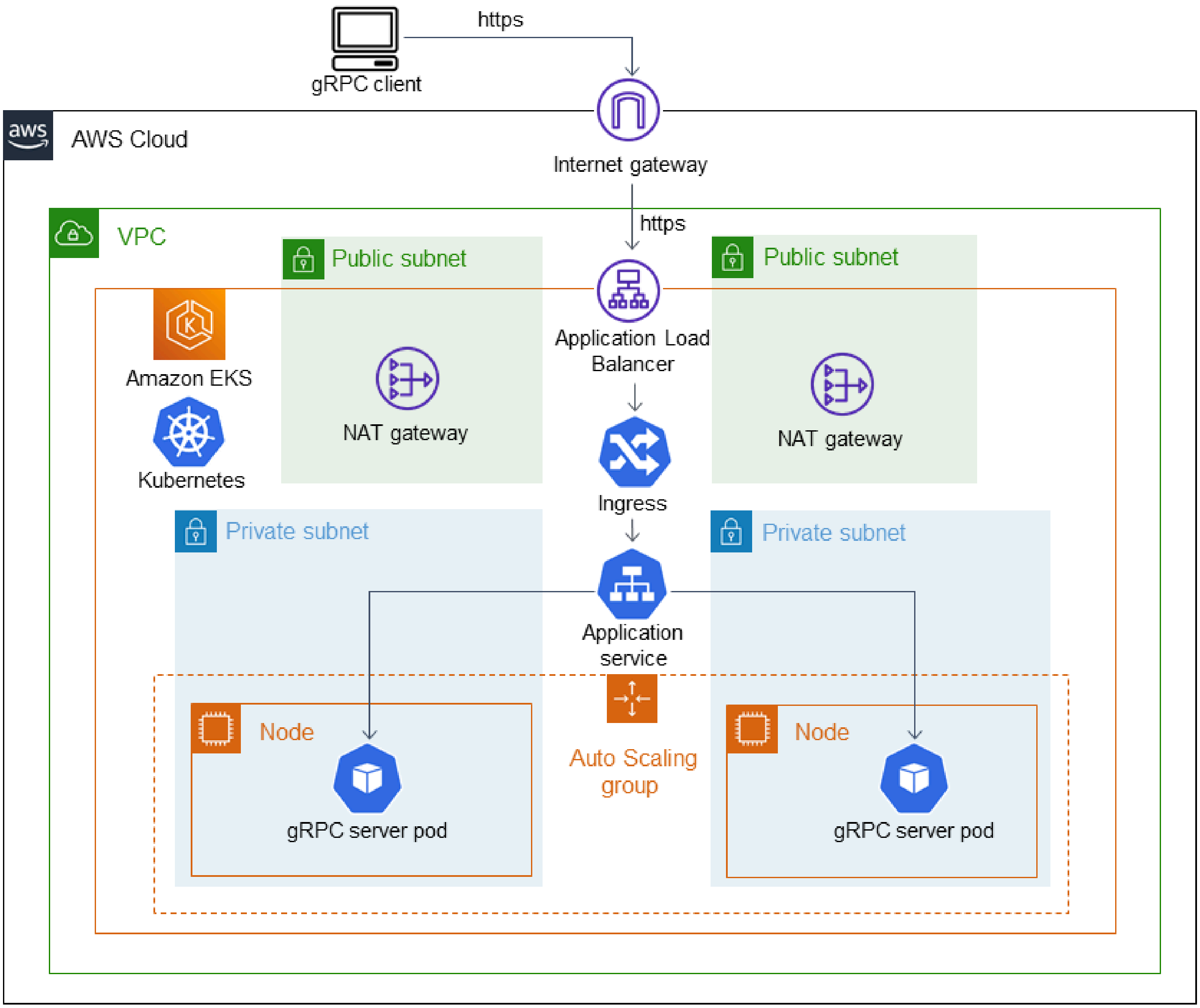
Prerequisites

- An active AWS account.
- Docker, installed and configured on Linux, macOS, or Windows.
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows.
- eksctl, installed and configured on Linux, macOS, or Windows.
- `kubectl`, installed and configured to access resources on your Amazon EKS cluster. For more information, see Installing or updating kubectl in the Amazon EKS documentation.
- gRPCurl, installed and configured.
- A new or existing Amazon EKS cluster. For more information, see Getting started with Amazon EKS.

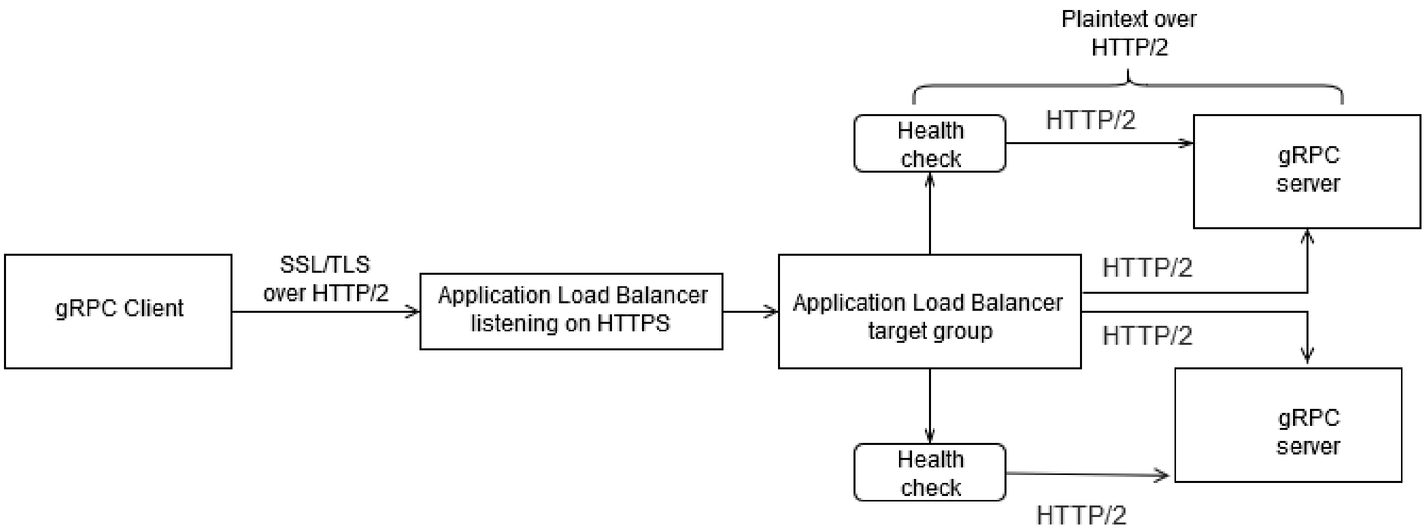
- Your computer terminal configured to access the Amazon EKS cluster. For more information, see Configure your computer to communicate with your cluster in the Amazon EKS documentation.
- AWS Load Balancer Controller, provisioned in the Amazon EKS cluster.
- An existing DNS host name with a valid SSL or SSL/TLS certificate. You can obtain a certificate for your domain by using AWS Certificate Manager (ACM) or uploading an existing certificate to ACM. For more information about these two options, see Requesting a public certificate and Importing certificates into AWS Certificate Manager in the ACM documentation.

Architecture

The following diagram shows the architecture implemented by this pattern.



The following diagram shows a workflow where SSL/TLS traffic is received from a gRPC client that offloads to an Application Load Balancer. Traffic is forwarded in plaintext to the gRPC server because it comes from a virtual private cloud (VPC).



Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command line shell.
- Elastic Load Balancing distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.

Tools

- eksctl is a simple CLI tool for creating clusters on Amazon EKS.
- kubectl is a command line utility for running commands against Kubernetes clusters.
- AWS Load Balancer Controller helps you manage AWS Elastic Load Balancers for a Kubernetes cluster.
- gRPCurl is a command line tool that helps you interact with gRPC services.

Code repository

The code for this pattern is available in the GitHub grpc-traffic-on-alb-to-eks repository.

Epics

Build and push the gRPC server's Docker image to Amazon ECR

Task	Description	Skills required
Create an Amazon ECR repository.	Sign in to the AWS Management Console, open the Amazon ECR console, and then create an Amazon ECR repository. For more information, see Creating a repository in the Amazon ECR documentation. Make sure that you record the Amazon ECR repository's URL. You can also create an Amazon ECR repository with AWS CLI by running the following command: `aws ecr create-repository --repository-name helloworld-grpc`	Cloud administrator
Build the Docker image.	1. Clone the GitHub grpc-traffic-on-alb-to-eks repository. `git clone https://github.com/aws-samples/grpc-traffic-on-alb-to-eks.git` 2. From the root directory of the repository, make sure	DevOps engineer

Task	Description	Skills required
	that the Dockerfile exists and then run the following command to build the Docker image: `docker build -t <amazon_ecr_repository_url>:<Tag> .` **Important** Make sure that you replace `<amazon_ecr_repository_url>` with the URL of the Amazon ECR repository that you created earlier.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Push the Docker image to Amazon ECR.</td><td>

1. Run the following command to log in to the Amazon ECR repository:

```
aws ecr get-login-password --region us-east-1 --no-cli-auto-prompt | docker login --username AWS --password-stdin <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com
```

2. Push the Docker image to the Amazon ECR repository by running the following command:

```
docker push <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com/helloworld-grpc:1.0
```

Important

Make sure that you replace `<your_aws_account_id>` with your AWS account ID.

</td><td>DevOps engineer</td></tr>
</table>

Deploy the Kubernetes manifests to the Amazon EKS cluster

Task	Description	Skills required
Modify the values in the Kubernetes manifest file.	1. Modify the `grpc-sample.yaml` Kubernetes manifest file in the Kubernetes folder of the repository according to your requirements. You must modify the annotations and host name in the ingress resource. For a sample ingress resource, see the Additional information section. For more information about ingress annotations, see Ingress annotations in the Kubernetes documentation. 2. In the Kubernetes deployment resource, change the deployment resource's `image` to the uniform resource identifier (URI) for the Amazon ECR repository that you pushed the Docker image to. For a sample deployment resource, see the Additional information section.	DevOps engineer
Deploy the Kubernetes manifest file.	Deploy the `grpc-sample.yaml` file to the Amazon EKS cluster by running	DevOps engineer

Task	Description	Skills required
	the following `kubectl` command: `kubectl apply -f ./kubernetes/grpc-sample.yaml`	

Create the DNS record for the Application Load Balancer's FQDN

Task	Description	Skills required
Record the FQDN for the Application Load Balancer.	1. Run the following `kubectl` command to describe the Kubernetes ingress resource that manages the Application Load Balancer: `kubectl get ingress -n grpcserver` Sample output is provided in the [Additional information](#) section. In the output, the `HOSTS` field displays the DNS host name that the SSL certificates were created for. 2. Record the Application Load Balancer’s fully qualified domain name (FQDN) from the `Address` field of the output.	DevOps engineer

Task	Description	Skills required
	3. Create a DNS record that points to the Application Load Balancer's FQDN. If your DNS provider is Amazon Route 53, you can create an alias record that points to the Application Load Balancer's FQDN. For more information about this option, see [Choosing between alias and non-alias records](#) in the Route 53 documentation.	

Test the solution

Task	Description	Skills required
Test the gRPC server.	Use gRPCurl to test the endpoint by running the following command: `grpcurl grpc.example.com:443 list grpc.reflection.v1alpha.ServerReflection helloworld.helloworld` **Note** Replace `grpc.example.com` with your DNS name.	DevOps engineer

Task	Description	Skills required
Test the gRPC server using a gRPC client.	In the `helloworld_client_ssl.py` sample gRPC client, replace the host name from `grpc.example.com` with the host name used for the gRPC server. The following code sample shows the response from the gRPC server for the client's request: `python ./app/helloworld_client_ssl.py message: "Hello to gRPC server from Client" message: "Thanks for talking to gRPC server!! Welcome to hello world. Received message is \"Hello to gRPC server from Client\"" received: true` This shows that the client can talk to the server and that the connection is successful.	DevOps engineer

Clean up

Task	Description	Skills required
Remove the DNS record.	Remove the DNS record that points to the Application Load Balancer's FQDN that you created earlier.	Cloud administrator
Remove the load balancer.	On the [Amazon EC2 console](https://console.aws.amazon.com/ec2/), choose **Load Balancers**, and then remove the load balancer that the Kubernetes controller created for your ingress resource.	Cloud administrator
Delete the Amazon EKS cluster.	Delete the Amazon EKS cluster by using `eksctl`: `eksctl delete cluster -f ./eks.yaml`	AWS DevOps

Related resources

- [Network load balancing on Amazon EKS]()
- [Target groups for your Application Load Balancers]()

Additional information

Sample ingress resource:

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    alb.ingress.kubernetes.io/healthcheck-protocol: HTTP
    alb.ingress.kubernetes.io/ssl-redirect: "443"
```

```
    alb.ingress.kubernetes.io/backend-protocol-version: "GRPC"
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS":443}]'
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:<AWS-
Region>:<AccountId>:certificate/<certificate_ID>
  labels:
    app: grpcserver
    environment: dev
  name: grpcserver
  namespace: grpcserver
spec:
  ingressClassName: alb
  rules:
  - host: grpc.example.com # <----- replace this as per your host name for which the
 SSL certtficate is available in ACM
    http:
      paths:
      - backend:
          service:
            name: grpcserver
            port:
              number: 9000
        path: /
        pathType: Prefix
```

Sample deployment resource:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grpcserver
  namespace: grpcserver
spec:
  selector:
    matchLabels:
      app: grpcserver
  replicas: 1
  template:
    metadata:
      labels:
        app: grpcserver
    spec:
```

```
      containers:
      - name: grpc-demo
        image: <your_aws_account_id>.dkr.ecr.us-east-1.amazonaws.com/helloworld-
grpc:1.0   #<------- Change to the URI that the Docker image is pushed to
        imagePullPolicy: Always
        ports:
        - name: grpc-api
          containerPort: 9000
        env:
        - name: POD_IP
          valueFrom:
            fieldRef:
              fieldPath: status.podIP
      restartPolicy: Always
```

Sample output:

```
NAME              CLASS           HOSTS                          Address
 PORTS          AGE
 grpcserver      <none>      <DNS-HostName>                  <ELB-address>
 80            27d
```

Deploy and debug Amazon EKS clusters

Svenja Raether and Mathew George, Amazon Web Services

Summary

Containers are becoming an essential part of cloud native application development. Kubernetes provides an efficient way to manage and orchestrate containers. Amazon Elastic Kubernetes Service (Amazon EKS) is a fully-managed, certified Kubernetes conformant service for building, securing, operating, and maintaining Kubernetes clusters on Amazon Web Services (AWS). It supports running pods on AWS Fargate to provide on-demand, right-sized compute capacity.

It's important for developers and administrators to know debugging options when running containerized workloads. This pattern walks you through deploying and debugging containers on Amazon EKS with AWS Fargate. It includes creating, deploying, accessing, debugging, and cleaning up the Amazon EKS workloads.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Identity and Access Management (IAM) role configured with sufficient permissions to create and interact with Amazon EKS, IAM roles, and service linked roles
- AWS Command Line Interface (AWS CLI) installed on the local machine
- eksctl
- kubectl
- Helm

Limitations

- This pattern provides developers with useful debugging practices for development environments. It does not state best practices for production environments.
- If you are running Windows, use your operating system–specific commands for setting the environment variables.

Product versions used

- AWS CLI version 2
- kubectl version within one minor version difference of the Amazon EKS control plane that you're using
- eksctl latest version
- Helm v3

Architecture

Technology stack

- Application Load Balancer
- Amazon EKS
- AWS Fargate

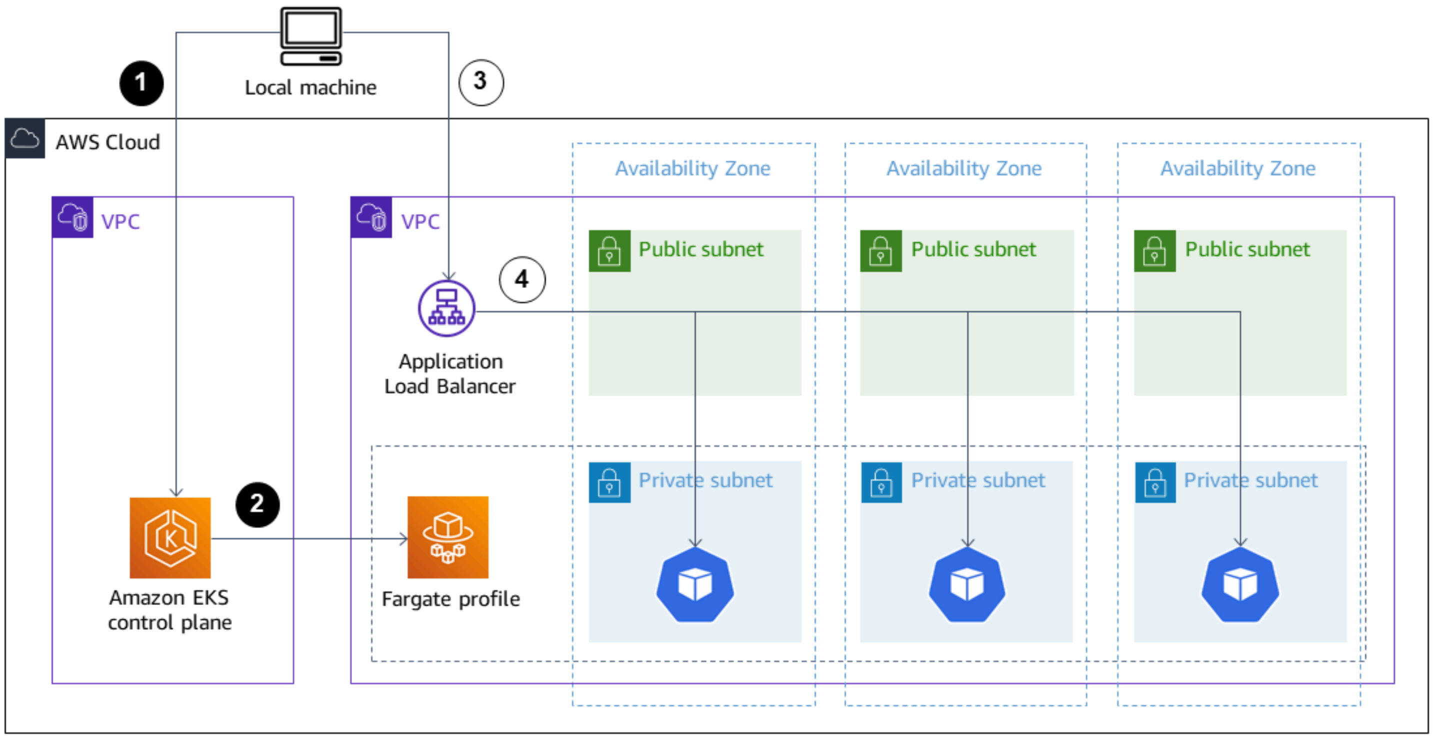
Target architecture

All resources shown in the diagram are provisioned by using `eksctl` and `kubectl` commands issued from a local machine. Private clusters must be run from an instance that is inside the private VPC.

The target architecture consists of an EKS cluster using the Fargate launch type. This provides on-demand, right-sized compute capacity without the need to specify server types. The EKS cluster has a control plane, which is used to manage the cluster nodes and workloads. The pods are provisioned into private VPC subnets spanning multiple Availability Zones. The Amazon ECR Public Gallery is referenced to retrieve and deploy an NGINX web server image to the cluster's pods.

The diagram shows how to access the Amazon EKS control plane using by `kubectl` commands and how to access the application by using the Application Load Balancer.

.



1. A local machine outside the AWS Cloud sends commands to the Kubernetes control plane inside an Amazon EKS managed VPC.
2. Amazon EKS schedules pods based on the selectors in the Fargate profile.
3. The local machine opens the Application Load Balancer URL in the browser.
4. The Application Load Balancer divides traffic between the Kubernetes pods in Fargate cluster nodes deployed in private subnets spanning multiple Availability Zones.

Tools

AWS services

- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes. This pattern also uses the eksctl command-line tool to work with Kubernetes clusters on Amazon EKS.
- AWS Fargate helps you run containers without needing to manage servers or Amazon Elastic Compute Cloud (Amazon EC2) instances. It's used in conjunction with Amazon Elastic Container Service (Amazon ECS).

- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones. This pattern uses the AWS Load Balancer Controller controlling component to create the Application Load Balancer when a Kubernetes ingress is provisioned. The Application Load Balancer distributes incoming traffic among multiple targets.

Other tools

- Helm is an open-source package manager for Kubernetes. In this pattern, Helm is used to install the AWS Load Balancer Controller.
- Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.
- NGINX is a high-performance web and reverse proxy server.

Epics

Create an EKS cluster

Task	Description	Skills required
Create the files.	Using the code in the *Additional information* section, create the following files: • `clusterconfig-fargate.yaml` • `nginx-deployment.yaml` • `nginx-service.yaml` • `nginx-ingress.yaml` • `index.html`	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required

Set environment variables.	**Note** If a command fails because of previous unfinished tasks, wait a few seconds, and then run the command again.
	
	This pattern uses the AWS Region and cluster name that are defined in the file `clusterconfig-fargate.yaml` . Set the same values as environment variables to reference them in further commands.
	
	`export AWS_REGION="us-east-1"`
	`export CLUSTER_NAME="my-fargate"`
		App developer, AWS DevOps, AWS systems administrator

Task	Description	Skills required
Create an EKS cluster.	To create an EKS cluster that uses the specifications from the `clusterconfig-fargate.yaml` file, run the following command. `eksctl create cluster -f clusterconfig-fargate.yaml` The file contains the `ClusterConfig` , which provisions a new EKS cluster named `my-fargate-cluster` in the `us-east-1` Region and one default Fargate profile (`fp-default`). The default Fargate profile is configured with two selectors (`default` and `kube-system`).	App developer, AWS DevOps, AWS administrator

Task	Description	Skills required

Check the created cluster.	To check the created cluster, run the following command.
	`eksctl get cluster --output yaml`
	The output should be the following.
	`- Name: my-fargate`
	`  Owned: "True"`
	`  Region: us-east-1`
	Check the created Fargate profile by using the `CLUSTER_NAME` .
	`eksctl get fargateprofile --cluster $CLUSTER_NAME --output yaml`
	This command displays information about the resources. You can use the information to verify the created cluster. The output should be the following.
	`- name: fp-default`
	`  podExecutionRoleARN: arn:aws:iam::<YOUR-ACCOUNT-ID>:role/eksctl-my-fargate-cluster-FargatePodExecutionRole-xxx`
	`  selectors:`
		App developer, AWS DevOps, AWS systems administrator

Task	Description	Skills required

	``
	  - namespace: default
	  - namespace: kube-
	system
	  status: ACTIVE
	  subnets:
	  - subnet-aaa
	  - subnet-bbb
	  - subnet-ccc
	``	

Deploy a container

Task	Description	Skills required

Deploy the NGINX web server.	To apply the NGINX web server deployment on the cluster, run the following command.
	`kubectl apply -f ./nginx-deployment.yaml`
	The output should be the following.
	`deployment.apps/nginx-deployment created`
	The deployment includes three replicas of the NGINX image taken from the Amazon ECR Public Gallery. The image is deployed to the default namespace and exposed on port 80 on the running pods.
		App developer, AWS DevOps, AWS systems administrator

Task	Description	Skills required

Check the deployment and pods.	(Optional) Check the deployment. You can verify the status of your deployment with the following command.
	`kubectl get deployment`
	The output should be the following.
	`NAME READY UP-TO-DATE AVAILABLE AGE`
	`nginx-deployment 3/3 3 3 7m14s`
	A pod is a deployable object in Kubernetes, containing one or more containers. To list all pods, run the following command.
	`kubectl get pods`
	The output should be the following.
	`NAME READY STATUS RESTARTS AGE`
	`nginx-deployment-xxxx-aaa 1/1 Running 0 94s`
		App developer, AWS DevOps, AWS administrator

Task	Description	Skills required
	``` nginx-deployment-xxxx- bbb 1/1 Running 0 94s nginx-deployment-xxxx- ccc 1/1 Running 0 94s ```	

Scale the deployment.	To scale the deployment from the three replicas that were specified in `deployment.yaml` to four replicas, use the following command.
	`kubectl scale deployment nginx-deployment --replicas 4`
	The output should be the following.
	`deployment.apps/nginx-deployment scaled`
		App developer, AWS DevOps, AWS systems administrator

Deploy an AWS Load Balancer Controller

Task	Description	Skills required
Set environment variables.	Describe the cluster's CloudFormation stack to retrieve information about its VPC. `aws cloudformation describe-stacks --stack-name eksctl-$CLUSTER_NAME-cluste`	App developer, AWS DevOps, AWS systems administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

```
r --query "Stacks[0
].Outputs[?OutputK
ey==\`VPC\`].Outpu
tValue"
```

The output should be the following.

```
[
    "vpc-<YOUR-VPC-ID>
"
]
```

Copy the VPC ID and export it as an environment variable.

```
export VPC_ID="vpc-
<YOUR-VPC-ID>"
```

</td><td></td></tr>
<tr><td>Configure IAM for the cluster service account.</td><td>

Use the `AWS_REGION` and `CLUSTER_NAME` from the earlier epic to create an IAM Open ID Connect provider for the cluster.

```
eksctl utils associate-
iam-oidc-provider \
--region $AWS_REGION \
--cluster $CLUSTER_
NAME \
--approve
```

</td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr>
<td>Download and create the IAM policy.</td>
<td>

Download the IAM policy for the AWS Load Balancer Controller that allows it to make calls to AWS APIs on your behalf.

```
curl -o iam-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/main/docs/install/iam_policy.json
```

Create the policy in your AWS account by using the AWS CLI.

```
aws iam create-policy \
--policy-name AWSLoadBalancerControllerIAMPolicy \
--policy-document file://iam-policy.json
```

You should see the following output.

```
{
"Policy": {
       "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
       "PolicyId": "<YOUR_POLICY_ID>",
       "Arn": "arn:aws:iam::<YOUR-ACCOUNT
```

</td>
<td>App developer, AWS DevOps, AWS systems administrator</td>
</tr>
</table>

Task	Description	Skills required

	``
	-ID>:policy/AWSLoa
	dBalancerControlle
	rIAMPolicy",
	        "Path": "/",
	         "DefaultV
	ersionId": "v1",
	         "Attachme
	ntCount": 0,
	         "Permissi
	onsBoundaryUsageCo
	unt": 0,
	         "IsAttachable":
	 true,
	         "CreateDate":
	 "<YOUR-DATE>",
	         "UpdateDate":
	 "<YOUR-DATE>"
	    }
	}
	``
	Save the Amazon Resource Name (ARN) of the policy as `$POLICY_ARN` .
	``
	export POLICY_AR
	N="arn:aws:iam::<Y
	OUR-ACCOUNT-ID>:po
	licy/AWSLoadBalanc
	erControllerIAMPol
	icy"
	``	

Task	Description	Skills required

Create an IAM service account.	Create an IAM service account named `aws-load-balancer-controller` in the `kube-system` namespace. Use the `CLUSTER_NAME`, `AWS_REGION`, and `POLICY_ARN` that you previously configured.
	```eksctl create iamserviceaccount \ --cluster=$CLUSTER_NAME \ --region=$AWS_REGION \ --attach-policy-arn=$POLICY_ARN \ --namespace=kube-system \ --name=aws-load-balancer-controller \ --override-existing-serviceaccounts \ --approve```
	Verify the creation.
	```eksctl get iamserviceaccount \ --cluster $CLUSTER_NAME \ --name aws-load-balancer-controller \ --namespace kube-system \ --output yaml```
		App developer, AWS DevOps, AWS systems administrator

Task	Description	Skills required

	The output should be the following.
	``
	- metadata:
	    name: aws-load-balancer-controller
	    namespace: kube-system
	  status:
	    roleARN: arn:aws:iam::<YOUR-ACCOUNT-ID>:role/eksctl-my-fargate-addon-iamserviceaccount-ku-Role1-<YOUR-ROLE-ID>
	  wellKnownPolicies:
	    autoScaler: false
	    awsLoadBalancerController: false
	    certManager: false
	    ebsCSIController: false
	    efsCSIController: false
	    externalDNS: false
	    imageBuilder: false
	``	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install the AWS Load Balancer Controller.</td><td>Update the Helm repository.

<pre><code>helm repo update</code></pre>

Add the Amazon EKS chart repository to the Helm repo.

<pre><code>helm repo add eks
 https://aws.github
.io/eks-charts</code></pre>

Apply the Kubernetes custom resource definitions (CRDs) that are used by the AWS Load Balancer Controller eks-chart in the background.

<pre><code>kubectl apply -k
 "github.com/aws/ek
s-charts/stable/aw
s-load-balancer-co
ntroller//crds?ref
=master"</code></pre>

The output should be the following.

<pre><code>customresourcedefi
nition.apiextensio
ns.k8s.io/ingressc
lassparams.elbv2.k
8s.aws created
customresourcedefin
ition.apiextension
s.k8s.io/targetgro
upbindings.elbv2.k
8s.aws created</code></pre></td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
</table>

Task	Description	Skills required

	Install the Helm chart, using the environment variables that you set previously.
	
	`helm install aws-load-balancer-controller eks/aws-load-balancer-controller \ --set clusterName=$CLUSTER_NAME \ --set serviceAccount.create=false \ --set region=$AWS_REGION \ --set vpcId=$VPC_ID \ --set serviceAccount.name=aws-load-balancer-controller \ -n kube-system`
	
	The output should be the following.
	
	`NAME: aws-load-balancer-controller LAST DEPLOYED: <YOUR-DATE> NAMESPACE: kube-system STATUS: deployed REVISION: 1 TEST SUITE: None NOTES: AWS Load Balancer controller installed!`	

Task	Description	Skills required

Create an NGINX service.	Create a service to expose the NGINX pods by using the `nginx-service.yaml` file.
	`kubectl apply -f nginx-service.yaml`
	The output should be the following.
	`service/nginx-service created`
		App developer, AWS DevOps, AWS systems administrator

Create the Kubernetes ingress resource.	Create a service to expose the Kubernetes NGINX ingress by using the `nginx-ingress.yaml` file.
	`kubectl apply -f nginx-ingress.yaml`
	The output should be the following.
	`ingress.networking.k8s.io/nginx-ingress created`
		App developer, AWS DevOps, AWS systems administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Get the load balancer URL.</td><td>To retrieve the ingress information, use the following command.
<pre>kubectl get ingress
 nginx-ingress</pre>
The output should be the following.
<pre>NAME   CLASS
 HOSTS   ADDRESS

          PORTS   AGE
nginx-ingress   <none>
   *   k8s-defau
lt-nginxing-xxx.us
-east-1.elb.amazon
aws.com   80   80s</pre>
Copy the <code>ADDRESS</code> (for example, <code>k8s-default-nginxing-xxx.us-east-1.elb.amazonaws.com</code>) from the output, and paste it into your browser to access the <code>index.html</code> file.</td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
</table>

Debug running containers

Task	Description	Skills required
Select a pod.	List all pods, and copy the desired pod's name.	App developer, AWS DevOps, AWS systems administrator

Task	Description	Skills required

	`kubectl get pods`
	
	The output should be the following.
	
	`NAME READY STATUS RESTARTS AGE`
	`nginx-deployment-xxxx-aaa 1/1 Running 0 55m`
	`nginx-deployment-xxxx-bbb 1/1 Running 0 55m`
	`nginx-deployment-xxxx-ccc 1/1 Running 0 55m`
	`nginx-deployment-xxxx-ddd 1/1 Running 0 42m`
	
	This command lists the existing pods and additional information.
	
	If you are interested in a specific pod, fill in the name of the pod you are interested in for the `POD_NAME` variable or set it as an environment variable. Otherwise, omit this parameter to look up all resources.	

Task	Description	Skills required
	`export POD_NAME="nginx-deployment-<YOUR-POD-NAME>"`	
Access the logs.	Get the logs from the pod that you want to debug. `kubectl logs $POD_NAME`	App developer, AWS systems administrator, AWS DevOps

Forward the NGINX port.	Use port-forwarding to map the pod's port for accessing the NGINX web server to a port on your local machine.
	
	`kubectl port-forward deployment/nginx-deployment 8080:80`
	
	In your browser, open the following URL.
	
	`http://localhost:8080`
	
	The `port-forward` command provides access to the `index.html` file without making it publicly available over a load balancer. This is useful for accessing the running application while debugging it. You can stop the port-forwarding by pressing the keyboard command **Ctrl+C**.
		App developer, AWS DevOps, AWS systems administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Run commands within the pod.</td><td>To look at the current <code>index.html</code> file, use the following command.
<pre>kubectl exec $POD_NAME
 -- cat /usr/share/
nginx/html/index.html</pre>You can use the <code>exec</code> command to issue any command directly in the pod. This is useful for debugging running applications.</td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
<tr><td>Copy files to a pod.</td><td>Remove the default <code>index.html</code> file on this pod.
<pre>kubectl exec $POD_NAME
 -- rm /usr/share/
nginx/html/index.html</pre>Upload the customized local file <code>index.html</code> to the pod.
<pre>kubectl cp index.html
 $POD_NAME:/usr/share/
nginx/html/</pre>You can use the <code>cp</code> command to change or add files directly to any of the pods.</td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Use port-forwarding to display the change.</td><td>Use port-forwarding to verify the changes that you made to this pod.
<pre>kubectl port-forward
 pod/$POD_NAME 8080:80</pre>Open the following URL in your browser.
<pre>http://localhost:8080</pre>The applied changes to the <code>index.html</code> file should be visible in the browser.</td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
</table>

Delete resources

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Delete the load balancer.</td><td>Delete the ingress.
<pre>kubectl delete ingress/n
ginx-ingress</pre>The output should be the following.
<pre>ingress.networking
.k8s.io "nginx-in
gress" deleted</pre>Delete the service.</td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
</table>

Task	Description	Skills required

	```kubectl delete service/nginx-service```
	The output should be the following.
	```service "nginx-service" deleted```
	Delete the load balancer controller.
	```helm delete aws-load-balancer-controller -n kube-system```
	The output should be the following.
	```release "aws-load-balancer-controller" uninstalled```
	Delete the service account.
	```eksctl delete iamserviceaccount --cluster $CLUSTER_NAME --namespace kube-system --name aws-load-balancer-controller```	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Delete the deployment.</td><td>To delete the deployment resources, use the following command.
<pre>kubectl delete deploy/nginx-deployment</pre>The output should be the following.
<pre>deployment.apps "nginx-deployment" deleted</pre></td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
<tr><td>Delete the cluster.</td><td>Delete the EKS cluster by using the following command, where <code>my-fargate</code> is the cluster name.
<pre>eksctl delete cluster --name $CLUSTER_NAME</pre>This command deletes the entire cluster, including all associated resources.</td><td>App developer, AWS DevOps, AWS systems administrator</td></tr>
<tr><td>Delete the IAM policy.</td><td>Delete the previously created policy by using the AWS CLI.
<pre>aws iam delete-policy --policy-arn $POLICY_ARN</pre></td><td>App developer, AWS administrator, AWS DevOps</td></tr>
</table>

Troubleshooting

Issue	Solution
You receive an error message upon cluster creation stating that your targeted Availability Zone doesn't have sufficient capacity to support the cluster. You should see a message similar to the following. `Cannot create cluster 'my-fargate' because us-east-1e, the targeted availability zone, does not currently have sufficient capacity to support the cluster. Retry and choose from these availability zones: us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1f`	Create the cluster again using the recommended Availability Zones from the error message. Specify a list of Availability Zones in the last line of your `clusterconfig-fargate.yaml` file (for example, `availabilityZones: ["us-east-1a", "us-east-1b", "us-east-1c"]`).

Related resources

- Amazon EKS documentation
- Application load balancing on Amazon EKS
- EKS Best Practices Guides
- AWS Load Balancer Controller documentation
- eksctl documentation
- Amazon ECR Public Gallery NGINX image
- Helm documentation
- Debug Running Pods (Kubernetes documentation)
- Amazon EKS Workshop
- EKS cluster creation errors

Additional information

clusterconfig-fargate.yaml

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-fargate
  region: us-east-1

fargateProfiles:
  - name: fp-default
    selectors:
      - namespace: default
      - namespace: kube-system
```

nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "nginx-deployment"
  namespace: "default"
spec:
  replicas: 3
  selector:
    matchLabels:
      app: "nginx"
  template:
    metadata:
      labels:
        app: "nginx"
    spec:
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:latest
        ports:
        - containerPort: 80
```

nginx-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  annotations:
```

```
    alb.ingress.kubernetes.io/target-type: ip
  name: "nginx-service"
  namespace: "default"
spec:
  ports:
  - port: 80
    targetPort: 80
    protocol: TCP
  type: NodePort
  selector:
    app: "nginx"
```

nginx-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: "default"
  name:  "nginx-ingress"
  annotations:
    kubernetes.io/ingress.class: alb
    alb.ingress.kubernetes.io/scheme: internet-facing
spec:
  rules:
    - http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: "nginx-service"
              port:
                number: 80
```

index.html

```
<!DOCTYPE html>
<html>

<body>
    <h1>Welcome to your customized nginx!</h1>
    <p>You modified the file on this running pod</p>
</body>
```

```
</html>
```

Deploy containers by using Elastic Beanstalk

Thomas Scott and Jean-Baptiste Guillois, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, AWS Elastic Beanstalk supports Docker as an available platform, so that containers can run with the created environment. This pattern shows how to deploy containers using the Elastic Beanstalk service. The deployment of this pattern will use the web server environment based on the Docker platform.

To use Elastic Beanstalk for deploying and scaling web applications and services, you upload your code and the deployment is automatically handled. Capacity provisioning, load balancing, automatic scaling, and application health monitoring are also included. When you use Elastic Beanstalk, you can take full control over the AWS resources that it creates on your behalf. There is no additional charge for Elastic Beanstalk. You pay only for the AWS resources that are used to store and run your applications.

This pattern includes instructions for deployment using the [AWS Elastic Beanstalk Command Line Interface (EB CLI)] and the AWS Management Console.

Use cases

Use cases for Elastic Beanstalk include the following:

- Deploy a prototype environment to demo a frontend application. (This pattern uses a Dockerfile as the example.)
- Deploy an API to handle API requests for a given domain.
- Deploy an orchestration solution using Docker-Compose (`docker-compose.yml` is not used as the practical example in this pattern).

Prerequisites and limitations

Prerequisites

- An AWS account
- AWS EB CLI locally installed
- Docker installed on a local machine

Limitations

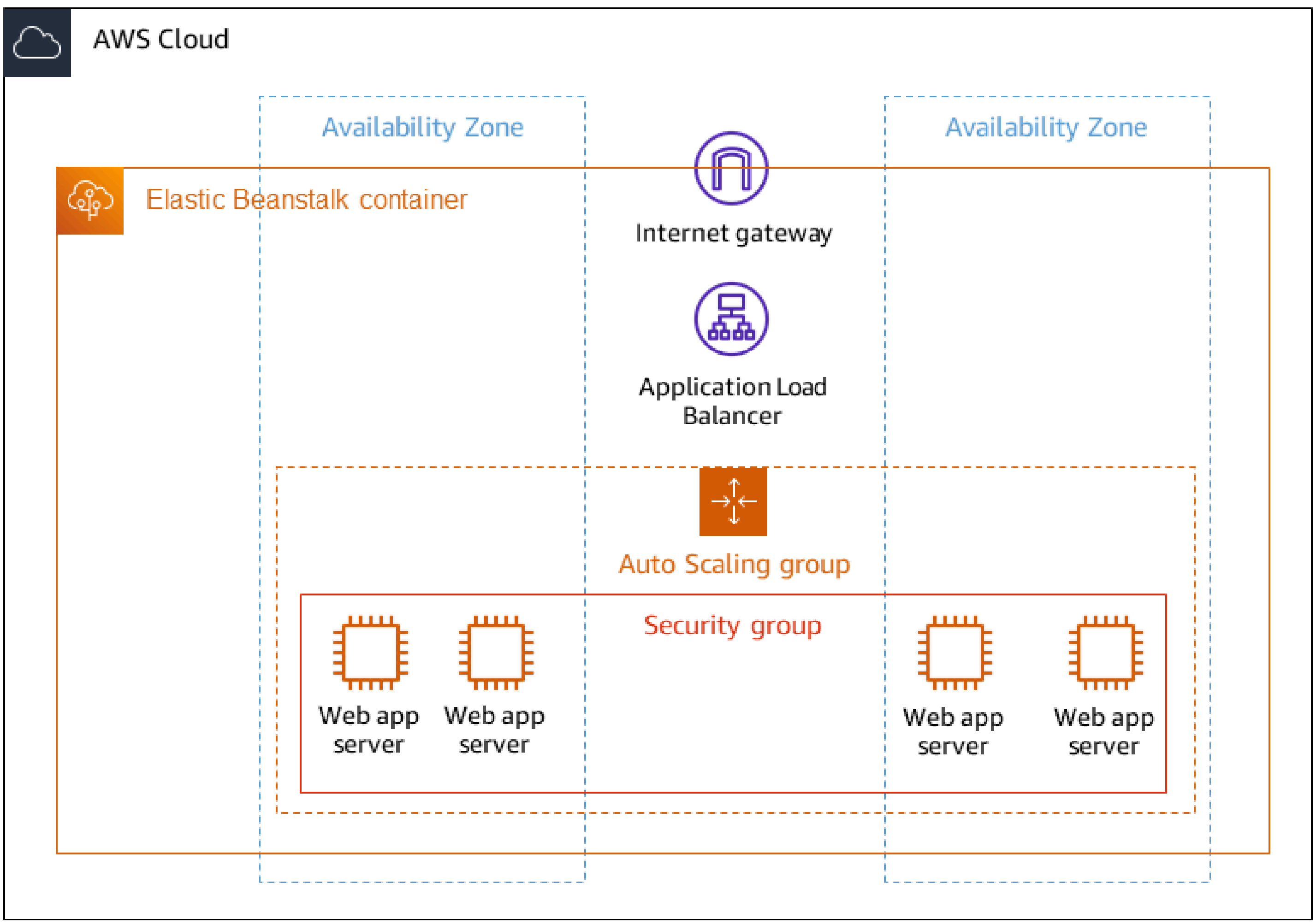
- There is a Docker pull limit of 100 pulls per 6 hours per IP address on the free plan.

Architecture

Target technology stack

- Amazon Elastic Compute Cloud (Amazon EC2) instances
- Security group
- Application Load Balancer
- Auto Scaling group

Target architecture



Automation and scale

AWS Elastic Beanstalk can automatically scale based on the number of requests made. AWS resources created for an environment include one Application Load Balancer, an Auto Scaling group, and one or more Amazon EC2 instances.

The load balancer sits in front of the Amazon EC2 instances, which are part of the Auto Scaling group. Amazon EC2 Auto Scaling automatically starts additional Amazon EC2 instances to accommodate increasing load on your application. If the load on your application decreases, Amazon EC2 Auto Scaling stops instances, but it keeps at least one instance running.

Automatic scaling triggers

The Auto Scaling group in your Elastic Beanstalk environment uses two Amazon CloudWatch alarms to initiate scaling operations. The default triggers scale when the average outbound network traffic from each instance is higher than 6 MB or lower than 2 MB over a period of five minutes. To use Amazon EC2 Auto Scaling effectively, configure triggers that are appropriate for your application, instance type, and service requirements. You can scale based on several statistics including latency, disk I/O, CPU utilization, and request count. For more information, see Auto Scaling triggers.

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS EB Command Line Interface (EB CLI) is a command-line client that you can use to create, configure, and manage Elastic Beanstalk environments.
- Elastic Load Balancing distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones.

Other services

- Docker packages software into standardized units called containers that include libraries, system tools, code, and runtime.

Code

The code for this pattern is available in the GitHub Cluster Sample Application repository.

Epics

Build with a Dockerfile

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Clone the remote repository.</td><td>

- To clone the repository, run the command `git clone https://github.com/aws-samples/cluster-sample-app.git`.< /p>

</td><td>App developer, AWS administrator, AWS DevOps</td></tr>
<tr><td>Initialize the Elastic Beanstalk Docker project.</td><td>

1. Create a file called `aws.json` at the root.
2. In the `aws.json` file, add the following code.

```
{
         "AWSEBDockerrunVersion":"1",
         "Image":{
            "Name":"cluster-sample-app"
         },
         "Ports":[
            {
               "ContainerPort":80,
               "HostPort":8080
            }
         ]
      }
```

</td><td>App developer, AWS administrator, AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	3. Run the command `eb init -p docker` at the root of the project.	
Test the project locally.	1. Run the command `eb local run` at the root of the project. 2. Test the application by navigating to `http://localhost`.	App developer, AWS administrator, AWS DevOps

Deploy using EB CLI

Task	Description	Skills required
Run deployment command	1. Run the command `eb create docker-sample-cluster-app` at the root of the project.	App developer, AWS administrator, AWS DevOps
Access the deployed version.	After the deployment command has finished, access the project using the `eb open` command.	App developer, AWS administrator, AWS DevOps

Deploy using the console

Task	Description	Skills required
Deploy the application by using the browser.	1. Open the console. 2. Navigate to the Elastic Beanstalk console. 3. Choose **Create Application.**	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required
	4. For the **Application Name**, enter **Cluster-Sample-App**. 5. Choose **Docker** as the platform. 6. Choose **Upload your code.** 7. Choose your local .zip file (in the root of the cloned project) or a public Amazon Simple Storage Service (Amazon S3) URL.	
Access the deployed version.	After deployment, access the deployed application, and choose the URL provided.	App developer, AWS administrator, AWS DevOps

Related resources

- Web server environments
- Install the EB CLI on macOS
- Manually install the EB CLI

Additional information

Advantages of using Elastic Beanstalk

- Automatic infrastructure provisioning
- Automatic management of the underlying platform
- Automatic patching and updates to support the application
- Automatic scaling of the application
- Ability to customize the number of nodes
- Ability to access the infrastructure components if needed
- Ease of deployment over other container deployment solutions

Generate a static outbound IP address using a Lambda function, Amazon VPC, and a serverless architecture

Thomas Scott, Amazon Web Services

Summary

This pattern describes how to generate a static outbound IP address in the Amazon Web Services (AWS) Cloud by using a serverless architecture. Your organization can benefit from this approach if it wants to send files to a separate business entity by using Secure File Transfer Protocol (SFTP). This means that the business entity must have access to an IP address that allows files through its firewall.

The pattern's approach helps you create an AWS Lambda function that uses an [Elastic IP address](#) as the outbound IP address. By following the steps in this pattern, you can create a Lambda function and a virtual private cloud (VPC) that routes outbound traffic through an internet gateway with a static IP address. To use the static IP address, you attach the Lambda function to the VPC and its subnets.

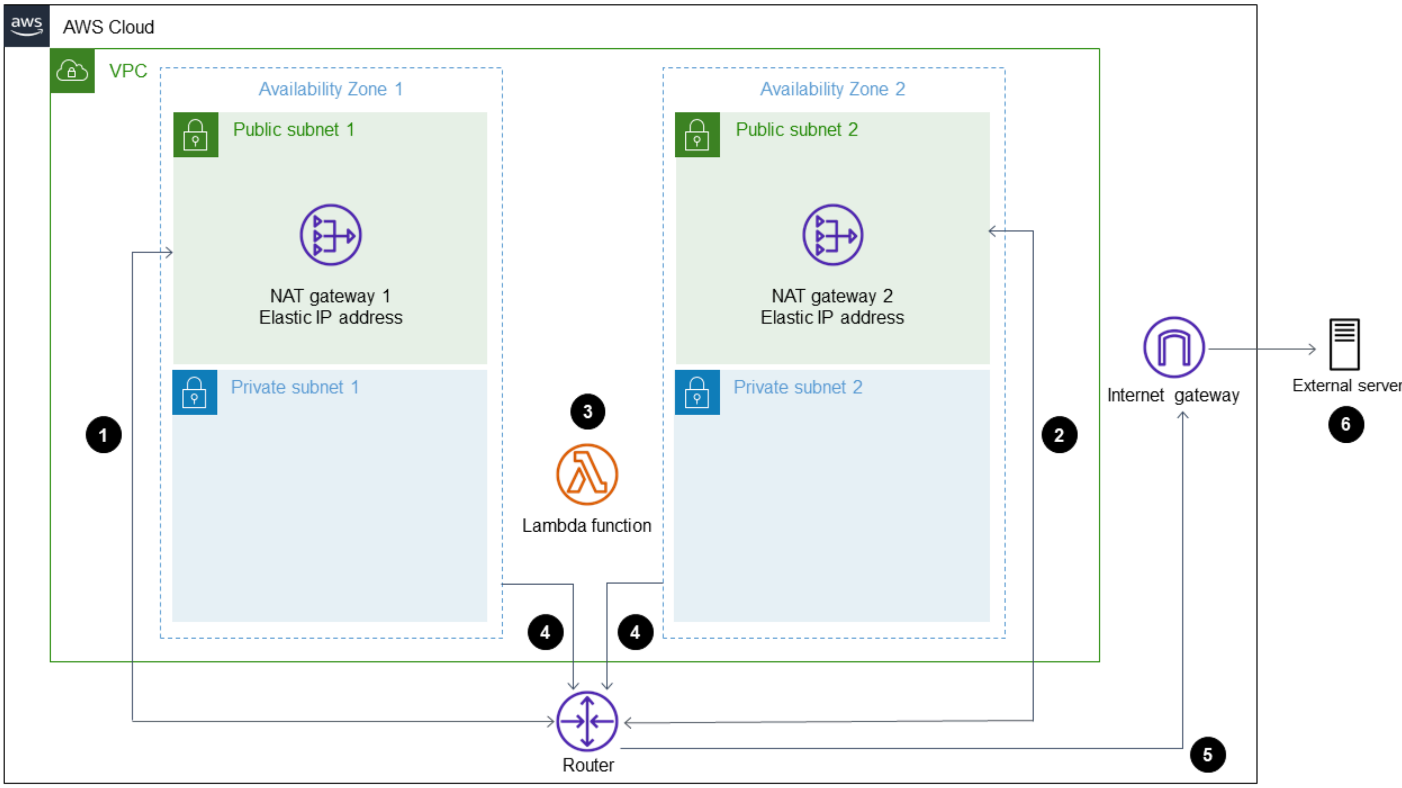
Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Identity and Access Management (IAM) permissions to create and deploy a Lambda function, and to create a VPC and its subnets. For more information about this, see Execution role and user permissions in the AWS Lambda documentation.
- If you plan to use infrastructure as code (IaC) to implement this pattern's approach, you need an integrated development environment (IDE) such as AWS Cloud9. For more information about this, see What is AWS Cloud9? in the AWS Cloud9 documentation.

Architecture

The following diagram shows the serverless architecture for this pattern.



The diagram shows the following workflow:

1. Outbound traffic leaves `NAT gateway 1` in `Public subnet 1`.
2. Outbound traffic leaves `NAT gateway 2` in `Public subnet 2`.
3. The Lambda function can run in `Private subnet 1` or `Private subnet 2`.
4. `Private subnet 1` and `Private subnet 2` route traffic to the NAT gateways in the public subnets.
5. The NAT gateways send outbound traffic to the internet gateway from the public subnets.
6. Outbound data is transferred from the internet gateway to the external server.

Technology stack

- Lambda
- Amazon Virtual Private Cloud (Amazon VPC)

Automation and scale

You can ensure high availability (HA) by using two public and two private subnets in different Availability Zones. Even if one Availability Zone becomes unavailable, the pattern’s solution continues to work.

Tools

- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- Amazon VPC – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Create a new VPC

Task	Description	Skills required
Create a new VPC.	Sign in to the AWS Management Console, open the Amazon VPC console, and then create a VPC named `Lambda VPC` that has `10.0.0.0/25` as the IPv4 CIDR range. For more information about creating a VPC, see Getting started with Amazon VPC in the Amazon VPC documentation.	AWS administrator

Create two public subnets

Task	Description	Skills required
Create the first public subnet.	1. On the Amazon VPC console, choose **Subnets** and then choose **Create Subnet**. 2. For **Name tag**, enter `public-one` . 3. For **VPC**, choose `Lambda VPC`. 4. Choose an Availability Zone and record it. 5. For **IPv4 CIDR block**, enter `10.0.0.0/28` and then choose **Create subnet**.	AWS administrator
Create the second public subnet.	1. On the Amazon VPC console, choose **Subnets** and then choose **Create Subnet**. 2. For **Name tag**, enter `public-two` . 3. For **VPC**, choose `Lambda VPC`. 4. **Important** Choose an Availability Zone and record it. : You cannot use the Availability Zone that contains the `public-one` subnet.	AWS administrator

Task	Description	Skills required
	5. For **IPv4 CIDR block**, enter `10.0.0.16/28` and then choose **Create subnet**.	

Create two private subnets

Task	Description	Skills required
Create the first private subnet.	1. On the Amazon VPC console, choose **Subnets** and then choose **Create Subnet**. 2. For **Name tag**, enter `private-one` . 3. For **VPC**, choose `Lambda VPC`. 4. Choose the Availability Zone that contains the `public-one` subnet that you created earlier. 5. For **IPv4 CIDR block**, enter `10.0.0.32/28` and then choose **Create subnet**.	AWS administrator
Create the second private subnet.	1. On the Amazon VPC console, choose **Subnets** and then choose **Create Subnet**. 2. For **Name tag**, enter `private-two` . 3. For **VPC**, choose `Lambda VPC`.	AWS administrator

Task	Description	Skills required
	4. Choose the same Availability Zone that contains the `public-two` subnet that you created earlier. 5. For **IPv4 CIDR block**, enter `10.0.0.64/28` and then choose **Create subnet**.	

Create two Elastic IP addresses for your NAT gateways

Task	Description	Skills required
Create the first Elastic IP address.	1. On the Amazon VPC console, choose **Elastic IPs** and then choose **Allocate new address**. 2. Choose **Allocate** and record the **Allocation ID** for your newly created Elastic IP address. **Note** This Elastic IP address is used for your first NAT gateway.	AWS administrator
Create the second Elastic IP address.	1. On the Amazon VPC console, choose **Elastic IPs** and then choose **Allocate new address**. 2. Choose **Allocate** and record the **Allocatio**	AWS administrator

Task	Description	Skills required
	n ID for this second Elastic IP address. **Note** This Elastic IP address is used for your second NAT gateway.	

Create an internet gateway

Task	Description	Skills required
Create an internet gateway.	1. On the Amazon VPC console, choose **Internet Gateways** and then choose **Create internet gateway**. 2. Enter `Lambda internet gateway` as the name and then choose **Create internet gateway**. Make sure that you record the internet gateway ID.	AWS administrator
Attach the internet gateway to the VPC.	Select the internet gateway that you just created, and then choose **Actions, Attach to VPC**.	AWS administrator

Create two NAT gateways

Task	Description	Skills required
Create the first NAT gateway.	1. On the Amazon VPC console, choose **NAT Gateways** and then choose **Create NAT Gateway**. 2. Enter `nat-one` as the NAT gateway name. 3. Choose `public-one` as the subnet to create the NAT gateway in. 4. For **Connectivity type**, choose **Public**. 5. For **Elastic IP allocation ID**, choose the first Elastic IP address that you created earlier and associate it with the NAT gateway. 6. Choose **Create NAT gateway**.	AWS administrator
Create the second NAT gateway.	1. On the Amazon VPC console, choose **NAT Gateways** and then choose **Create NAT Gateway**. 2. Enter `nat-two` as the NAT gateway name. 3. Choose `public-two` as the subnet to create the NAT gateway in.	AWS administrator

Task	Description	Skills required
	4. For **Connectivity type**, choose **Public**. 5. For **Elastic IP allocation ID**, choose the second Elastic IP address that you created earlier and associate it with the NAT gateway. 6. Choose **Create NAT gateway**.	

Create route tables for your public and private subnets

Task	Description	Skills required
Create the route table for the public-one subnet.	1. On the Amazon VPC console, choose **Route Tables** and then choose **Create route table**. 2. Enter `public-one-subnet` as the route table name and then choose **Create route table**. 3. Choose the `public-one-subnet` route table, choose **Edit routes**, and then choose **Add route**. 4. Specify `0.0.0.0` in the **Destination** box and then choose the internet gateway ID in the **Target** list. 5. On the **Subnet associations** tab, choose **Edit**	AWS administrator

Task	Description	Skills required
	subnet associations, choose the `public-one` subnet with the `10.0.0.0/28` CIDR range, and then choose **Save associations**. 6. Choose **Save Changes**.	

Task	Description	Skills required
Create the route table for the public-two subnet.	1. On the Amazon VPC console, choose **Route Tables** and then choose **Create route table**. 2. Enter `public-two-subnet` as the route table name and then choose **Create route table**. 3. Choose the `public-two-subnet` route table, choose **Edit routes**, and then choose **Add route**. 4. Specify `0.0.0.0` in the **Destination** box and then choose the internet gateway ID in the **Target** list. 5. On the **Subnet associations** tab, choose **Edit subnet associations**, choose the `public-two` subnet with the `10.0.0.16/28` CIDR range, and then choose **Save associations**. 6. Choose **Save Changes**.	AWS administrator

Task	Description	Skills required
Create the route table for the private-one subnet.	1. On the Amazon VPC console, choose **Route Tables** and then choose **Create route table**. 2. Enter `private-one-subnet` as the route table name and then choose **Create route table**. 3. Choose the `private-one-subnet` route table, choose **Edit routes**, and then choose **Add route**. 4. Specify `0.0.0.0` in the **Destination** box and then choose the NAT gateway in the `public-one` subnet in the **Target** list. 5. On the **Subnet associations** tab, choose **Edit subnet associations**, choose the `private-one` subnet with the `10.0.0.32/28` CIDR range, and then choose **Save associations**. 6. Choose **Save Changes**.	AWS administrator

Task	Description	Skills required
Create the route table for the private-two subnet.	1. On the Amazon VPC console, choose **Route Tables** and then choose **Create route table**. 2. Enter `private-two-subnet` as the route table name and then choose **Create route table**. 3. Choose the `private-two-subnet` route table, choose **Edit routes**, and then choose **Add route**. 4. Specify `0.0.0.0` in the **Destination** box and then choose the NAT gateway in the `public-two` subnet in the **Target** list. 5. On the **Subnet associations** tab, choose **Edit subnet associations**, choose the `private-two` subnet with the `10.0.0.64/28` CIDR range, and then choose **Save associations**. 6. Choose **Save Changes**.	AWS administrator

Create the Lambda function, add it to the VPC, and test the solution

Task	Description	Skills required
Create a new Lambda function.	1. Open the AWS Lambda console and choose **Create function**. 2. Under **Basic information**, enter `Lambda test` under **Function name** and then choose the language of your choice under **Runtime**. 3. Choose **Create function**.	AWS administrator
Add the Lambda function to your VPC.	1. On the AWS Lambda console, choose **Functions** and then choose the function that you created earlier. 2. Choose **Configuration** and then choose **VPC**. 3. Choose **Edit** and then choose `Lambda VPC` and both private subnets. 4. Choose **Default security group** for testing purposes and then choose **Save**.	AWS administrator
Write code to call an external service.	1. In the programming language of your choice, write code to call an external service that returns your IP address.	AWS administrator

Task	Description	Skills required
	2. Verify that the returned IP address matches one of your Elastic IP addresses.	

Related resources

- Configuring a Lambda function to access resources in a VPC

Identify duplicate container images automatically when migrating to an Amazon ECR repository

Rishabh Yadav and Rishi Singla, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

The pattern provides an automated solution to identify whether images that are stored in different container repositories are duplicates. This check is useful when you plan to migrate images from other container repositories to Amazon Elastic Container Registry (Amazon ECR).

For foundational information, the pattern also describes the components of a container image, such as the image digest, manifest, and tags. When you plan a migration to Amazon ECR, you might decide to synchronize your container images across container registries by comparing the digests of the images. Before you migrate your container images, you need to check whether these images already exist in the Amazon ECR repository to prevent duplication. However, it can be difficult to detect duplication by comparing image digests, and this might lead to issues in the initial migration phase. This pattern compares the digests of two similar images that are stored in different container registries and explains why the digests vary, to help you compare images accurately.

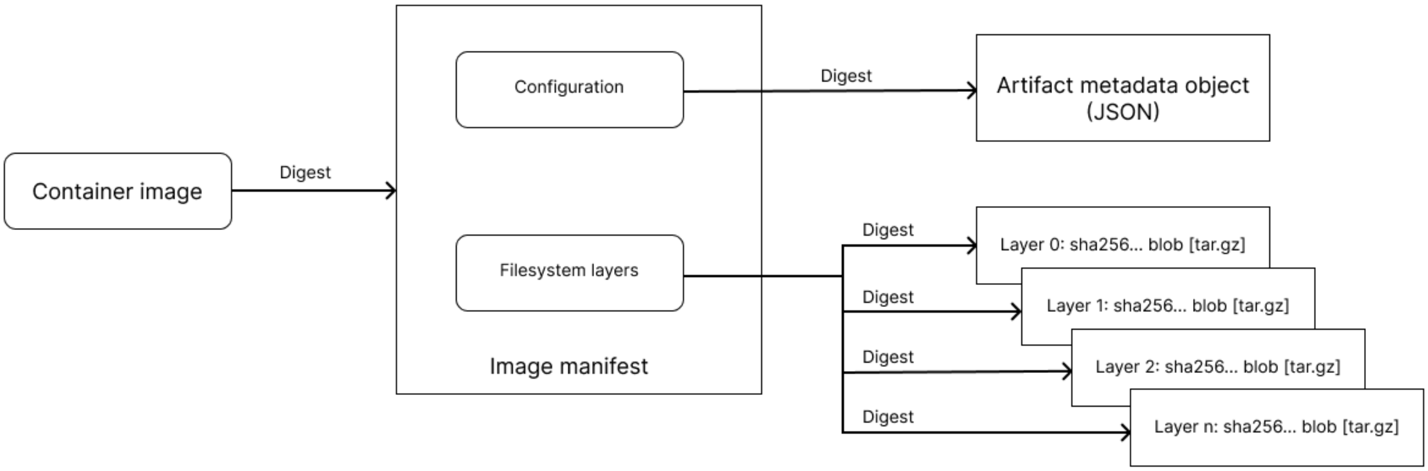
Prerequisites and limitations

- An active AWS account
- Access to the Amazon ECR public registry
- Familiarity with the following AWS services:
  - AWS CodeCommit
  - AWS CodePipeline
  - AWS CodeBuild
  - AWS Identity and Access Management (IAM)
  - Amazon Simple Storage Service (Amazon S3)
- Configured CodeCommit credentials (see instructions)

Architecture

Container image components

The following diagram illustrates some of the components of a container image. These components are described after the diagram.



Terms and definitions

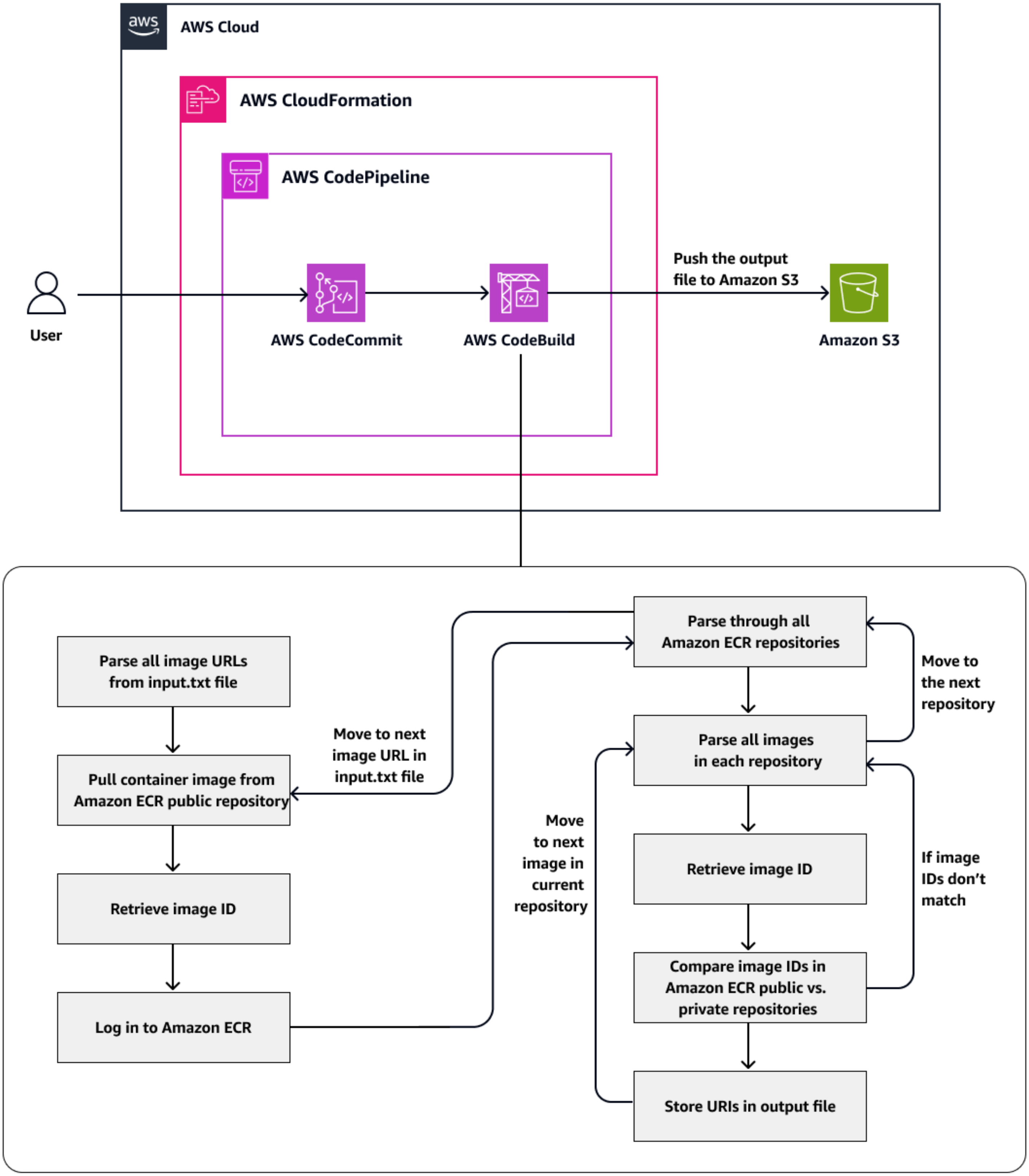
The following terms are defined in the Open Container Initiative (OCI) Image Specification.

- **Registry:** A service for image storage and management.
- **Client:** A tool that communicates with registries and works with local images.
- **Push:** The process for uploading images to a registry.
- **Pull:** The process for downloading images from a registry.
- **Blob:** The binary form of content that is stored by a registry and can be addressed by a digest.
- **Index:** A construct that identifies multiple image manifests for different computer platforms (such as x86-64 or ARM 64-bit) or media types. For more information, see the OCI Image Index Specification.
- **Manifest:** A JSON document that defines an image or artifact that is uploaded through the manifest's endpoint. A manifest can reference other blobs in a repository by using descriptors. For more information, see the OCI Image Manifest Specification.
- **Filesystem layer:** System libraries and other dependencies for an image.
- **Configuration:** A blob that contains artifact metadata and is referenced in the manifest. For more information, see the OCI Image Configuration Specification.

- **Object or artifact:** A conceptual content item that's stored as a blob and associated with an accompanying manifest with a configuration.
- **Digest:** A unique identifier that's created from a cryptographic hash of the contents of a manifest. The image digest helps uniquely identify an immutable container image. When you pull an image by using its digest, you will download the same image every time on any operating system or architecture. For more information, see the OCI Image Specification.
- **Tag:** A human-readable manifest identifier. Compared with image digests, which are immutable, tags are dynamic. A tag that points to an image can change and move from one image to another, although the underlying image digest remains the same.

Target architecture

The following diagram displays the high-level architecture of the solution provided by this pattern to identify duplicate container images by comparing images that are stored in Amazon ECR and private repositories.


AWS Cloud
AWS CloudFormation
AWS CodePipeline
User
AWS CodeCommit
AWS CodeBuild
Push the output file to Amazon S3
Amazon S3
Parse all image URLs from input.txt file
Pull container image from Amazon ECR public repository
Retrieve image ID
Log in to Amazon ECR
Move to next image URL in input.txt file
Parse through all Amazon ECR repositories
Parse all images in each repository
Retrieve image ID
Compare image IDs in Amazon ECR public vs. private repositories
Store URIs in output file
Move to the next repository
If image IDs don't match
Move to next image in current repository

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CodeBuildis a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.

Code

The code for this pattern is available in the GitHub repository Automated solution to identify duplicate container images between repositories.

Best practices

- AWS CloudFormation best practices
- AWS CodePipeline best practices

Epics

Pull container images from Amazon ECR public and private repositories

Task	Description	Skills required
Pull an image from the Amazon ECR public repository.	From the terminal, run the following command to pull the image `amazonlinux` from the Amazon ECR public repository.	App developer, AWS DevOps, AWS administrator

Task	Description	Skills required

	`$~ % docker pull public.ecr.aws/amazonlinux/amazonlinux:2018.03`
	
	When the image has been pulled to your local machine, you'll see the following pull digest, which represents the image index.
	
	`2018.03: Pulling from amazonlinux/amazonlinux`
	`4ddc0f8d367f: Pull complete`
	
	`Digest: sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b5`
	
	`Status: Downloaded newer image for public.ecr.aws/amazonlinux/amazonlinux:2018.03`
	`public.ecr.aws/amazonlinux/amazonlinux:2018.03`	

Task	Description	Skills required

Push the image to an Amazon ECR private repository.	1. Create a private Amazon ECR repository named `test_ecr_repository` in the US East (N. Virginia) Region (`us-east-1`).
	`$~ % aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin <account-id>.dkr.ecr.us-east-1.amazonaws.com`
	`Login Succeeded`
	where `<account-id>` refers to your AWS account.
	2. Tag the local image that you pulled previously. Use the value `public.ecr.aws/amazonlinux/amazonlinux:2018.03` and push it to the Amazon ECR private repository.
	`$~ % docker tag public.ecr.aws/amazonlinux/amazonlinux:2018.03 <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
		AWS administrator, AWS DevOps, App developer

Task	Description	Skills required

	`$~ % docker push <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	
	When you push the image to the Amazon ECR repository, Docker will push the underlying image and not the image index.
	
	`The push refers to repository [<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository] d5655967c2c4: Pushed latest: digest: sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02 size: 529`	

Task	Description	Skills required

Pull the same image from the Amazon ECR private repositor y.	1. From the terminal, run the following command to pull the image that you previously pushed to the Amazon ECR private repository.
	<pre>$~ % docker pull
	 <account-id>.dkr.e
	cr.us-east-1.amazo
	naws.com/test_ecr_
	repository:latest
	latest: Pulling from
	 test_ecr_repository
	Digest: sha256:52
	db9000073d93b9bdee
	6a7246a68c35a741aa
	ade05a8f4febba0bf7
	95cdac02
	Status: Image is up
	 to date for <account-
	id>.dkr.ecr.us-eas
	t-1.amazonaws.com/
	test_ecr_repositor
	y:latest
	<account-id>.dkr.e
	cr.us-east-1.amazo
	naws.com/test_ecr_
	repository:latest</pre>The digest for this image matches the digest of the image that you pushed to the Amazon ECR private repository, and represent s the underlying image. This value doesn't match the image index that you
		App developer, AWS DevOps, AWS administrator

Task	Description	Skills required
	pulled from the public repository. 2. To verify, retrieve the image index by digest. `curl -k -H "Authorization: Bearer $TOKEN" https://public.ecr.aws/v2/amazonlinux/amazonlinux/manifests/sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b55 { "schemaVersion": 2, "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json", "manifests": [{ "mediaType": "application/vnd.docker.distribution.manifest.v2+json", "size": 529, "digest": "sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02", "platform": { "architecture": "amd64",`	

Task	Description	Skills required
	```"os": "linux" } }] }```	

Compare the image manifests

Task	Description	Skills required
Find the manifest of the image stored in the Amazon ECR public repository.	From the terminal, run the following command to pull the manifest of the image `public.ecr.aws/amazonlinux/amazonlinux:2018.03` from the Amazon ECR public repository. ```$~ % docker manifest inspect public.ecr.aws/amazonlinux/amazonlinux:2018.03 { "schemaVersion": 2, "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json", "manifests": [{ "mediaType": "application/vnd.docker.distribution.manifest.v2+json", "size": 529,```	AWS administrator, AWS DevOps, App developer

Task	Description	Skills required
	"digest": "sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02", "platform": { "architecture": "amd64", "os": "linux" } }] }	

Task	Description	Skills required

Find the manifest of the image stored in the Amazon ECR private repository.	From the terminal, run the following command to pull the manifest of the image `<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest` from the Amazon ECR private repository.
	`$~ % docker manifest inspect <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	
	`{ "schemaVersion": 2, "mediaType": "application/vnd.docker.distribution.manifest.v2+json", "config": { "mediaType": "application/vnd.docker.container.image.v1+json", "size": 1477, "digest": "sha256:f7cee5e1af28ad4e147589c474d399b12d9b551ef4c3e11e02d982fce5eebc68" }, "layers": [ {`
		AWS DevOps, AWS systems administrator, App developer

Task	Description	Skills required
	"mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip", "size": 62267075, "digest": "sha256:4ddc0f8d367f424871a060e2067749f32bd36a91085e714dcb159952f2d71453" }] }	

Task	Description	Skills required
Compare the digest pulled by Docker with the manifest digest for the image in the Amazon ECR private repository.	Another question is why the digest provided by the **docker pull** command differs from the manifest's digest for the image `<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest` . The digest used for **docker pull** represents the digest of the image manifest, which is stored in a registry. This digest is considered the root of a hash chain, because the manifest contains the hash of the content that will be downloaded and imported into Docker. The image ID used within Docker can be found in this manifest as `config.digest` . This represents the image configuration that Docker uses. So you could say that the manifest is the envelope, and the image is the content of the envelope. The manifest digest is always different from the image ID. However, a specific manifest should always produce the same image ID. Because the	AWS DevOps, AWS systems administrator, App developer

Task	Description	Skills required
	manifest digest is a hash chain, we cannot guarantee that it will always be the same for a given image ID. In most cases, it produces the same digest, although Docker cannot guarantee that. The possible difference in the manifest digest stems from Docker not storing the blobs that are compressed with gzip locally. Therefore , exporting layers might produce a different digest, although the uncompressed content remains the same. The image ID verifies that uncompressed content is the same; that is, the image ID is now a content addressable identifier (`chainID`). To confirm this informati on, you can compare the output of the **docker inspect** command on the Amazon ECR public and private repositor ies: 1. Run the following command from your terminal for the image stored in the Amazon ECR public repository.	

Task	Description	Skills required

	`$~ % docker inspect public.ecr.aws/amazonlinux/amazonlinux:2018.03`
	
	For the output from the command, see the [Additional information](#) section.
	
	2. Run the following command from your terminal for the image stored in the Amazon ECR private repository.
	
	`$~ % docker inspect <account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest`
	
	For the output from the command, see the [Additional information](#) section.
	
	The results verify that both images have the same image ID digest and layer digest.
	
	ID: `f7cee5e1af28ad4e147589c474d399b12d9b551ef4c3e11e02d982fce5eebc68`	

Task	Description	Skills required
	Layers: d5655967c `2c4e8d68f8ec7cf753` `218938669e6c16ac13` `24303c073c736a2e2a2` Additionally, the digests are based on the bytes of the object that's managed locally (the local file is a tar of the container image layer) or the blob that's pushed to the registry server. However, when you push the blob to a registry, the tar is compresse d and the digest is computed in the compressed tar file. Therefore, the difference in the **docker pull** digest value arises from compressi on that is applied at the registry (Amazon ECR private or public) level. **Note** This explanation is specific to using a Docker client. You won’t see this behavior with other clients such as **nerdctl** or **Finch**, because they don’t automatically compress the image	

Task	Description	Skills required
	during push and pull operations.	

Automatically identify duplicate images between Amazon ECR public and private repositories

Task	Description	Skills required
Clone the repository.	Clone the Github repository for this pattern into a local folder: `$git clone https://github.com/aws-samples/automated-solution-to-identify-duplicate-container-images-between-repositories`	AWS administrator, AWS DevOps
Set up a CI/CD pipeline.	The GitHub repository includes a `.yaml` file that creates an AWS CloudFormation stack to set up a pipeline in AWS CodePipeline. 1. Sign in to the AWS Management Console and open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/). 2. Create a stack by using the template `pipeline.yaml` file, which is in the code folder in the cloned repository.	AWS administrator, AWS DevOps

Task	Description	Skills required
	3. Accept or change the default values of parameters. Specify values for the following: • Stack name • `ArtifactStoreBucketName` – An existing S3 bucket that will be used to store AWS CodePipeline artifacts • `OutputBucket` – An existing S3 bucket that will be used to store the URIs for the duplicate images • `SourceImageFile` – An existing text file named `input.txt` that contains the image URIs from the public repository that will be checked against the Amazon ECR private repository to detect duplication 4. Review and adjust stack options, and then choose **Submit** to run the template. The pipeline will be set up with two stages (CodeCommit and CodeBuild, as shown in	

Task	Description	Skills required
	the architecture diagram) to identify images in the private repository that also exist in the public repository. The pipeline is configured with the following resources: • CodePipeline for the orchestration of the deployment pipeline. • A CodeCommit repository to store the bash script and input file. The bash script is used to compare the container image IDs in the public and private repositories to find duplications. This check is performed across all the repositories in the specified AWS account in a single AWS Region. • A CodeBuild project to invoke the bash script to identify images that are already present in the Amazon ECR repository. • Necessary IAM roles to allow access. • An S3 bucket to store the output file that contains image URIs. • Another S3 bucket to store CodePipeline artifacts.	

Task	Description	Skills required
Populate the CodeCommit repository.	To populate the CodeCommi t repository, perform these steps: 1. Open the [CodeCommit console](#) and navigate to the AWS Region where you created the CloudForm ation stack. 2. Find the repository that you provisioned by using the CloudFormation script from the list, choose **Clone URL,** and then copy the **HTTPS URL** protocol to connect to the repository. 3. Open a command prompt and run the **git clone** command with the HTTPS URL that you copied in the previous step. 4. Navigate to the root directory. Create a file named `input.txt` and populate this file with the Amazon ECR public image registry URIs that you would like to search for in the private Amazon ECR repository. 5. Copy the files `script.sh` , `buildspec.yml` , and `input.txt` from your local copy of the GitHub	AWS administrator, AWS DevOps

Task	Description	Skills required

	repository [Automated solution to identify duplicate container images between repositories](#) to the cloned CodeCommit repository.
	6. Upload the files to CodeCommit by using these commands:
	`git add .`
	`git commit -m "added input files"`
	`git push`	

Clean up.	To avoid incurring future charges, delete the resources by following these steps: 1. Navigate to the S3 bucket that stores the CodePipeline artifacts, and empty the bucket. 2. Navigate to the S3 bucket that stores the duplicate image URIs, and empty the bucket. 3. Navigate to the CloudFormation console and delete the stack that you created to set up the pipeline.	AWS administrator

Troubleshooting

Issue	Solution
When you try to push, pull, or otherwise interact with a CodeCommit repository from the terminal or command line, you are prompted to provide a user name and password, and you must supply the Git credentials for your IAM user.	The most common causes for this error are the following: • Your local computer is running an operating system that doesn't support credential management, or it doesn't have a credential management utility installed. • The Git credentials for your IAM user haven't been saved to one of these credential management systems. Depending on your operating system and local environment, you might need to install a credential manager, configure the credentia l manager that is included in your operating system, or customize your local environment to use credential storage. For example, if your computer is running macOS, you can use the Keychain Access utility to store your credentia ls. If your computer is running Windows, you can use the Git Credential Manager that is installed with Git for Windows. For more information, see Setup for HTTPS users using Git credentials in the CodeCommit documentation and Credential Storage in the Git documentation.
You encounter HTTP 403 or "no basic auth credentials" errors when you push an image to the Amazon ECR repository.	You might encounter these error messages from the **docker push** or **docker pull** command, even if you have successfully authenticated to Docker by using the **aws ecr get-login-password** command. Known causes are:

Issue	Solution
	• You have authenticated to a different Region. For more information, see Private registry authentication in the Amazon ECR documentation. • You have authenticated to push to a repository that you don't have permissio ns for. For more information, see Private repository policies in the Amazon ECR documentation. • Your token has expired. The default expiration period for tokens obtained by using the `GetAuthorizationToken` operation is 12 hours.

Related resources

- Automated solution to identify duplicate container images between repositories (GitHub repository)
- Amazon ECR public gallery
- Private images in Amazon ECR (Amazon ECR documentation)
- AWS::CodePipeline::Pipeline resource (AWS CloudFormation documentation)
- OCI Image Format Specification

Additional information

Output of Docker inspection for image in Amazon ECR public repository

```
[
    {
        "Id":
 "sha256:f7cee5e1af28ad4e147589c474d399b12d9b551ef4c3e11e02d982fce5eebc68",
        "RepoTags": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest",
            "public.ecr.aws/amazonlinux/amazonlinux:2018.03"
```

```
        ],
        "RepoDigests": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/
test_ecr_repository@sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02",
            "public.ecr.aws/amazonlinux/
amazonlinux@sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b5"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2023-02-23T06:20:11.575053226Z",
        "Container":
 "ec7f2fc7d2b6a382384061247ef603e7d647d65f5cd4fa397a3ccbba9278367c",
        "ContainerConfig": {
            "Hostname": "ec7f2fc7d2b6",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/sh",
                "-c",
                "#(nop) ",
                "CMD [\"/bin/bash\"]"
            ],
            "Image":
 "sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": {}
        },
        "DockerVersion": "20.10.17",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
```

```
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/bash"
            ],
            "Image":
 "sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 167436755,
        "VirtualSize": 167436755,
        "GraphDriver": {
            "Data": {
                "MergedDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/merged",
                "UpperDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/diff",
                "WorkDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/work"
            },
            "Name": "overlay2"
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [

 "sha256:d5655967c2c4e8d68f8ec7cf753218938669e6c16ac1324303c073c736a2e2a2"
            ]
        },
        "Metadata": {
```

```
            "LastTagTime": "2023-03-02T10:28:47.142155987Z"
        }
    }
]
```

Output of Docker inspection for image in Amazon ECR private repository

```
[
    {
        "Id":
 "sha256:f7cee5e1af28ad4e147589c474d399b12d9b551ef4c3e11e02d982fce5eebc68",
        "RepoTags": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/test_ecr_repository:latest",
            "public.ecr.aws/amazonlinux/amazonlinux:2018.03"
        ],
        "RepoDigests": [
            "<account-id>.dkr.ecr.us-east-1.amazonaws.com/
test_ecr_repository@sha256:52db9000073d93b9bdee6a7246a68c35a741aaade05a8f4febba0bf795cdac02",
            "public.ecr.aws/amazonlinux/
amazonlinux@sha256:f972d24199508c52de7ad37a298bda35d8a1bd7df158149b381c03f6c6e363b5"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2023-02-23T06:20:11.575053226Z",
        "Container":
 "ec7f2fc7d2b6a382384061247ef603e7d647d65f5cd4fa397a3ccbba9278367c",
        "ContainerConfig": {
            "Hostname": "ec7f2fc7d2b6",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/sh",
                "-c",
                "#(nop) ",
```

```
                "CMD [\"/bin/bash\"]"
            ],
            "Image":
 "sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": {}
        },
        "DockerVersion": "20.10.17",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/bash"
            ],
            "Image":
 "sha256:c1bced1b5a65681e1e0e52d0a6ad17aaf76606149492ca0bf519a466ecb21e51",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "Architecture": "amd64",
        "Os": "linux",
        "Size": 167436755,
        "VirtualSize": 167436755,
        "GraphDriver": {
            "Data": {
                "MergedDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/merged",
```

```
                "UpperDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/diff",
                "WorkDir": "/var/lib/docker/overlay2/
c2c2351a82b26cbdf7782507500e5adb5c2b3a2875bdbba79788a4b27cd6a913/work"
            },
            "Name": "overlay2"
        },
        "RootFS": {
            "Type": "layers",
            "Layers": [

 "sha256:d5655967c2c4e8d68f8ec7cf753218938669e6c16ac1324303c073c736a2e2a2"
            ]
        },
        "Metadata": {
            "LastTagTime": "2023-03-02T10:28:47.142155987Z"
        }
    }
]
```

Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet

Mahendra Revanasiddappa, Amazon Web Services

Summary

Note, September 2021: The latest Amazon EKS optimized AMIs install SSM Agent automatically. For more information, see the release notes for the June 2021 AMIs.

In Amazon Elastic Kubernetes Service (Amazon EKS), because of security guidelines, worker nodes don't have Secure Shell (SSH) key pairs attached to them. This pattern shows how you can use the Kubernetes DaemonSet resource type to install AWS Systems Manager Agent (SSM Agent) on all worker nodes, instead of installing it manually or replacing the Amazon Machine Image (AMI) for the nodes. DaemonSet uses a cron job on the worker node to schedule the installation of SSM Agent. You can also use this pattern to install other packages on worker nodes.

When you're troubleshooting issues in the cluster, installing SSM Agent on demand enables you to establish an SSH session with the worker node, to collect logs or to look into instance configuration, without SSH key pairs.

Prerequisites and limitations

Prerequisites

- An existing Amazon EKS cluster with Amazon Elastic Compute Cloud (Amazon EC2) worker nodes.
- Container instances should have the required permissions to communicate with the SSM service. The AWS Identity and Access Management (IAM) managed role **AmazonSSMManagedInstanceCore** provides the required permissions for SSM Agent to run on EC2 instances. For more information, see the AWS Systems Manager documentation.

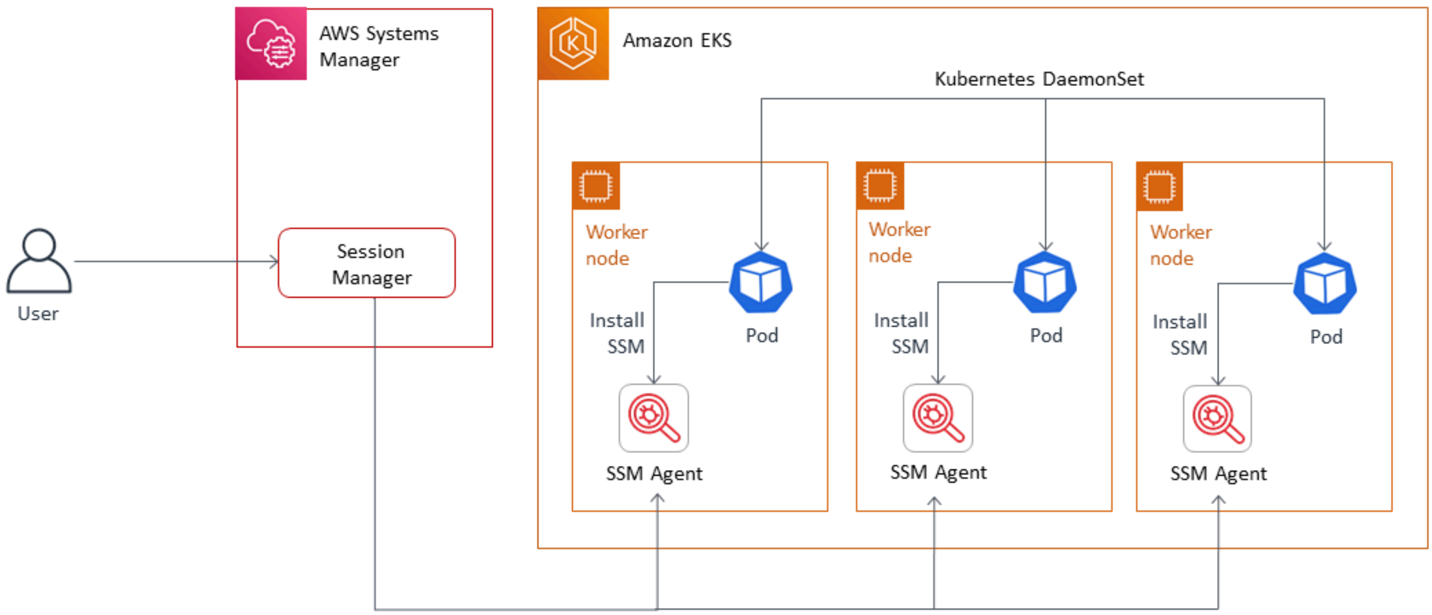
Limitations

- This pattern isn't applicable to AWS Fargate, because DaemonSets aren't supported on the Fargate platform.
- This pattern applies only to Linux-based worker nodes.

- The DaemonSet pods run in privileged mode. If the Amazon EKS cluster has a webhook that blocks pods in privileged mode, the SSM Agent will not be installed.

Architecture

The following diagram illustrates the architecture for this pattern.



Tools

Tools

- kubectl is a command-line utility that is used to interact with an Amazon EKS cluster. This pattern uses `kubectl` to deploy a DaemonSet on the Amazon EKS cluster, which will install SSM Agent on all worker nodes.
- Amazon EKS makes it easy for you to run Kubernetes on AWS without having to install, operate, and maintain your own Kubernetes control plane or nodes. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.
- AWS Systems Manager Session Manager lets you manage your EC2 instances, on-premises instances, and virtual machines (VMs) through an interactive, one-click, browser-based shell or through the AWS Command Line Interface (AWS CLI).

Code

Use the following code to create a DaemonSet configuration file that will install SSM Agent on the Amazon EKS cluster. Follow the instructions in the Epics section.

```
cat << EOF > ssm_daemonset.yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  labels:
    k8s-app: ssm-installer
  name: ssm-installer
  namespace: kube-system
spec:
  selector:
    matchLabels:
      k8s-app: ssm-installer
  template:
    metadata:
      labels:
        k8s-app: ssm-installer
    spec:
      containers:
      - name: sleeper
        image: busybox
        command: ['sh', '-c', 'echo I keep things running! && sleep 3600']
      initContainers:
      - image: amazonlinux
        imagePullPolicy: Always
        name: ssm
        command: ["/bin/bash"]
        args: ["-c","echo '* * * * * root yum install -y https://s3.amazonaws.com/
ec2-downloads-windows/SSMAgent/latest/linux_amd64/amazon-ssm-agent.rpm & rm -rf /etc/
cron.d/ssmstart' > /etc/cron.d/ssmstart"]
        securityContext:
          allowPrivilegeEscalation: true
        volumeMounts:
        - mountPath: /etc/cron.d
          name: cronfile
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      volumes:
      - name: cronfile
        hostPath:
          path: /etc/cron.d
```

```
          type: Directory
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      terminationGracePeriodSeconds: 30
EOF
```

Epics

Set up kubectl

Task	Description	Skills required
Install and configure kubectl to access the EKS cluster.	If `kubectl` isn't already installed and configured to access the Amazon EKS cluster, see [Installing kubectl](#) in the Amazon EKS documentation.	DevOps

Deploy the DaemonSet

Task	Description	Skills required
Create the DaemonSet configuration file.	Use the code in the [Code](#) section earlier in this pattern to create a DaemonSet configuration file called `ssm_daemonset.yaml` , which will be deployed to the Amazon EKS cluster. The pod launched by DaemonSet has a main container and an `init` container. The main container has a `sleep` command. The	DevOps

Task	Description	Skills required
	`init` container includes a `command` section that creates a cron job file to install SSM Agent at the path `/etc/cron.d/` . The cron job runs only once, and the file it creates is automatically deleted after the job is complete. When the init container has finished, the main container waits for 60 minutes before exiting. After 60 minutes, a new pod is launched. This pod installs SSM Agent, if it's missing, or updates SSM Agent to the latest version. If required, you can modify the `sleep` command to restart the pod once a day or to run more often.	
Deploy the DaemonSet on the Amazon EKS cluster.	To deploy the DaemonSet configuration file you created in the previous step on the Amazon EKS cluster, use the following command: `kubectl apply -f ssm_daemonset.yaml` This command creates a DaemonSet to run the pods on worker nodes to install SSM Agent.	DevOps

Related resources

- Installing kubectl (Amazon EKS documentation)
- Setting up Session Manager (AWS Systems Manager documentation)

Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands

Akkamahadevi Hiremath, Amazon Web Services

Summary

This pattern provides code samples and steps to install the AWS Systems Manager Agent (SSM Agent) and Amazon CloudWatch agent on Amazon Elastic Kubernetes Service (Amazon EKS) worker nodes in the Amazon Web Services (AWS) Cloud during Amazon EKS cluster creation. You can install the SSM Agent and CloudWatch agent by using the `preBootstrapCommands` property from the `eksctl` config file schema (Weaveworks documentation). Then, you can use the SSM Agent to connect to your worker nodes without using an Amazon Elastic Compute Cloud (Amazon EC2) key pair. Additionally, you can use the CloudWatch agent to monitor memory and disk utilization on your Amazon EKS worker nodes.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The eksctl command line utility, installed and configured on macOS, Linux, or Windows
- The kubectl command line utility, installed and configured on macOS, Linux, or Windows

Limitations

- We recommend that you avoid adding long-running scripts to the `preBootstrapCommands` property, because this delays the node from joining the Amazon EKS cluster during scaling activities. We recommend that you create a custom Amazon Machine Image (AMI) instead.
- This pattern applies to Amazon EC2 Linux instances only.

Architecture

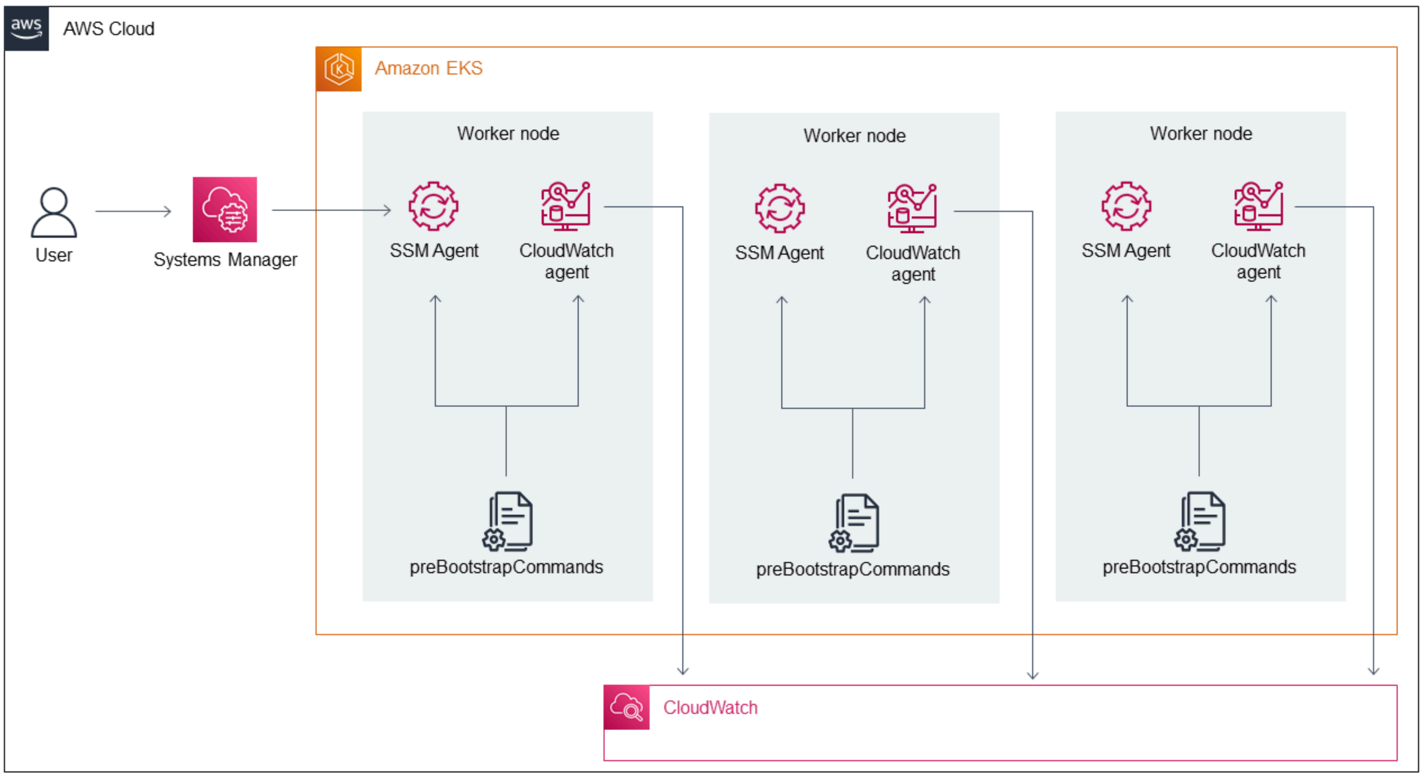
Technology stack

- Amazon CloudWatch

- Amazon Elastic Kubernetes Service (Amazon EKS)
- AWS Systems Manager Parameter Store

Target architecture

The following diagram shows an example of a user connecting to Amazon EKS worker nodes using SSM Agent which was installed using the `preBootstrapCommands`.



The diagram shows the following workflow:

1. The user creates an Amazon EKS cluster by using the `eksctl` configuration file with the `preBootstrapCommands` property, which installs the SSM Agent and CloudWatch agent.
2. Any new instances that join the cluster later due to scaling activities get created with the pre-installed SSM Agent and CloudWatch agent.
3. The user connects to Amazon EC2 by using the SSM Agent and then monitors memory and disk utilization by using the CloudWatch agent.

Tools

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications that you run on AWS in real time.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management.
- AWS Systems Manager Session Manager helps you manage your EC2 instances, on-premises instances, and virtual machines through an interactive, one-click, browser-based shell or through the AWS Command Line Interface (AWS CLI).
- eksctl is a command-line utility for creating and managing Kubernetes clusters on Amazon EKS.
- kubectl is a command-line utility for communicating with the cluster API server.

Epics

Create an Amazon EKS cluster

Task	Description	Skills required
Store the CloudWatch agent configuration file.	Store the CloudWatch agent configuration file in the AWS Systems Manager Parameter Store in the AWS Region where you want to create your Amazon EKS cluster. To do this, create a parameter in AWS Systems Manager Parameter Store and note the name of the parameter (for example, `AmazonCloudwatch-linux`). For more information, see the *Example CloudWatch agent configuration file* code	DevOps engineer

Task	Description	Skills required
	in the [Additional information](#) section of this pattern.	
Create the eksctl configura tion file and cluster.	1. Create an `eksctl` configuration file that includes the CloudWatc h agent and SSM Agent installation steps. For more information, see the *Example eksctl configura tion file* code in the [Additional information](#) section of this pattern. 2. Create a cluster by running the `eksctl create cluster -f cluster.y aml` command.	AWS DevOps

Verify that the SSM Agent and CloudWatch agent work

Task	Description	Skills required
Test the SSM Agent.	Use SSH to connect to your Amazon EKS cluster nodes by using any of the methods covered in [Start a session](#) from the AWS Systems Manager documentation.	AWS DevOps
Test the CloudWatch agent.	Use the CloudWatch console to validate the CloudWatch agent: 1. Sign in to the AWS Management Console	AWS DevOps

Task	Description	Skills required
	and open the CloudWatch console. 2. On the navigation pane, expand **Metrics** and then choose **All metrics**. 3. In the search box on the **Browse** tab, enter and then choose **CWAgent metrics** to see the memory and disk metrics.	

Related resources

- Installing and running the CloudWatch agent on your servers (Amazon CloudWatch documentation)
- Create a Systems Manager parameter (console) (AWS Systems Manager documentation)
- Create the CloudWatch agent configuration file (Amazon CloudWatch documentation)
- Starting a session (AWS CLI) (AWS Systems Manager documentation)
- Starting a session (Amazon EC2 console) (AWS Systems Manager documentation)

Additional information

Example CloudWatch agent configuration file

In the following example, the CloudWatch agent is configured to monitor disk and memory utilization on Amazon Linux instances:

```
{
    "agent": {
        "metrics_collection_interval": 60,
        "run_as_user": "cwagent"
    },
    "metrics": {
        "append_dimensions": {
            "AutoScalingGroupName": "${aws:AutoScalingGroupName}",
```

```
            "ImageId": "${aws:ImageId}",
            "InstanceId": "${aws:InstanceId}",
            "InstanceType": "${aws:InstanceType}"
        },
        "metrics_collected": {
            "disk": {
                "measurement": [
                    "used_percent"
                ],
                "metrics_collection_interval": 60,
                "resources": [
                    "*"
                ]
            },
            "mem": {
                "measurement": [
                    "mem_used_percent"
                ],
                "metrics_collection_interval": 60
            }
        }
    }
}
```

Example eksctl configuration file

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: test
  region: us-east-2
  version: "1.24"
managedNodeGroups:
  - name: test
    minSize: 2
    maxSize: 4
    desiredCapacity: 2
    volumeSize: 20
    instanceType: t3.medium
    preBootstrapCommands:
    - sudo yum install amazon-ssm-agent -y
    - sudo systemctl enable amazon-ssm-agent
    - sudo systemctl start amazon-ssm-agent
```

```
    - sudo yum install amazon-cloudwatch-agent -y
    - sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-
config -m ec2 -s -c ssm:AmazonCloudwatch-linux
    iam:
      attachPolicyARNs:
        - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
        - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
        - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
        - arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
```

Additional code details

- In the last line of the `preBootstrapCommands` property, `AmazonCloudwatch-linux` is the name of the parameter created in AWS System Manager Parameter Store. You must include `AmazonCloudwatch-linux` in Parameter Store in the same AWS Region where you created the Amazon EKS cluster. You can also specify a file path, but we recommend using Systems Manager for easier automation and reusability.
- If you use `preBootstrapCommands` in the `eksctl` configuration file, you see two launch templates in the AWS Management Console. The first launch template includes the commands specified in `preBootstrapCommands`. The second template includes the commands specified in `preBootstrapCommands` and default Amazon EKS user data. This data is required to get the nodes to join the cluster. The node group's Auto Scaling group uses this user data to spin up new instances.
- If you use the `iam` attribute in the `eksctl` configuration file, you must list the default Amazon EKS policies with any additional policies required in your attached AWS Identity and Access Management (IAM) policies. In the code snippet from the *Create the eksctl configuration file and cluster* step, `CloudWatchAgentServerPolicy` and `AmazonSSMMangedInstanceCore` are additional policies added to make sure that the CloudWatch agent and SSM Agent work as expected. The `AmazonEKSWorkerNodePolicy`, `AmazonEKS_CNI_Policy`, `AmazonEC2ContainerRegistryReadOnly` policies are mandatory policies required for the Amazon EKS cluster to function correctly.

Migrate NGINX Ingress Controllers when enabling Amazon EKS Auto Mode

Olawale Olaleye and Shamanth Devagari, Amazon Web Services

Summary

EKS Auto Mode for Amazon Elastic Kubernetes Service (Amazon EKS) can reduce the operational overhead of running your workloads on Kubernetes clusters. This mode allows AWS to also set up and manage the infrastructure on your behalf. When you enable EKS Auto Mode on an existing cluster, you must carefully plan the migration of NGINX Ingress Controller configurations. This is because the direct transfer of Network Load Balancers isn't possible.

You can use a blue/green deployment strategy to migrate an NGINX Ingress Controller instance when you enable EKS Auto Mode in an existing Amazon EKS cluster.

Prerequisites and limitations

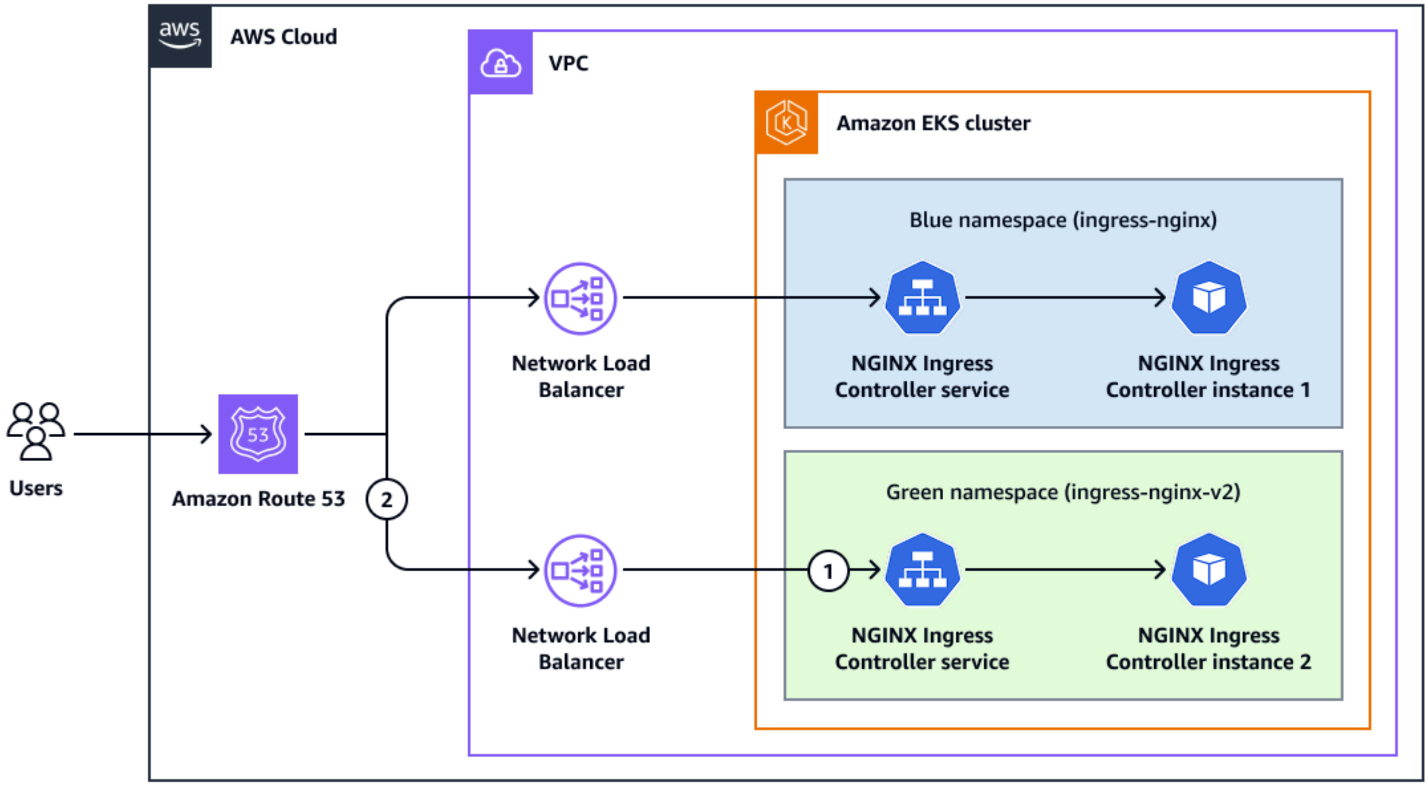
Prerequisites

- An active AWS account
- An Amazon EKS cluster running Kubernetes version 1.29 or later
- Amazon EKS add-ons running minimum versions
- Latest version of kubectl
- An existing NGINX Ingress Controller instance
- (Optional) A hosted zone in Amazon Route 53 for DNS-based traffic shifting

Architecture

A *blue/green deployment* is a deployment strategy in which you create two separate but identical environments. Blue/green deployments provide near-zero downtime release and rollback capabilities. The fundamental idea is to shift traffic between two identical environments that are running different versions of your application.

The following image shows the migration of Network Load Balancers from two different NGINX Ingress Controller instances when enabling EKS Auto Mode. You use a blue/green deployment to shift traffic between the two Network Load Balancers.



The original namespace is the *blue* namespace. This is where the original NGINX Ingress Controller service and instance run, before you enable EKS Auto Mode. The original service and instance connect to a Network Load Balancer that has a DNS name that is configured in Route 53. The [AWS Load Balancer Controller] deployed this Network Load Balancer in the target virtual private cloud (VPC).

The diagram shows the following workflow to set up an environment for a blue/green deployment:

1. Install and configure another NGINX Ingress Controller instance in a different namespace, a *green* namespace.
2. In Route 53, configure a DNS name for a new Network Load Balancer.

Tools

AWS services

- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.

- Elastic Load Balancing distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones.
- Amazon Route 53 is a highly available and scalable DNS web service.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- Helm is an open source package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.
- kubectl is a command-line interface that helps you run commands against Kubernetes clusters.
- NGINX Ingress Controller connects Kubernetes apps and services with request handling, auth, self-service custom resources, and debugging.

Epics

Review the existing environment

Task	Description	Skills required
Confirm that the original NGINX Ingress Controller instance is operational.	Enter the following command to verify that the resources in the `ingress-nginx` namespace are operational. If you have deployed NGINX Ingress Controller in another namespace, update the namespace name in this command. `kubectl get all -n ingress-nginx`	DevOps engineer

Task	Description	Skills required

	In the output, confirm that NGINX Ingress Controller pods are in running state. The following is an example output:
	``
	NAME
	
	   READY STATUS
	 RESTARTS AGE
	pod/ingress-nginx-
	admission-create-xqn9d
	       0/1 Completed
	   0 88m
	pod/ingress-nginx-
	admission-patch-lhk4j
	       0/1 Completed
	   1 88m
	pod/ingress-nginx-
	controller-68f68f859-
	xrz74 1/1 Running
	     2 (10m ago) 72m
	
	NAME
	
	 TYPE CLUSTER-
	IP EXTERNAL-IP
	
	 PORT(S)
	      AGE
	service/ingress-nginx-
	controller
	     LoadBalancer
	 10.100.67.255 k8s-
	ingressn-ingressn-
	abcdefg-12345.elb.eu-
	west-1.amazonaws.com
	   80:30330/TCP,443:3
	1462/TCP 88m
	``	

Task	Description	Skills required

	``
	service/ingress-
	nginx-controller-
	admission ClusterIP
	      10.100.201.176
	 <none>
	
	         443/TCP
	              88m
	
	NAME
	
	 READY UP-TO-DATE
	 AVAILABLE AGE
	deployment.apps/
	ingress-nginx-co
	ntroller 1/1 1
	         1 88m
	
	NAME
	
	     DESIRED CURRENT
	 READY AGE
	replicaset.apps/
	ingress-nginx-co
	ntroller-68f68f859 1
	         1 1
	   72m
	replicaset.apps/
	ingress-nginx-co
	ntroller-d8c96cf68 0
	         0 0
	   88m
	
	NAME
	
	 STATUS COMPLETIONS
	   DURATION AGE
	job.batch/ingress-
	nginx-admission-create
	   Complete 1/1
	    4s 88m
	``	

Task	Description	Skills required

	``
	job.batch/ingress-
	nginx-admission-patch
	   Complete 1/1
	    5s 88m
	``	

Deploy a sample HTTPd workload to use the NGINX Ingress Controller

Task	Description	Skills required

Create the Kubernetes resources.	Enter the following commands to create a sample Kubernetes deployment, service, and ingress:
	`kubectl create deployment demo --image=httpd --port=80`
	`kubectl expose deployment demo`
	`kubectl create ingress demo --class=nginx \ --rule nginxautomode.local.dev/=demo:80`
		DevOps engineer

Review the deployed resources.	Enter the following command to view a list of the deployed resources: `kubectl get all,ingress` In the output, confirm that the sample HTTPd pod is in a	DevOps engineer

Task	Description	Skills required

	running state. The following is an example output:
	``
	NAME
	      READY STATUS
	 RESTARTS AGE
	pod/demo-7d94f8cb4f-
	q68wc 1/1 Running
	   0 59m
	
	NAME
	 TYPE CLUSTER-I
	P EXTERNAL-IP
	 PORT(S) AGE
	service/demo
	 ClusterIP 10.100.78
	.155 <none>
	 80/TCP 59m
	service/kubernetes
	 ClusterIP 10.100.0.1
	      <none> 443/
	TCP 117m
	
	NAME
	  READY UP-TO-DATE
	 AVAILABLE AGE
	deployment.apps/demo
	 1/1 1 1
	          59m
	
	NAME
	             DESIRED
	 CURRENT READY AGE
	replicaset.apps/
	demo-7d94f8cb4f 1
	      1 1
	 59m
	
	NAME
	         CLASS HOSTS
	``	

Task	Description	Skills required
	``` ADDRESS PORTS AGE ingress.networking.k8s.io/demo nginx nginxautomode.local.dev k8s-ingressn-ingressn-abcdefg-12345.elb.eu-west-1.amazonaws.com 80 56m ```	

Confirm the service is reachable.	Enter the following command to confirm that the service is reachable through the DNS name of the Network Load Balancer:
	`curl -H "Host: nginxautomode.local.dev" http://k8s-ingressn-ingressn-abcdefg-12345.elb.eu-west-1.amazonaws.com`
	The following is the expected output:
	`<html><body><h1>It works!</h1></body></html>`
		DevOps engineer

Task	Description	Skills required

(Optional) Create a DNS record.	1. Follow the instructions in [Creating records by using the Amazon Route 53 console](#) (Route 53 documentation) to create a DNS record for the configured domain.
	2. Enter the following command to confirm that the service is reachable through the configured domain name:
	`curl "http://nginxautomode.local.dev/?[1-5]"`
	The following is the expected output:
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
	`<html><body><h1>It works!</h1></body></html>`
		DevOps engineer, AWS DevOps

Enable EKS Auto Mode on the existing cluster

Task	Description	Skills required
Enable EKS Auto Mode.	Follow the instructions in [Enable EKS Auto Mode on an existing cluster](#) (Amazon EKS documentation).	AWS DevOps

Install a new NGINX Ingress Controller

Task	Description	Skills required

Configure a new NGINX Ingress Controller instance.	1. Download the [deploy.yaml](#) template.
	2. Open the **deploy.yaml** template in your preferred editor.
	3. In the `kind: Namespace` section, enter a unique name for the namespace, such as `ingress-nginx-v2` :
	<pre>apiVersion: v1
	kind: Namespace
	metadata:
	  labels:
	    app.kubernetes.io/instance: ingress-nginx
	    app.kubernetes.io/name: ingress-nginx
	  name: ingress-nginx-v2</pre>
		DevOps engineer

Task	Description	Skills required

	4. For each section, update the `namespace` value to the new name.
	5. In the `kind: Deployment` section, do the following:
	a. Enter a unique value for `--controller-class`, such as `k8s.io/ingress-nginx-v2`.
	b. Enter a unique value for `--ingress-class`, such as `nginx-v2`.
	<pre>apiVersion: apps/v1
	kind: Deployment
	  name: ingress-nginx-controller
	  namespace: ingress-nginx-v2
	...
	    spec:
	      containers:
	      - args:
	        - /nginx-ingress-controller
	        - --publish-service=$(POD_NAMESPACE)/ingress-nginx-controller
	        - --election-id=ingress-nginx-leader
	        - --controller-class=k8s.io/ingress-nginx-v2</pre>	

Task	Description	Skills required

	``
	      - --ingress-
	class=nginx-v2
	``
	6. In the `kind: IngressClass` section, enter the same values for `--controller-class` and `--ingress-class` that you used in the previous section:
	``
	apiVersion: networking.k8s.io/v1
	kind: IngressClass
	metadata:
	  labels:
	    app.kubernetes.io/component: controller
	    app.kubernetes.io/instance: ingress-nginx
	    app.kubernetes.io/name: ingress-nginx
	    app.kubernetes.io/part-of: ingress-nginx
	    app.kubernetes.io/version: 1.12.0
	  name: nginx-v2
	spec:
	  controller: k8s.io/ingress-nginx-v2
	``
	7. In the following section, add `loadBalancerClass: eks.amazonaws.com/nlb` to	

Task	Description	Skills required

	provision a Network Load Balancer for the NGINX Ingress Controller instance:
	
	`apiVersion: v1`
	`kind: Service`
	`metadata:`
	`  name: ingress-nginx-controller`
	`  namespace: ingress-nginx-v2`
	`spec:`
	`...`
	`  selector:`
	`    app.kubernetes.io/component: controller`
	`    app.kubernetes.io/instance: ingress-nginx`
	`    app.kubernetes.io/name: ingress-nginx`
	`  type: LoadBalancer`
	`  loadBalancerClass: eks.amazonaws.com/nlb`
	
	8. Save and close the **deploy.yaml** template.	

Deploy the new NGINX Instance Controller instance.	Enter the following command to apply the modified manifest file: `kubectl apply -f deploy.yaml`	DevOps engineer

Task	Description	Skills required

Confirm successful deployment.	Enter the following command to verify that the resources in the `ingress-nginx-v2` namespace are operational:
	`kubectl get all -n ingress-nginx-v2`
	In the output, confirm that NGINX Ingress Controller pods are in a running state. The following is an example output:
	`NAME READY STATUS RESTARTS AGE`
	`pod/ingress-nginx-admission-create-7shrj 0/1 Completed 0 24s`
	`pod/ingress-nginx-admission-patch-vkxr5 0/1 Completed 1 24s`
	`pod/ingress-nginx-controller-757bfcbc6d-4fw52 1/1 Running 0 24s`
	
	`NAME TYPE CLUSTER-IP EXTERNAL-IP`
		DevOps engineer

Task	Description	Skills required

	``
	 PORT(S)
	      AGE
	service/ingress-
	nginx-controller
	         LoadBalancer
	   10.100.208.114
	 k8s-ingressn-ingre
	ssn-2e5e37fab6-848
	337cd9c9d520f.elb.eu-
	west-1.amazonaws.com
	   80:31469/TCP,443:3
	0658/TCP 24s
	service/ingress-
	nginx-controller-
	admission ClusterIP
	      10.100.150.114
	 <none>
	
	         443/TCP
	              24s
	
	NAME
	
	 READY UP-TO-DATE
	 AVAILABLE AGE
	deployment.apps/
	ingress-nginx-co
	ntroller 1/1 1
	         1 24s
	
	NAME
	
	      DESIRED CURRENT
	  READY AGE
	replicaset.apps/
	ingress-nginx-co
	ntroller-757bfcbc6d
	 1 1 1
	     24s
	``	

Task	Description	Skills required

	``
	NAME
	
	 STATUS COMPLETIONS
	   DURATION AGE
	job.batch/ingress-
	nginx-admission-create
	   Complete 1/1
	    4s 24s
	job.batch/ingress-
	nginx-admission-patch
	   Complete 1/1
	    5s 24s
	``	

Create a new ingress for the sample HTTPd workload.	Enter the following command to create a new ingress for the existing sample HTTPd workload: ``` kubectl create ingress demo-new --class=n ginx-v2 \ --rule nginxauto mode.local.dev/=de mo:80 ```	DevOps engineer

Task	Description	Skills required

Confirm that the new ingress works.	Enter the following command to confirm that the new ingress works:
	`curl -H "Host: nginxautomode.local.dev" k8s-ingressn-ingressn-2e5e37fab6-848337cd9c9d520f.elb.eu-west-1.amazonaws.com`
	The following is the expected output:
	`<html><body><h1>It works!</h1></body></html>`
		DevOps engineer

Cut over

Task	Description	Skills required
Cut over to the new namespace.	1. (Optional) Follow the instructions in [Editing records](#) (Route 53 documentation) to update the DNS record. 2. When you have confirmed that the new NGINX Ingress Controller instance is operating as expected, delete the original. 3. Delete the self-managed AWS Load Balancer	AWS DevOps, DevOps engineer

Task	Description	Skills required
	Controller. For instructions, see Migrate apps from deprecated ALB Ingress Controller (Amazon EKS documentation). 4. Drain the managed node groups. For instructions, see Deleting and draining node groups (eksctl documentation).	

Task	Description	Skills required

Review the two ingresses.	Enter the following command to review the two ingresses that were created for the sample HTTPd workload:
	`kubectl get ingress`
	The following is an example output:
	<pre>NAME CLASS
	 HOSTS
	                 ADDRESS
	
	 PORTS AGE
	demo nginx
	  nginxautomode.loca
	l.dev k8s-ingre
	ssn-ingressn-abcde
	fg-12345.elb.eu-we
	st-1.amazonaws.com
	
	  80 95m
	demo-new nginx-v2
	  nginxautomode.loca
	l.dev k8s-ingre
	ssn-ingressn-2e5e3
	7fab6-848337cd9c9d
	520f.elb.eu-west-1
	.amazonaws.com
	        80 33s</pre>
		DevOps engineer

Related resources

- Enable EKS Auto Mode on an existing cluster (Amazon EKS documentation)

- Troubleshoot load balancers created by the Kubernetes service controller in Amazon EKS (AWS re:Post Knowledge Center)
- NGINX Ingress Controller (NGINX documentation)

Migrate your container workloads from Azure Red Hat OpenShift (ARO) to Red Hat OpenShift Service on AWS (ROSA)

Naveen Ramasamy, Srikanth Rangavajhala, and Gireesh Sreekantan, Amazon Web Services

Summary

This pattern provides step-by-step instructions for migrating container workloads from Azure Red Hat OpenShift (ARO) to Red Hat OpenShift Service on AWS (ROSA). ROSA is a managed Kubernetes service provided by Red Hat in collaboration with AWS. It helps you deploy, manage, and scale your containerized applications by using the Kubernetes platform, and benefits from both Red Hat's expertise in Kubernetes and the AWS Cloud infrastructure.

Migrating container workloads from ARO, from other clouds, or from on premises to ROSA involves transferring applications, configurations, and data from one platform to another. This pattern helps ensure a smooth transition while optimizing for AWS Cloud services, security, and cost efficiency. It covers two methods for migrating your workloads to ROSA clusters: CI/CD and Migration Toolkit for Containers (MTC).

This pattern covers both methods. The method you choose depends on the complexity and certainty of your migration process. If you have full control over your application's state and can guarantee a consistent setup through a pipeline, we recommend that you use the CI/CD method. However, if your application state involves uncertainties, unforeseen changes, or a complex ecosystem, we recommend that you use MTC as a reliable and controlled path to migrate your application and its data to a new cluster. For a detailed comparison of the two methods, see the Additional information section.

Benefits of migrating to ROSA:

- ROSA seamlessly integrates with AWS as a native service. It is easily accessible through the AWS Management Console and billed through a single AWS account. It offers full compatibility with other AWS services and provides collaborative support from both AWS and Red Hat.
- ROSA supports hybrid and multi-cloud deployments. It enables applications to run consistently across on-premises data centers and multiple cloud environments.
- ROSA benefits from Red Hat's security focus, and provides features such as role-based access control (RBAC), image scanning, and vulnerability assessments to ensure a secure container environment.

- ROSA is designed to scale applications easily and provides high availability options. It allows applications to grow as needed while maintaining reliability.
- ROSA automates and simplifies the deployment of a Kubernetes cluster compared with manual setup and management methods. This accelerates the development and deployment process.
- ROSA benefits from AWS Cloud services, and provides seamless integration with AWS offerings such as database services, storage solutions, and security services.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Permissions configured for AWS services that ROSA relies on to deliver functionality. For more information, see Prerequisites in the ROSA documentation.
- ROSA enabled on the ROSA console. For instructions, see the ROSA documentation.
- The ROSA cluster installed and configured. For more information, see Get started with ROSA in the ROSA documentation. To understand the different methods for setting up a ROSA cluster, see the AWS Prescriptive Guidance guide ROSA implementation strategies.
- Network connectivity established from the on-premises network to AWS through AWS Direct Connect (preferred) or AWS Virtual Private Network (AWS VPN).
- An Amazon Elastic Compute Cloud (Amazon EC2) instance or another virtual server to install tools such as `aws client`, OpenShift CLI (`oc`) client, ROSA client, and Git binary.

Additional prerequisites for the CI/CD method:

- Access to the on-premises Jenkins server with permissions to create a new pipeline, add stages, add OpenShift clusters, and perform builds.
- Access to the Git repository where application source code is maintained, with permissions to create a new Git branch and perform commits to the new branch.

Additional prerequisites for the MTC method:

- An Amazon Simple Storage Service (Amazon S3) bucket, which will be used as a replication repository.
- Administrative access to the source ARO cluster. This is required to set up the MTC connection.

Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

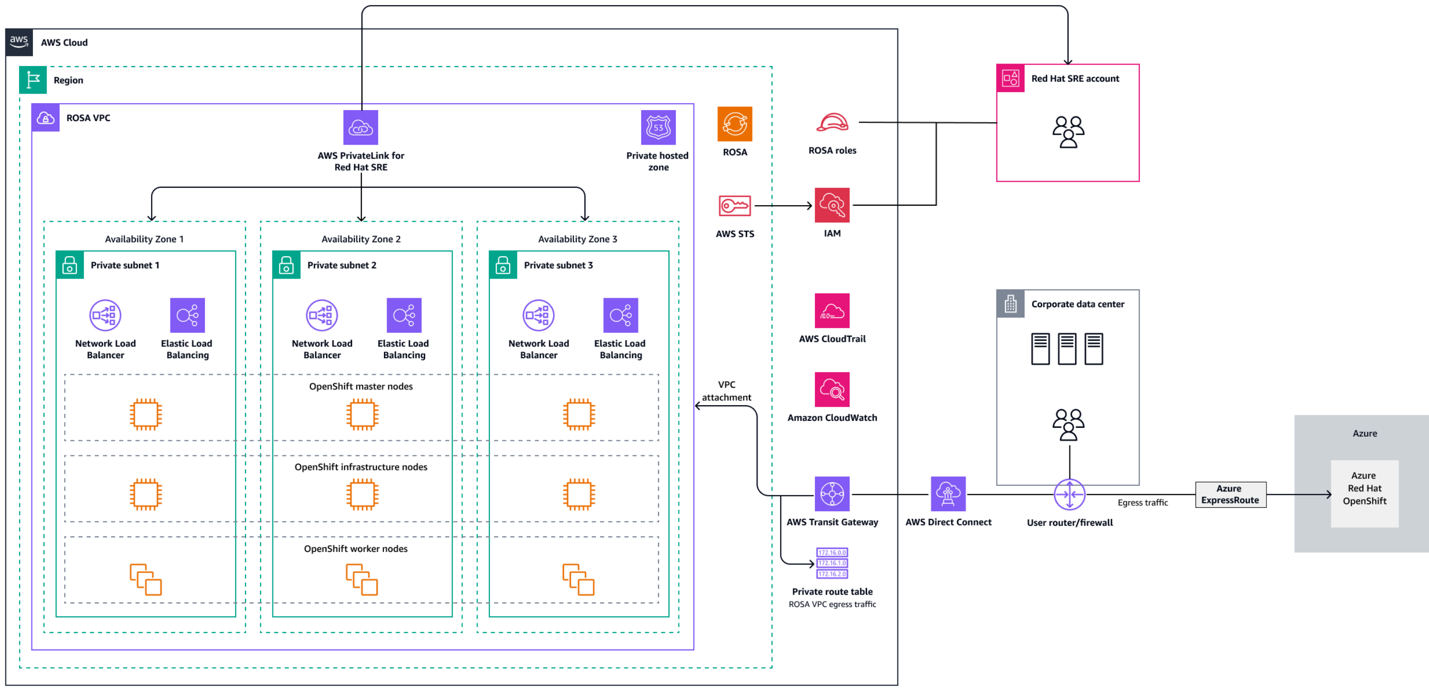
ROSA provides three network deployment patterns: public, private, and AWS PrivateLink. PrivateLink enables Red Hat site reliability engineering (SRE) teams to manage the cluster by using a private subnet connected to the cluster's PrivateLink endpoint in an existing VPC.

Choosing the PrivateLink option provides the most secure configuration. For that reason, we recommend it for sensitive workloads or strict compliance requirements. For information about the public and private network deployment options, see the Red Hat OpenShift documentation.

Important

You can create a PrivateLink cluster only at installation time. You cannot change a cluster to use PrivateLink after installation.

The following diagram illustrates the PrivateLink architecture for a ROSA cluster that uses AWS Direct Connect to connect to the on-premises and ARO environments.



AWS permissions to ROSA

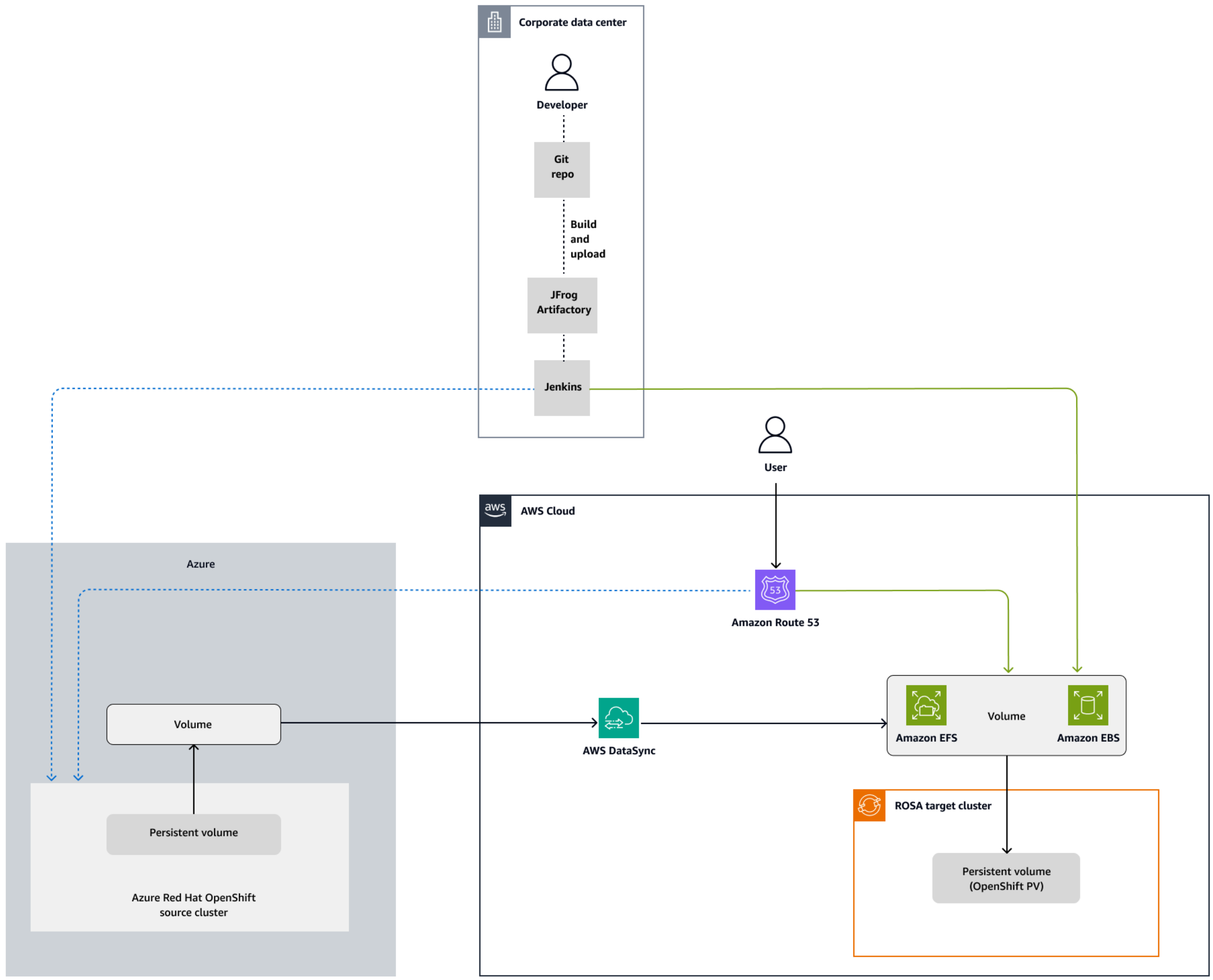
For AWS permissions to ROSA, we recommend that you use AWS Security Token Service (AWS STS) with short-lived, dynamic tokens. This method uses least-privilege predefined roles and policies to grant ROSA minimal permissions to operate in the AWS account, and supports ROSA installation, control plane, and compute functionality.

CI/CD pipeline redeployment

CI/CD pipeline redeployment is the recommended method for users who have a mature CI/CD pipeline. When you choose this option, you can use any DevOps deployment strategy to gradually shift your application load to deployments on ROSA.

> **Note**
>
> This pattern assumes a common use case where you have an on-premises Git, JFrog Artifactory, and Jenkins pipeline. This approach requires that you establish network connectivity from your on-premises network to AWS through AWS Direct Connect, and that you set up the ROSA cluster before you follow the instructions in the Epics section. See the Prerequisites section for details.

The following diagram shows the workflow for this method.



MTC method

You can use the [Migration Toolkit for Containers (MTC)]() to migrate your containerized workloads between different Kubernetes environments, such as from ARO to ROSA. MTC simplifies the migration process by automating several key tasks and providing a comprehensive framework for managing the migration lifecycle.

The following diagram shows the workflow for this method.

Tools

AWS services

- AWS DataSync is an online data transfer and discovery service that helps you move files or object data to, from, and between AWS storage services.
- AWS Direct Connect links your internal network to a AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- AWS PrivateLink helps you create unidirectional, private connections from your virtual private clouds (VPCs) to services outside of the VPC.
- Red Hat OpenShift Service on AWS (ROSA) is a managed service that helps Red Hat OpenShift users to build, scale, and manage containerized applications on AWS.
- AWS Security Token Service (AWS STS) helps you request temporary, limited-privilege credentials for users.

Other tools

- Migration Toolkit for Containers (MTC) provides a console and API for migrating containerized applications from ARO to ROSA.

Best practices

- For resilience and if you have security compliance workloads, set up a Multi-AZ ROSA cluster that uses PrivateLink. For more information, see the ROSA documentation.

> **Note**
>
> PrivateLink cannot be configured after installation.

- The S3 bucket that you use for replication repository should not be public. Use the appropriate S3 bucket policies to restrict access.
- If you choose the MTC method, use the **Stage** migration option to reduce the downtime window during cutover.
- Review your service quotas before and after you provision the ROSA cluster. If necessary, request a quota increase according to your requirements. For more information, see the Service Quotas documentation.
- Review the ROSA security guidelines and implement security best practices.
- We recommend that you remove the default cluster administrator after installation. For more information, see the Red Hat OpenShift documentation.
- Use machine pool automatic scaling to scale down unused worker nodes in the ROSA cluster to optimize costs. For more information, see the ROSA Workshop.
- Use the Red Hat Cost Management service for OpenShift Container Platform to better understand and track costs for clouds and containers. For more information, see the ROSA Workshop.
- Monitor and audit ROSA cluster infrastructure services and applications by using AWS services. For more information, see the ROSA Workshop.

Epics

Option 1: Use a CI/CD pipeline

Task	Description	Skills required
Add the new ROSA cluster to Jenkins.	1. On the Jenkins console, choose **Manage Jenkins**, **Configure System**.	AWS administrator, AWS systems administrator, AWS DevOps

Task	Description	Skills required
	2. On the **Manage Jenkins** page, in the **OpenShift Plug-in** section, choose **Add a new cluster**. 3. Provide required information such as the cluster name, API server URL, and token information to authenticate with ROSA.	
Add the `oc` client to your Jenkins nodes.	1. On the Jenkins console, choose **Manage Jenkins**, **Global Tool Configuration**. 2. In the **OpenShift Client Tools** section, install the OpenShift CLI (`oc`) client version that is the same as your ROSA cluster version.	AWS administrator, AWS systems administrator, AWS DevOps
Create a new Git branch.	Create a new branch in your Git repository for `rosa-dev`. This branch separates the code or configuration parameter changes for ROSA from your existing pipeline.	AWS DevOps
Tag images for ROSA.	In your build stage, use a different tag to identify the images that are built from the ROSA pipeline.	AWS administrator, AWS systems administrator, AWS DevOps

Task	Description	Skills required
Create a pipeline.	Create a new Jenkins pipeline that's similar to your existing pipeline. For this pipeline, use the `rosa-dev` Git branch that you created earlier, and make sure to include the Git checkout, code scan, and build stages that are identical to your existing pipeline.	AWS administrator, AWS systems administrator, AWS DevOps
Add a ROSA deployment stage.	In the new pipeline, add a stage to deploy to the ROSA cluster and reference the ROSA cluster that you added to the Jenkins global configuration.	AWS administrator, AWS DevOps, AWS systems administrator
Start a new build.	In Jenkins, select your pipeline and choose **Build now**, or start a new build by committing a change to the `rosa-dev` branch in Git.	AWS administrator, AWS DevOps, AWS systems administrator
Verify the deployment.	Use the **oc** command or the [ROSA console](https://console.redhat.com/openshift) to verify that the application has been deployed on your target ROSA cluster.	AWS administrator, AWS DevOps, AWS systems administrator

Task	Description	Skills required
Copy data to the target cluster.	For stateful workloads, copy the data from the source cluster to the target cluster by using AWS DataSync or open source utilities such as **rsync**, or you can use the MTC method. For more informati on, see the [AWS DataSync documentation]().	AWS administrator, AWS DevOps, AWS systems administrator
Test your application.	1. Retrieve the route URL for your application by using the **oc route** command or the [ROSA console](). 2. Test your application by using the route URL.	AWS administrator, AWS DevOps, AWS systems administrator
Cut over.	If your testing is successful, use the appropriate Amazon Route 53 policy to move the traffic from the ARO-hoste d application to the ROSA-hosted application. When you complete this step, your application's workload will fully transition to the ROSA cluster.	AWS administrator, AWS systems administrator

Option 2: Use MTC

Task	Description	Skills required
Install the MTC operator.	Install the MTC operator on both ARO and ROSA clusters: 1. In the ARO or ROSA console, navigate to the **Operators**, **OperatorHub** page. 2. In the **Filter by** keyword box, find or type **MTC**. 3. Select the MTC operator to display additional informati on. 4. After you read the information about the operator, choose **Install**.	AWS administrator, AWS DevOps, AWS systems administrator
Configure network traffic to the replication repository.	If you're using a proxy server, configure it to allow network traffic between the replication repository and the clusters. The replication repository is an intermedi ate storage object that MTC uses to migrate data. The source and target clusters must have network access to the replication repository during migration.	AWS administrator, AWS DevOps, AWS systems administrator
Add the source cluster to MTC.	On the MTC web console, add the ARO source cluster.	AWS administrator, AWS DevOps, AWS systems administrator

Task	Description	Skills required
Add Amazon S3 as your replication repository.	On the MTC web console, add the Amazon S3 bucket (see Prerequisites) as the replicati on repository.	AWS administrator, AWS DevOps, AWS systems administrator
Create a migration plan.	On the MTC web console, create a migration plan and specify the data transfer type as **Copy**. This will instruct MTC to copy the data from the source (ARO) cluster to the S3 bucket, and from the bucket to the target (ROSA) cluster.	AWS administrator, AWS DevOps, AWS systems administrator

Task	Description	Skills required
Run the migration plan.	Run the migration plan by using the **Stage** or **Cutover** option: • Choose **Stage** to copy data to the target cluster without stopping your application. You can run multiple stage migrations to copy most of your data before cutover migration. This shortens the duration of cutover migration. • Choose **Cutover** to stop your application on the source cluster while moving the resources to the target cluster. To prevent stopping the application, you can clear the **Halt transactions on the source cluster during migration** check box.	AWS administrator, AWS DevOps, AWS systems administrator

Troubleshooting

Issue	Solution
Connectivity issues	When you migrate your container workloads from ARO to ROSA, you might encounter connectivity issues that should be resolved to ensure a successful migration. To address these connectivity issues (listed in this table)

Issue	Solution
	during migration, meticulous planning, coordination with your network and security teams, and thorough testing are vital. Implementing a gradual migration strategy and verifying connectivity at each step will help minimize potential disruptions and ensure a smooth transition from ARO to ROSA.
Network configuration differences	ARO and ROSA might have variations in their network configurations, such as virtual network (VNet) settings, subnets, and network policies. For proper communication between services, make sure that the network settings align between the two platforms.
Security group and firewall rules	ROSA and ARO might have different default security group and firewall settings. Make sure to adjust and update these rules to permit necessary traffic to maintain connectiv ity among containers and services during migration.
IP address and DNS changes	When you migrate workloads, IP addresses and DNS names might change. Reconfigure applications that rely on static IPs or specific DNS names.
External service access	If your application depends on external services such as databases or APIs, you might have to update their connection settings to make sure they can communicate with the new services from ROSA.

Issue	Solution
Azure Private Link configuration	If you use Azure Private Link or private endpoint services in ARO, you will need to set up the equivalent functionality in ROSA to ensure private connectivity between resources.
AWS VPN or AWS Direct Connect setup	If there are existing AWS VPN or AWS Direct Connect connections between your on-premises network and ARO, you will need to establish similar connections with ROSA for uninterrupted communication with your local resources.
Ingress and load balancer settings	Configurations for ingress controllers and load balancers might differ between ARO and ROSA. Reconfigure these settings to maintain external access to your services.
Certificate and TLS handling	If your applications use SSL certificates or TLS, make sure that the certificates are valid and configured correctly in ROSA.
Container registry access	If your containers are hosted in an external container registry, set up the proper authentic ation and access permissions for ROSA.
Monitoring and logging	Update monitoring and logging configurations to reflect the new infrastructure on ROSA so you can continue to monitor the health and performance of your containers effectively.

Related resources

AWS references

- What is Red Hat OpenShift Service on AWS? (ROSA documentation)
- Get started with ROSA (ROSA documentation)

- Red Hat OpenShift Service on AWS implementation strategies (AWS Prescriptive Guidance)
- Red Hat OpenShift Service on AWS Now GA (AWS blog post)
- ROSA Workshop
- ROSA FAQ
- ROSA Workshop FAQ
- ROSA pricing

Red Hat OpenShift documentation

- Installing a cluster quickly on AWS
- Installing a cluster on AWS in a restricted network
- Installing a cluster on AWS into an existing VPC
- Installing a cluster on user-provisioned infrastructure in AWS by using CloudFormation templates
- Installing a cluster on AWS in a restricted network with user-provisioned infrastructure
- Installing a cluster on AWS with customizations
- Getting started with the OpenShift CLI

Additional information

Choosing between MTC and CI/CD pipeline redeployment options

Migrating applications from one OpenShift cluster to another requires careful consideration. Ideally, you would want a smooth transition by using a CI/CD pipeline to redeploy the application and handle the migration of persistent volume data. However, in practice, a running application on a cluster is susceptible to unforeseen changes over time. These changes can cause the application to gradually deviate from its original deployment state. MTC offers a solution for scenarios where the exact contents of a namespace are uncertain and a seamless migration of all application components to a new cluster is important.

Making the right choice requires evaluating your specific scenario and weighing the benefits of each approach. By doing so, you can ensure a successful and seamless migration that aligns with your needs and priorities. Here are additional guidelines for choosing between the two options.

CI/CD pipeline redeployment

The CI/CD pipeline method is the recommended approach if your application can be confidently redeployed by using a pipeline. This ensures that the migration is controlled, predictable, and aligned with your existing deployment practices. When you choose this method, you can use blue/green deployment or canary deployment strategies to gradually shift the load to deployments on ROSA. For this scenario, this pattern assumes that Jenkins is orchestrating application deployments from the on-premises environment.

Advantages:

- You do not require administrative access to the source ARO cluster or need to deploy any operators on the source or destination cluster.
- This approach helps you switch traffic gradually by using a DevOps strategy.

Disadvantages:

- It requires more effort to test the functionality of your application.
- If your application contains persistent data, it requires additional steps to copy the data by using AWS DataSync or other tools.

MTC migration

In the real world, running applications can undergo unanticipated changes that cause them to drift away from the initial deployment. Choose the MTC option when you're unsure about the current state of your application on the source cluster. For example, if your application ecosystem spans various components, configurations, and data storage volumes, we recommend that you choose MTC to ensure a complete migration that covers the application and its entire environment.

Advantages:

- MTC provides complete backup and restore of the workload.
- It will copy the persistent data from source to target while migrating the workload.
- It does not require access to the source code repository.

Disadvantages:

- You need administrative privileges to install the MTC operator on the source and destination clusters.

- The DevOps team requires training to use the MTC tool and perform migrations.

Place Kubernetes Pods on Amazon EKS by using node affinity, taints, and tolerations

Hitesh Parikh and Raghu Bhamidimarri, Amazon Web Services

Summary

This pattern demonstrates the use of Kubernetes *node affinity*, node *taints*, and Pod *tolerations* to intentionally schedule application Pods on specific worker nodes in an Amazon Elastic Kubernetes Service (Amazon EKS) cluster on the Amazon Web Services (AWS) Cloud.

A *taint* is a node property that enables nodes to reject a set of pods. A *toleration* is a Pod property that enables the Kubernetes scheduler to schedule Pods on nodes that have matching taints.

However, tolerations alone can’t prevent a scheduler from placing a Pod on a worker node that doesn’t have any taints. For example, a compute intensive Pod with a toleration can unintentionally get scheduled on a general-purpose untainted node. In that scenario, the *node affinity* property of a Pod instructs the scheduler to place the Pod on a node that meets the node selection criteria specified in the node affinity.

Taints, tolerations, and node affinity together instruct the scheduler to schedule Pods consistently on the nodes with matching taints and the node labels that match the node affinity node-selection criteria specified on the Pod.

This pattern provides an example Kubernetes deployment manifest file, and the steps to create an EKS cluster, deploy an application, and validate Pod placement.

Prerequisites and limitations

Prerequisites

- An AWS account with credentials configured to create resources on your AWS account
- AWS Command Line Interface (AWS CLI)
- eksctl
- kubectl
- Docker installed (for the operating system being used), and the engine started (for information about Docker licensing requirements, see the Docker site)

- Java version 11 or later
- A Java microservice running on your favorite integrated development environment (IDE); for example, IntelliJ IDEA Community Edition or Eclipse (if you don't have a Java microservice, see the Deploy a sample Java microservice on Amazon EKS pattern and Microservices with Spring for help with creating the microservice)

Limitations

- This pattern doesn't provide the Java code, and it assumes that you are already familiar with Java. To create a basic Java microservice, see Deploy a sample Java microservice on Amazon EKS.
- The steps in this article create AWS resources that can accrue cost. Make sure that you clean up the AWS resources after you have completed the steps to implement and validate the pattern.

Architecture

Target technology stack

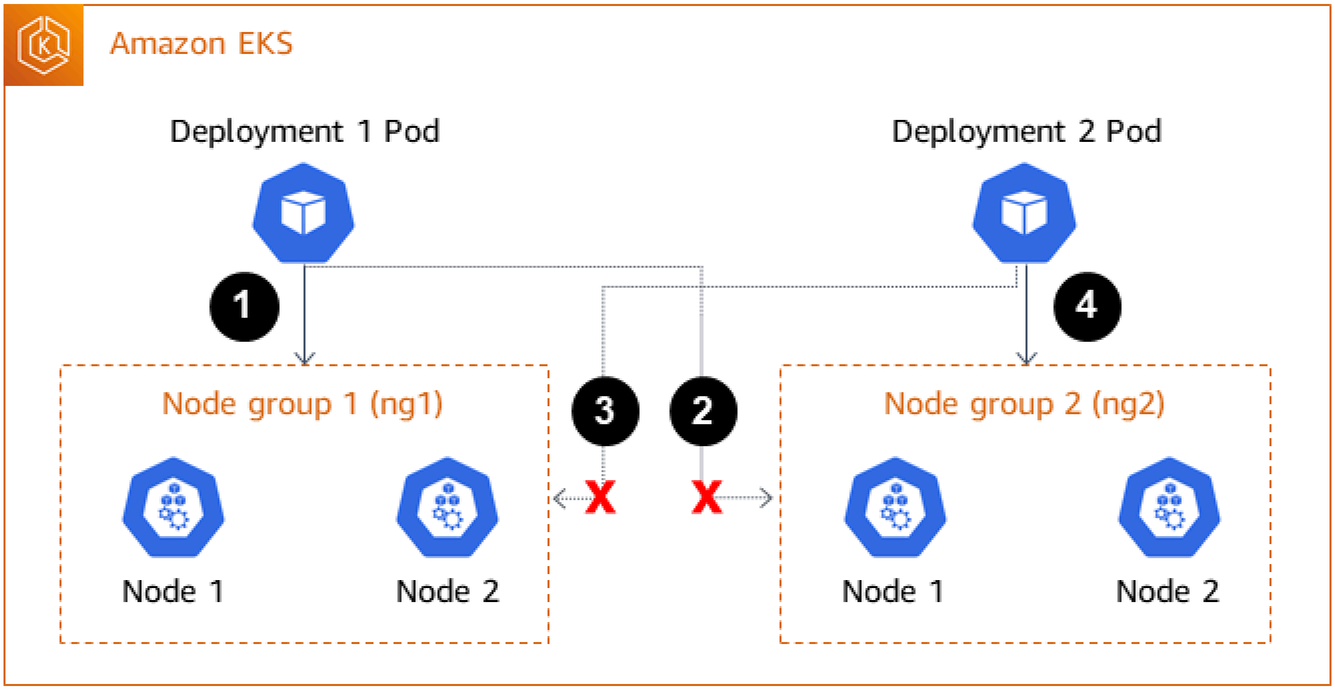
- Amazon EKS
- Java
- Docker
- Amazon Elastic Container Registry (Amazon ECR)

Target architecture

The solution architecture diagram shows Amazon EKS with two Pods (Deployment 1 and Deployment 2) and two node groups (ng1 and ng2) with two nodes each. The Pods and nodes have the following properties.

	Deployment 1 Pod	Deployment 2 Pod	Node group 1 (ng1)	Node group 2 (ng2)
Toleration	key: classified_workload, value: true, effect: NoSchedule	None		

	key: machine_learning_workload, value: true, effect: NoSchedule			
Node affinity	key: alpha.eksctl.io/nodegroup-name = ng1;	None	nodeGroups.name = ng1	
Taint			key: classified_workload, value: true, effect: NoSchedule key: machine_learning_workload, value: true, effect: NoSchedule	None



1. The Deployment 1 Pod has tolerations and node affinity defined, which instructs the Kubernetes scheduler to place the deployment Pods on the Node group 1 (ng1) nodes.
2. Node group 2 (ng2) doesn't have a node label that matches the node affinity node selector expression for Deployment 1, so the Pods will not be scheduled on ng2 nodes.
3. The Deployment 2 Pod doesn't have any tolerations or node affinity defined in the deployment manifest. The scheduler will reject scheduling Deployment 2 Pods on Node group 1 because of the taints on the nodes.
4. The Deployment 2 Pods will be placed on Node group 2 instead, because the nodes don't have any taints.

This pattern demonstrates that by using taints and tolerations, combined with node affinity, you can control placement of Pods on specific sets of worker nodes.

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.

- [Amazon Elastic Kubernetes Service (Amazon EKS)](#) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- [eksctl](#) is AWS equivalent of kubectl and helps with creating EKS.

Other tools

- [Docker](#) is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- [kubectl](#) is a command-line interface that helps you run commands against Kubernetes clusters.

Epics

Create the EKS cluster

Task	Description	Skills required
Create the cluster.yaml file.	Create a file called `cluster.yaml` with the following code. ```apiVersion: eksctl.io/v1alpha5 kind: ClusterConfig metadata: name: eks-taint-demo region: us-west-1 # Unmanaged nodegroups with and without taints. nodeGroups: - name: ng1 instanceType: m5.xlarge minSize: 2 maxSize: 3 taints:```	App owner, AWS DevOps, Cloud administrator, DevOps engineer

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: eks-taint-demo
  region: us-west-1

# Unmanaged nodegroups with and without taints.
nodeGroups:
  - name: ng1
    instanceType: m5.xlarge
    minSize: 2
    maxSize: 3
    taints:
```

Task	Description	Skills required

	``
	      - key: classifie
	d_workload
	        value: "true"
	        effect:
	 NoSchedule
	      - key: machine_l
	earning_workload
	        value: "true"
	        effect:
	 NoSchedule
	
	  - name: ng2
	    instanceType:
	 m5.xlarge
	    minSize: 2
	    maxSize: 3
	``	

Create the cluster by using eksctl.	Run the `cluster.yaml` file to create the EKS cluster. Creating the cluster might take a few minutes. `eksctl create cluster -f cluster.yaml`	AWS DevOps, AWS systems administrator, App developer

Create an image and upload it to Amazon ECR

Task	Description	Skills required
Create an Amazon ECR private repository.	To create an Amazon ECR repository, see [Creating a private repository](). Note the URI of the repo.	AWS DevOps, DevOps engineer, App developer
Create the Dockerfile.	If you have an existing Docker container image that you want to use to test the	AWS DevOps, DevOps engineer

Task	Description	Skills required

	pattern, you can skip this step.
	
	To create a Dockerfile, use the following snippet as a reference. If you encounter errors, see the [Troubleshooting](#) section.
	
	`FROM adoptopenjdk/openjdk11:jdk-11.0.14.1_1-alpine`
	
	`RUN apk add maven`
	`WORKDIR /code`
	
	`# Prepare by downloading dependencies`
	`ADD pom.xml /code/pom.xml`
	`RUN ["mvn", "dependency:resolve"]`
	`RUN ["mvn", "verify"]`
	
	`# Adding source, compile and package into a fat jar`
	`ADD src /code/src`
	`RUN ["mvn", "package"]`
	
	`EXPOSE 4567`
	`CMD ["java", "-jar", "target/eksExample-jar-with-dependencies.jar"]`	

Task	Description	Skills required
Create the pom.xml and source files, and build and push the Docker image.	To create the `pom.xml` file and the Java source file, see [Deploy a sample Java microservice on Amazon EKS](#) pattern. Use the instructions in that pattern to build and push the Docker image.	AWS DevOps, DevOps engineer, App developer

Deploy to Amazon EKS

Task	Description	Skills required
Create the deployment.yaml file.	To create the `deploymen t.yaml` file, use the code in the [*Additional information*](#) section. In the code, the key for node affinity is any label that you create while creating node groups. This pattern uses the default label created by eksctl. For information about customizing labels, see [Assigning Pods to Nodes](#) in the Kubernetes documenta tion. The value for the node affinity key is the name of the node group that was created by `cluster.yaml` .	AWS DevOps, DevOps engineer, App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>To get the key and value for the taint, run the following command.
<pre>kubectl get nodes -o
 json | jq '.items[]
.spec.taints'</pre>The image is the URI of the Amazon ECR repository that you created in an earlier step.</td><td></td></tr>
<tr><td>Deploy the file.</td><td>To deploy to Amazon EKS, run the following command.
<pre>kubectl apply -f
 deployment.yaml</pre></td><td>App developer, DevOps engineer, AWS DevOps</td></tr>
</table>

Task	Description	Skills required

Check the deployment.	1. To check if the pods are READY, run the following command.
	`kubectl get pods -o wide`
	If the POD is ready, the output should look similar to the following, with the STATUS as Running.
	`NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES`
	`<pod_name> 1/1 Running 0 12d 192.168.18.50 ip-192-168-20-110.us-west-1.compute.internal <none> <none>`
	Note the name of the Pod and the name of the node. You can skip the next step.
	2. (Optional) To get additional details about the Pod and check the tolerations on the Pod, run the following command.
		App developer, DevOps engineer, AWS DevOps

Task	Description	Skills required

	`kubectl describe pod <pod_name>`
	
	An example of the output is in the *Additional information* section.
	
	3. To validate that the Pod placement on the node is correct, run the following command.
	
	`kubectl describe node <node name> | grep -A 1 "Taints"`
	
	Confirm that the taint on the node matches toleration, and the label on the node matches the node affinity defined in `deployment.yaml` .
	
	The Pod with tolerations and node affinity should be placed on a node with the matching taints and the node affinity labels. The previous command gives you the taints on the node. The following is an example output.
	
	`kubectl describe node ip-192-168-29-181.us-west-1.compute.`	

Task	Description	Skills required

	<pre>internal | grep -A 1
	 "Taints"
	Taints:
	 classifled_workloa
	d=true:NoSchedule
	
	 machine_learning_w
	orkload=true:NoSch
	edule</pre>Additionally, run the following command to check that the node on which the Pod is placed has a label matching the node affinity node label.<pre>kubectl get node <node
	 name> --show-labels</pre>4. To verify that the applicati on is doing what it is intended to do, check the Pod logs by running the following command.<pre>kubectl logs -f <name-
	of-the-pod></pre>	

Task	Description	Skills required

Create a second deploymen t .yaml file without toleration and node affinity.	This additional step is to validate that when no node affinity or tolerations are specified in the deploymen t manifest file, the resulting Pod is not scheduled on a node with taints. (It should be scheduled on a node that doesn't have any taints). Use the following code to create a new deployment file called `deploy_no_taint.yaml` .
	`apiVersion: apps/v1`
	`kind: Deployment`
	`metadata:`
	`  name: microservice-deployment-non-tainted`
	`spec:`
	`  replicas: 1`
	`  selector:`
	`    matchLabels:`
	`      app.kubernetes.io/name: java-microservice-no-taint`
	`  template:`
	`    metadata:`
	`      labels:`
	`        app.kubernetes.io/name: java-microservice-no-taint`
	`    spec:`
	`      containers:`
	`        - name: java-microservice-container-2`
	`          image: <account_number>.dkr.ecr<region>.ama`
		App developer, AWS DevOps, DevOps engineer

Task	Description	Skills required
	```zonaws.com/<repository_name>:latest ports: - containerPort: 4567```	
Deploy the second deployment .yaml file, and validate Pod placement	1. Run the following command. ```kubectl apply -f deploy_no_taint.yaml``` 2. After the deployment is successful, run the same commands that you ran previously to check the Pod placement in a node group with no taint. ```kubectl describe node <node_name> | grep "Taints"``` The output should be the following. ```Taints: <none>``` This completes the testing.	App developer, AWS DevOps, DevOps engineer

Clean up resources

Task	Description	Skills required
Clean up the resources.	To avoid incurring AWS charges for resources that are left running, use the following command. `eksctl delete cluster --name <Name of the cluster> --region <region-code>`	AWS DevOps, App developer

Troubleshooting

Issue	Solution
Some of these commands might not run if your system uses arm64 architecture (especially if you are running this on an M1 Mac). The following line might error out. `FROM adoptopenjdk/openjdk11:jdk-11.0.14.1_1-alpine`	If you have errors when running the Dockerfile, replace the `FROM` line with the following line. `FROM bellsoft/liberica-openjdk-alpine-musl:17`

Related resources

- Deploy a sample Java microservice on Amazon EKS
- Create an Amazon ECR private repository
- Assigning Pods to Nodes (Kubernetes documentation)
- Taints and Tolerations (Kubernetes documentation)
- Amazon EKS
- Amazon ECR

- [AWS CLI]
- [Docker]
- [IntelliJ IDEA CE]
- [Eclipse]

Additional information

deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: microservice-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: java-microservice
  template:
    metadata:
      labels:
        app.kubernetes.io/name: java-microservice
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: alpha.eksctl.io/nodegroup-name
                    operator: In
                    values:
                      - <node-group-name-from-cluster.yaml>
      tolerations: #only this pod has toleration and is viable to go to ng with taint
        - key: "<Taint key>" #classified_workload in our case
          operator: Equal
          value: "<Taint value>" #true
          effect: "NoSchedule"
        - key: "<Taint key>" #machine_learning_workload in our case
          operator: Equal
          value: "<Taint value>" #true
          effect: "NoSchedule"
```

```
      containers:
        - name: java-microservice-container
          image: <account_number>.dkr.ecr<region>.amazonaws.com/
<repository_name>:latest
          ports:
            - containerPort: 4567
```

describe pod example output

```
Name:          microservice-deployment-in-tainted-nodes-5684cc495b-vpcfx
Namespace:     default
Priority:      0
Node:          ip-192-168-29-181.us-west-1.compute.internal/192.168.29.181
Start Time:    Wed, 14 Sep 2022 11:06:47 -0400
Labels:        app.kubernetes.io/name=java-microservice-taint
               pod-template-hash=5684cc495b
Annotations:   kubernetes.io/psp: eks.privileged
Status:        Running
IP:            192.168.13.44
IPs:
  IP:            192.168.13.44
Controlled By:  ReplicaSet/microservice-deployment-in-tainted-nodes-5684cc495b
Containers:
  java-microservice-container-1:
    Container ID:
 docker://5c158df8cc160de8f57f62f3ee16b12725a87510a809d90a1fb9e5d873c320a4
    Image:          934188034500.dkr.ecr.us-east-1.amazonaws.com/java-eks-apg
    Image ID:       docker-pullable://934188034500.dkr.ecr.us-east-1.amazonaws.com/
java-eks-apg@sha256:d223924aca8315aab20d54eddf3443929eba511b6433017474d01b63a4114835
    Port:           4567/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 14 Sep 2022 11:07:02 -0400
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-ddvvw (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
```

```
  PodScheduled      True
Volumes:
  kube-api-access-ddvvw:
    Type:                    Projected (a volume that contains injected data from
 multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 classifled_workload=true:NoSchedule
                             machine_learning_workload=true:NoSchedule
                             node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for
 300s
Events:                      <none>
```

Replicate filtered Amazon ECR container images across accounts or Regions

Abdal Garuba, Amazon Web Services

Summary

Amazon Elastic Container Registry (Amazon ECR) can replicate all container images in an image repository across Amazon Web Services (AWS) Regions and AWS accounts natively, by using the cross-Region and cross-account replication features. (For more information, see the AWS blog post Cross region replication in Amazon ECR has landed.) However, there is no way to filter the images that are copied across AWS Regions or accounts based on any criteria.

This pattern describes how to replicate container images that are stored in Amazon ECR across AWS accounts and Regions, based on image tag patterns. The pattern uses Amazon CloudWatch Events to listen for push events for images that have a predefined, custom tag. A push event starts an AWS CodeBuild project and passes the image details to it. The CodeBuild project copies the images from the source Amazon ECR registry to the destination registry based on the details provided.

This pattern copies images that have specific tags across accounts. For example, you can use this pattern to copy only production-ready, secure images to the production AWS account. In the development account, after images are thoroughly tested, you can add a predefined tag to the secure images and use the steps in this pattern to copy the marked images to the production account.

Prerequisites and limitations

Prerequisites

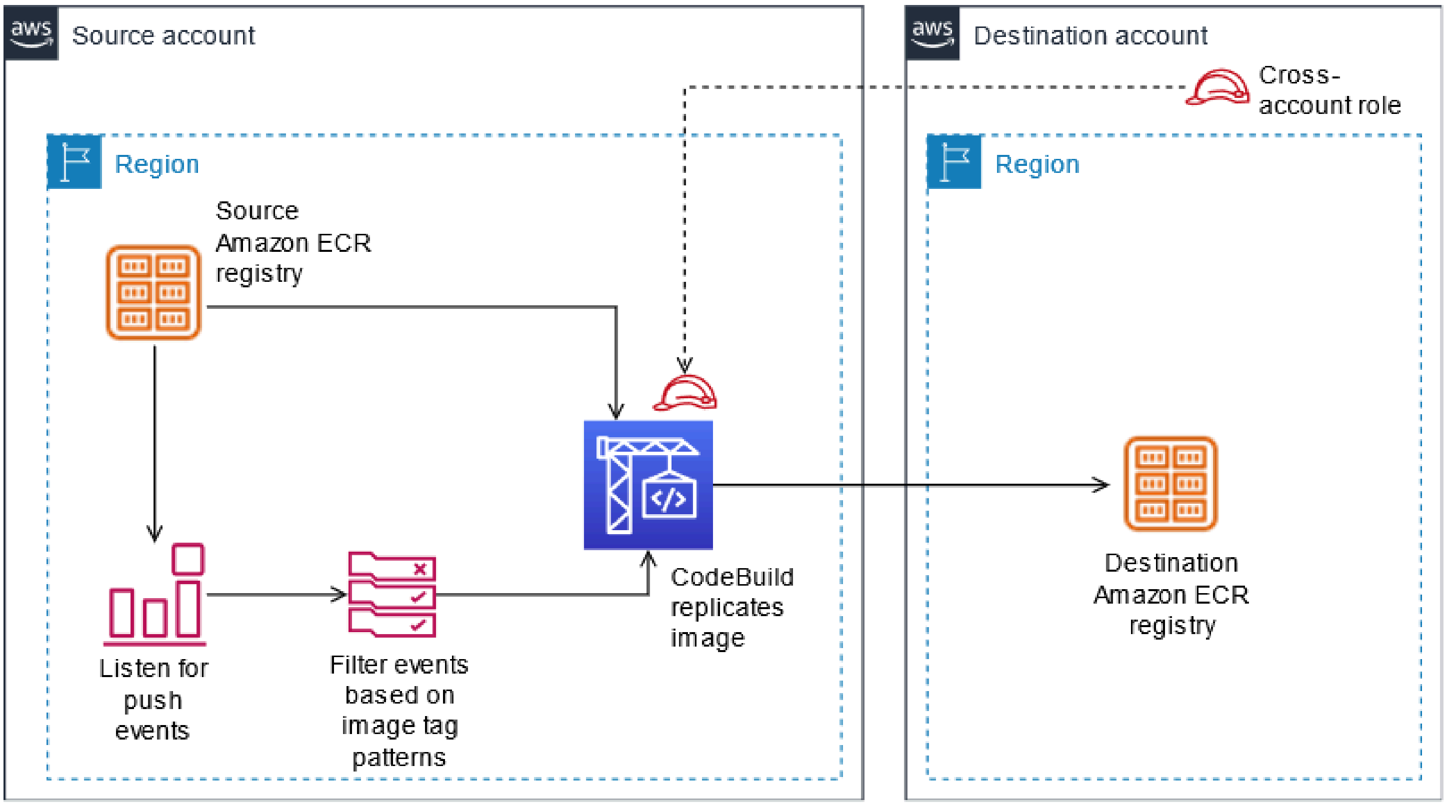
- An active AWS account for source and destination Amazon ECR registries
- Administrative permissions for the tools used in this pattern
- Docker installed on your local machine for testing
- AWS Command Line Interface (AWS CLI), for authenticating into Amazon ECR

Limitations

- This pattern watches the push events of the source registry in only one AWS Region. You can deploy this pattern to other Regions to watch registries in those Regions.
- In this pattern, one Amazon CloudWatch Events rule listens for a single image tag pattern. If you want to check for multiple patterns, you can add events to listen for additional image tag patterns.

Architecture

Target architecture



Automation and scale

This pattern can be automated with an infrastructure as code (IaC) script and deployed at scale. To use AWS CloudFormation templates to deploy this pattern, download the attachment and follow the instructions in the [Additional information](#) section.

You can point multiple Amazon CloudWatch Events events (with different custom event patterns) to the same AWS CodeBuild project to replicate multiple image tag patterns, but you will need to update the secondary validation in the `buildspec.yaml` file (which is included in the attachment and in the [Tools](#) section) as follows to support multiple patterns.

```
...
if [[ ${IMAGE_TAG} != release-* ]]; then
...
```

Tools

Amazon services

- IAM – AWS Identity and Access Management (IAM) enables you to manage access to AWS services and resources securely. In this pattern, you would need to create the cross-account IAM role that AWS CodeBuild will assume when pushing container images to the destination registry.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is a fully managed container registry that makes it easy to store, manage, share, and deploy your container images and artifacts anywhere. Image push actions to the source registry send system event details to the event bus that is picked up by Amazon CloudWatch Events.
- AWS CodeBuild – AWS CodeBuild is a fully managed continuous integration service that provides compute power to perform jobs such as compiling source code, running tests, and producing artifacts that are ready to be deployed. This pattern uses AWS CodeBuild to perform the copy action from the source Amazon ECR registry to the destination registry.
- CloudWatch Events – Amazon CloudWatch Events delivers a stream of system events that describe changes in AWS resources. This pattern uses rules to match Amazon ECR push actions with a specific image tag pattern.

Tools

- Docker CLI – Docker is a tool that makes it easier to create and manage containers. Containers pack an application and all its dependencies into one unit or package that can easily be deployed on any platform that supports the container runtime.

Code

You can implement this pattern in two ways:

- Automated setup: Deploy the two AWS CloudFormation templates provided in the attachment. For instructions, see the Additional information section.
- Manual setup: Follow the steps in the Epics section.

Sample buildspec.yaml

If you're using the CloudFormation templates that are provided with this pattern, the `buildspec.yaml` file is included in the CodeBuild resources.

```
  version: 0.2
  env:
    shell: bash
  phases:
    install:
      commands:
        - export CURRENT_ACCOUNT=$(echo ${CODEBUILD_BUILD_ARN} | cut -d':' -f5)
        - export CURRENT_ECR_REGISTRY=${CURRENT_ACCOUNT}.dkr.ecr.
${AWS_REGION}.amazonaws.com
        - export DESTINATION_ECR_REGISTRY=${DESTINATION_ACCOUNT}.dkr.ecr.
${DESTINATION_REGION}.amazonaws.com
    pre_build:
      on-failure: ABORT
      commands:
        - echo "Validating Image Tag ${IMAGE_TAG}"
        - |
          if [[ ${IMAGE_TAG} != release-* ]]; then
            aws codebuild stop-build --id ${CODEBUILD_BUILD_ID}
            sleep 60
            exit 1
          fi
        - aws ecr get-login-password --region ${AWS_REGION} | docker login -u AWS --
password-stdin ${CURRENT_ECR_REGISTRY}
        - docker pull ${CURRENT_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
    build:
      commands:
        - echo "Assume cross-account role"
        - CREDENTIALS=$(aws sts assume-role --role-arn ${CROSS_ACCOUNT_ROLE_ARN} --
role-session-name Rolesession)
        - export AWS_DEFAULT_REGION=${DESTINATON_REGION}
        - export AWS_ACCESS_KEY_ID=$(echo ${CREDENTIALS} | jq -r
 '.Credentials.AccessKeyId')
        - export AWS_SECRET_ACCESS_KEY=$(echo ${CREDENTIALS} | jq -r
 '.Credentials.SecretAccessKey')
        - export AWS_SESSION_TOKEN=$(echo ${CREDENTIALS} | jq -r
 '.Credentials.SessionToken')
        - echo "Logging into cross-account registry"
        - aws ecr get-login-password --region ${DESTINATION_REGION} | docker login -u
 AWS --password-stdin ${DESTINATION_ECR_REGISTRY}
```

```
        - echo "Check if Destination Repository exists, else create"
        - |
          aws ecr describe-repositories --repository-names ${REPO_NAME} --region
${DESTINATION_REGION} \
          || aws ecr create-repository --repository-name ${REPO_NAME} --region
${DESTINATION_REGION}
        - echo "retag image and push to destination"
        - docker tag ${CURRENT_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
${DESTINATION_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
        - docker push ${DESTINATION_ECR_REGISTRY}/${REPO_NAME}:${IMAGE_TAG}
```

Epics

Create IAM roles

Task	Description	Skills required
Create a CloudWatch Events role.	In the source AWS account, create an IAM role for Amazon CloudWatch Events to assume. The role should have permissions to start a AWS CodeBuild project. To create the role by using the AWS CLI, follow the instructions in the IAM documentation. Example trust policy (`trustpolicy.json`): `{ "Version": "2012-10-17", "Statement": { "Effect": "Allow", "Principal": {"Service": "events.amazonaws.com"},`	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect , DevOps engineer

Task	Description	Skills required

	``
	    "Action": "sts:AssumeRole"
	  }
	}
	``
	Example permission policy (`permissionpolicy.json`):
	``
	{
	  "Version": "2012-10-17",
	  "Statement": {
	    "Effect": "Allow",
	    "Action": "codebuild:StartBuild",
	    "Resource": "<CodeBuild Project ARN>"
	  }
	}
	``	

Task	Description	Skills required

Create a CodeBuild role.	Create an IAM role for AWS CodeBuild to assume, by following the [instructions in the IAM documentation](#). The role should have the following permissions:
	
	• Permission to assume the destination cross-account role
	• Permission to create log groups and log streams, and to put log events
	• Read-only permissions to all Amazon ECR repositories, by adding the [AmazonEC2ContainerRegistryReadOnly](#) managed policy to the role
	• Permission to stop CodeBuild
	
	Example trust policy (`trustpolicy.json`):
	
	`{`
	`  "Version": "2012-10-17",`
	`  "Statement": [`
	`    {`
	`      "Effect": "Allow",`
	`      "Principal": {`
	`        "Service": "codebuild.amazonaws.com"`
	`      },`
		AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer

Task	Description	Skills required
	``` "Action": "sts:AssumeRole" }] } ``` Example permission policy (`permissionpolicy.json`): ``` { "Version": "2012-10-17", "Statement": [{ "Action": ["codebuild:StartBuild", "codebuild:StopBuild", "codebuild:Get*", "codebuild:List*", "codebuild:BatchGet*"], "Resource": "*", "Effect": "Allow" }, { "Action": ["logs:CreateLogGroup", ```	

Task	Description	Skills required

	``
	 "logs:CreateLogStr
	eam",
	
	 "logs:PutLogEvents"
	            ],
	            "Resource":
	 "*",
	            "Effect":
	 "Allow"
	        },
	        {
	            "Action":
	 "sts:AssumeRole",
	            "Resource":
	 "<ARN of destination
	 role>",
	            "Effect":
	 "Allow",
	            "Sid":
	 "AssumeCrossAccoun
	tArn"
	        }
	    ]
	}
	``
	Attach the managed policy `AmazonEC2ContainerRegistryReadOnly` to the CLI command as follows:
	``
	~$ aws iam attach-role-
	policy \
	--policy-arn arn:aws:i
	am::aws:policy/Ama
	zonEC2ContainerReg
	istryReadOnly \
	--role-name <name of
	 CodeBuild Role>
	``	

Task	Description	Skills required
Create a cross-account role.	In the destination AWS account, create an IAM role for the AWS CodeBuild role for the source account to assume. The cross-account role should allow container images to create a new repository and upload container images to Amazon ECR. To create the IAM role by using the AWS CLI, follow the [instructions in the IAM documentation](#). To allow the AWS CodeBuild project from the previous step, use the following trust policy: `{ "Version": "2012-10-17", "Statement": { "Effect": "Allow", "Principal": { "AWS": "<ARN of source codebuild role>" }, "Action": "sts:AssumeRole" } }` To allow the AWS CodeBuild project from the previous	AWS administrator, AWS DevOps, Cloud administrator, Cloud architect, DevOps engineer, AWS systems administrator

Task	Description	Skills required

	step to save images in the destination registry, use the following permission policy:
	
	`{`
	`    "Version":`
	` "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Action": [`
	
	` "ecr:GetDownloadUr`
	`lForLayer",`
	
	` "ecr:BatchCheckLay`
	`erAvailability",`
	
	` "ecr:PutImage",`
	
	` "ecr:InitiateLayer`
	`Upload",`
	
	` "ecr:UploadLayerPa`
	`rt",`
	
	` "ecr:CompleteLayer`
	`Upload",`
	
	` "ecr:GetRepository`
	`Policy",`
	
	` "ecr:DescribeRepos`
	`itories",`
	
	` "ecr:GetAuthorizat`
	`ionToken",`
	
	` "ecr:CreateReposit`
	`ory"`
	`            ],`	

Task	Description	Skills required

	``
	            "Resource":
	 "*",
	            "Effect":
	 "Allow"
	        }
	    ]
	}
	``	

Create the CodeBuild project

Task	Description	Skills required
Create a CodeBuild project.	Create a AWS CodeBuild project in the source account by following the [instructions in the AWS CodeBuild documentation](). The project should be in the same Region as the source registry. Configure the project as follows: • Environment type: `LINUX CONTAINER` • Service role: `CodeBuild Role` • Privileged mode: `true` • Environment image: `aws/codebuild/standard:x.x` (use the latest image available) • Environment variables: • `CROSS_ACCOUNT_ROLE_ARN` : The Amazon	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer

Task	Description	Skills required
	Resource Name (ARN) of the cross-account role • `DESTINATION_REGION` : The name of the cross-account Region • `DESTINATION_ACCOUNT` : The number of the destination account • Build specifications: Use the `buildspec.yaml` file listed in the Tools section.	

Create the event

Task	Description	Skills required
Create an events rule.	Because the pattern uses the content filtering feature, you need to create the event by using Amazon EventBridge. Create the event and target by following the instructions in the EventBridge documentation, with a few modifications: • For **Define pattern**, choose **Event Pattern**, and then choose **Custom pattern.** • Copy the following custom events pattern sample code into the text box provided: `{`	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect, DevOps engineer

Task	Description	Skills required

	``
	  "source": ["aws.ecr
	"],
	  "detail-type":
	 ["ECR Image
	 Action"],
	  "detail": {
	    "action-type":
	 ["PUSH"],
	    "result":
	 ["SUCCESS"],
	    "image-ta
	g": [{ "prefix":
	 "release-"}]
	  }
	}
	``
	• For **Select targets**, choose the AWS CodeBuild project, and paste the ARN for the AWS CodeBuild project that you created in the previous epic.
	• For **Configure Input**, choose **Input Transformer**.
	  • In the **Input Path** text box, paste:
	``
	{"IMAGE_TAG":"$.de
	tail.image-tag","R
	EPO_NAME":"$.detai
	l.repository-name"
	}
	``
	  • In the **Input Template** text box, paste:
	``
	{"environmentVaria
	blesOverride":
	 [{"name":
	``	

Task	Description	Skills required
	`"IMAGE_TAG", "value":<IMAGE_TAG>},{"name":"REPO_NAME","value":<REPO_NAME>}]}` • Choose **Use existing role**, and choose the name of the CloudWatch Events role you created previously in the *Create IAM roles* epic.	

Validate

Task	Description	Skills required
Authenticate with Amazon ECR.	Authenticate to both source and destination registries by following the steps in the [Amazon ECR documentation]().	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, DevOps engineer, Cloud architect
Test image replication.	In your source account, push a container image to a new or existing Amazon ECR source repository with an image tag prefixed with `release-`. To push the image, follow the steps in the [Amazon ECR documentation](). You can monitor the progress of the CodeBuild project in the [CodeBuild console]().	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect , DevOps engineer

Task	Description	Skills required
	After the CodeBuild project has completed successfully, sign in to the destination AWS account, open the Amazon ECR console, and confirm that the image exists in the destination Amazon ECR registry.	
Test image exclusion.	In your source account, push a container image to a new or existing Amazon ECR source repository with an image tag that doesn't have the custom prefix. Confirm that the CodeBuild project isn't started, and that no container images appear in the destination registry.	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect , DevOps engineer

Related resources

- Getting started with CodeBuild
- Getting started with Amazon EventBridge
- Content-based filtering in Amazon EventBridge event patterns
- Delegate access across AWS accounts using IAM roles
- Private image replication

Additional information

To automatically deploy the resources for this pattern, follow these steps:

1. Download the attachment and extract the two CloudFormation templates: `part-1-copy-tagged-images.yaml` and `part-2-destination-account-role.yaml`.
2. Log in to the [AWS CloudFormation console](), and deploy `part-1-copy-tagged-images.yaml` in the same AWS account and Region as the source Amazon ECR registries. Update the parameters as needed. The template deploys the following resources:
   - Amazon CloudWatch Events IAM role
   - AWS CodeBuild project IAM role
   - AWS CodeBuild project
   - AWS CloudWatch Events rule
3. Take note of the value of `SourceRoleName` in the **Outputs** tab. You will need this value in the next step.
4. Deploy the second CloudFormation template, `part-2-destination-account-role.yaml`, in the AWS account that you want to copy the Amazon ECR container images to. Update the parameters as needed. For the `SourceRoleName` parameter, specify the value from step 3. This template deploys the cross-account IAM role.
5. Validate image replication and exclusion, as described in the last step of the Epics section.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Rotate database credentials without restarting containers

Josh Joy, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, you can use AWS Secrets Manager to rotate, manage, and retrieve database credentials throughout their lifecycle. Users and applications retrieve secrets with a call to the Secrets Manager API, removing the need to hardcode sensitive information in plaintext.

If you're using containers for microservice workloads, you can securely store credentials in AWS Secrets Manager. To separate out configuration from code, these credentials are commonly injected into the container. However, it's important to rotate your credentials periodically and automatically. It's also important to support the ability to refresh credentials after revocation. At the same time, applications require the ability to rotate credentials while reducing any potential downstream availability impact.

This pattern describes how to rotate your secrets that are secured with AWS Secrets Manager within your containers without requiring your containers to restart. In addition, this pattern reduces the number of credential lookups to Secrets Manager by using the Secrets Manager client-side caching component. When you use the client-side caching component to refresh the credentials within the application, the container doesn't need to be restarted to fetch a rotated credential.

This approach works for Amazon Elastic Kubernetes Service (Amazon EKS) and Amazon Elastic Container Service (Amazon ECS).

Two scenarios are covered. In the single-user scenario, the database credential is refreshed on secret rotation by detecting the expired credential. The credential cache is instructed to refresh the secret, and then the application re-establishes the database connection. The client-side caching component caches the credential within the application and helps avoid reaching out to Secrets Manager for each credential lookup. The credential is rotated within the application without the need to force the credential refresh by restarting the container.

The second scenario rotates the secret by alternating between two users. Having two active users reduces the potential for downtime, because one user's credentials are always active. Two-user credential rotation is helpful when you have a large deployment with clusters in which there might be a small propagation delay of credential updates.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An application running in a container in Amazon EKS or Amazon ECS.
- Credentials stored in Secrets Manager, with rotation enabled.
- A second set of credentials stored in Secrets Manager, if deploying the two-user solution. Code examples can be found in the GitHub repo aws-secrets-manager-rotation-lambdas.
- An Amazon Aurora database.

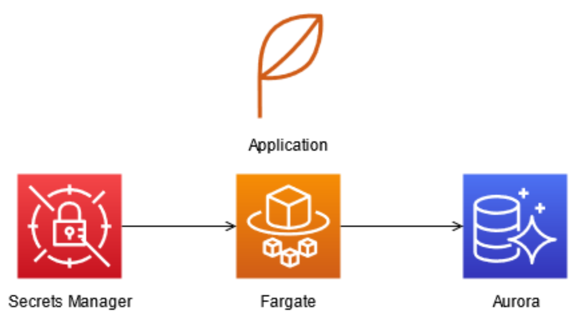
Limitations

- This example is targeted for Python applications. For Java applications, you can use the Java client-side caching component or the JDBC client-side caching library for Secrets Manager.

Architecture

Target architecture

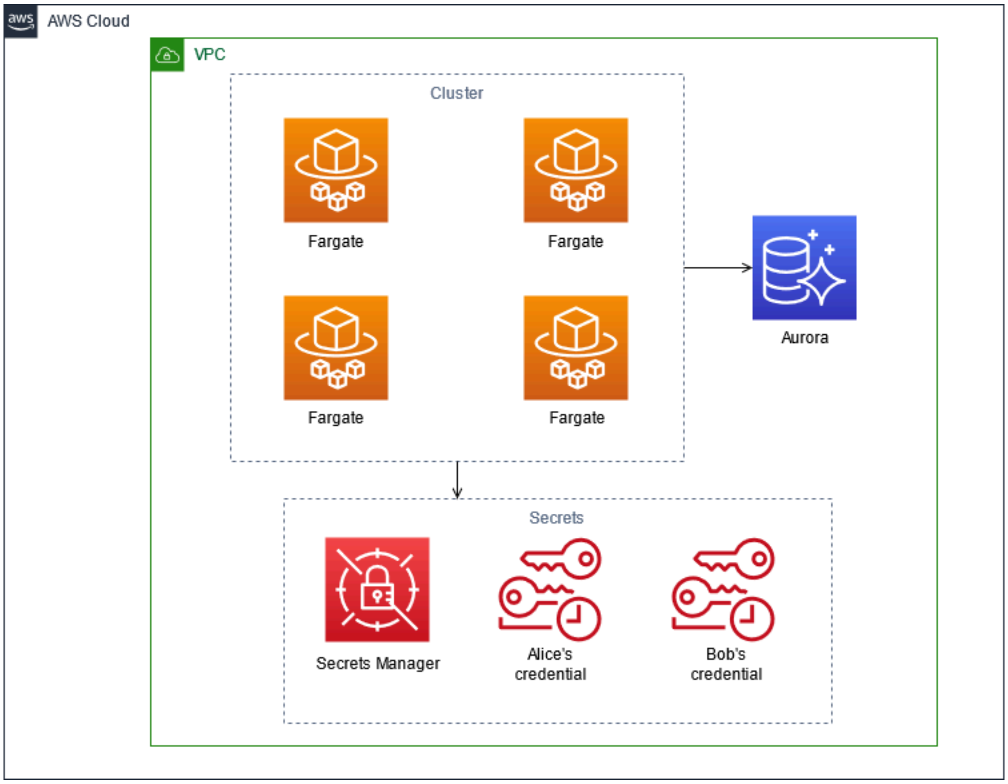
Scenario 1 – Rotation of a credential for a single user



In the first scenario, a single database credential is periodically rotated by Secrets Manager. The application container runs in Fargate. When the first database connection is established, the application container fetches the database credential for Aurora. The Secrets Manager caching component then caches the credential for future connection establishment. When rotation period has elapsed, the credential expires and the database returns an authentication error. The application then fetches the rotated credential, invalidates the cache, and updates the credential cache via the Secrets Manager client-side caching component.

In this scenario, there might be a minimal disruption while the credential is being rotated and stale connections are using the outdated credential. This concern can be addressed by using the two-user scenario.

Scenario 2 – Rotation of credentials for two users



In the second scenario, two database user credentials (Alice's and Bob's) are periodically rotated by Secrets Manager. The application container runs in a Fargate cluster. When the first database connection is established, the application container fetches the Aurora database credential for the first user (Alice). The Secrets Manager caching component then caches the credential for future connection establishment.

Although there are two users and credentials, one only active credential is managed by Secrets Manager. In this case, the caching component periodically expires and fetches the latest credential. If the Secrets Manager rotation period is longer than the cache timeout, the caching component picks up the rotated credential for the second user (Bob). For example, if the cache expiration is measured in minutes and the rotation period is measured in days, the caching component fetches the new credential as part of its periodic cache refresh. In this way, the downtime is minimized because each user's credential is active for one Secrets Manager rotation.

Automation and scale

You can use AWS CloudFormation to deploy this pattern by using infrastructure as code. This builds and creates the application container, creates the Fargate task, deploys the container into Fargate, and sets up and configure Secrets Manager with Aurora. For step-by-step deployment instructions, see the readme file.

Tools

Tools

- AWS Secrets Manager enables the replacement of hardcoded credentials, including passwords, with an API call to Secrets Manager to retrieve the secret. Because Secrets Manager can automatically rotate the secret according to a schedule, you can replace long-term secrets with short-term ones, reducing the risk of compromise.
- Docker helps developers to pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

Code

Example Python code

This pattern uses the Python client-side caching component for Secrets Manager to retrieve the authentication credentials when establishing the database connection. The client-side caching component helps avoid reaching out to Secrets Manager each time.

Now, when the rotation period elapses, the cached credential will be expired, and connecting to the database will result in an authentication error. For MySQL, the authentication error code is 1045. This example uses Amazon Aurora for MySQL, though you could use another engine such as PostgreSQL. Upon the authentication error, the database connection exception handling code catches the error. It then informs the Secrets Manager client-side caching component to refresh the secret, then to reauthenticate and re-establish the database connection. If you are using PostgreSQL or another engine, you must look up the corresponding authentication error code.

The container application can now update the database password with the rotated password without restarting the container.

Place the following code in your application code that handles database connections. This example uses Django, and it subclasses the database backend with a database wrapper for connections. If you are using a different programming language or database connection library, see your database connection library to review how to subclass database connection retrieval.

```
    def get_new_connection(self, conn_params):
        try:
            logger.info("get connection")
            databasecredentials.get_conn_params_from_secrets_manager(conn_params)
            conn =super(DatabaseWrapper,self).get_new_connection(conn_params)
            return conn
        except MySQLdb.OperationalError as e:
            error_code=e.args[0]
            if error_code!=1045:
                raise e

            logger.info("Authentication error. Going to refresh secret and try again.")
            databasecredentials.refresh_now()
            databasecredentials.get_conn_params_from_secrets_manager(conn_params)
            conn=super(DatabaseWrapper,self).get_new_connection(conn_params)
            logger.info("Successfully refreshed secret and established new database
 connection.")
            return conn
```

AWS CloudFormation and Python code

- https://github.com/aws-samples/aws-secrets-manager-credential-rotation-without-container-restart

Epics

Maintain application availability during credential rotation

Task	Description	Skills required
Install the caching component .	Download and install the Secrets Manager client-side caching component for Python. For the download link, see the *Related resources* section.	Developer
Cache the working credential.	Use the Secrets Manager client-side caching	Developer

Task	Description	Skills required
	component to cache the working credential locally.	
Update the application code to refresh the credential upon the unauthorized error from the database connection.	Update the application code to use Secrets Manager to fetch and refresh database credentials. Add the logic to handle unauthorized error codes, and then fetch the newly rotated credential. See the *Example Python code* section.	Developer

Related resources

Create a Secrets Manager secret

- Create keys in AWS KMS
- Create and manage secrets with AWS Secrets Manager

Create an Amazon Aurora cluster

- Creating an Amazon RDS DB instance

Create the Amazon ECS components

- Creating a cluster using the classic console
- Create a Docker image
- Creating a private repository
- Amazon ECR private registry
- Pushing a Docker image
- Amazon ECS task definitions
- Creating an Amazon ECS service in the classic console

Download and install the Secrets Manager client-side caching component

- Python caching client

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Run Amazon ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere

Akash Kumar, Amazon Web Services

Summary

Amazon Elastic Container Service (Amazon ECS) Anywhere supports the deployment of Amazon ECS tasks in any environment, including Amazon Web Services (AWS) managed infrastructure and customer managed infrastructure. You can do this while using a fully AWS managed control plane that's running in the cloud and always up to date.

Enterprises often use Amazon WorkSpaces for developing container-based applications. This has required Amazon Elastic Compute Cloud (Amazon EC2) or AWS Fargate with an Amazon ECS cluster to test and run ECS tasks. Now, by using Amazon ECS Anywhere, you can add Amazon WorkSpaces as external instances directly to an ECS cluster, and you can run your tasks directly. This reduces your development time, because you can test your container with an ECS cluster locally on Amazon WorkSpaces. You can also save the cost of using EC2 or Fargate instances for testing your container applications.

This pattern showcases how to deploy ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere. It sets up the ECS cluster and uses AWS Directory Service Simple AD to launch the WorkSpaces. Then the example ECS task launches NGINX in the WorkSpaces.

Prerequisites and limitations

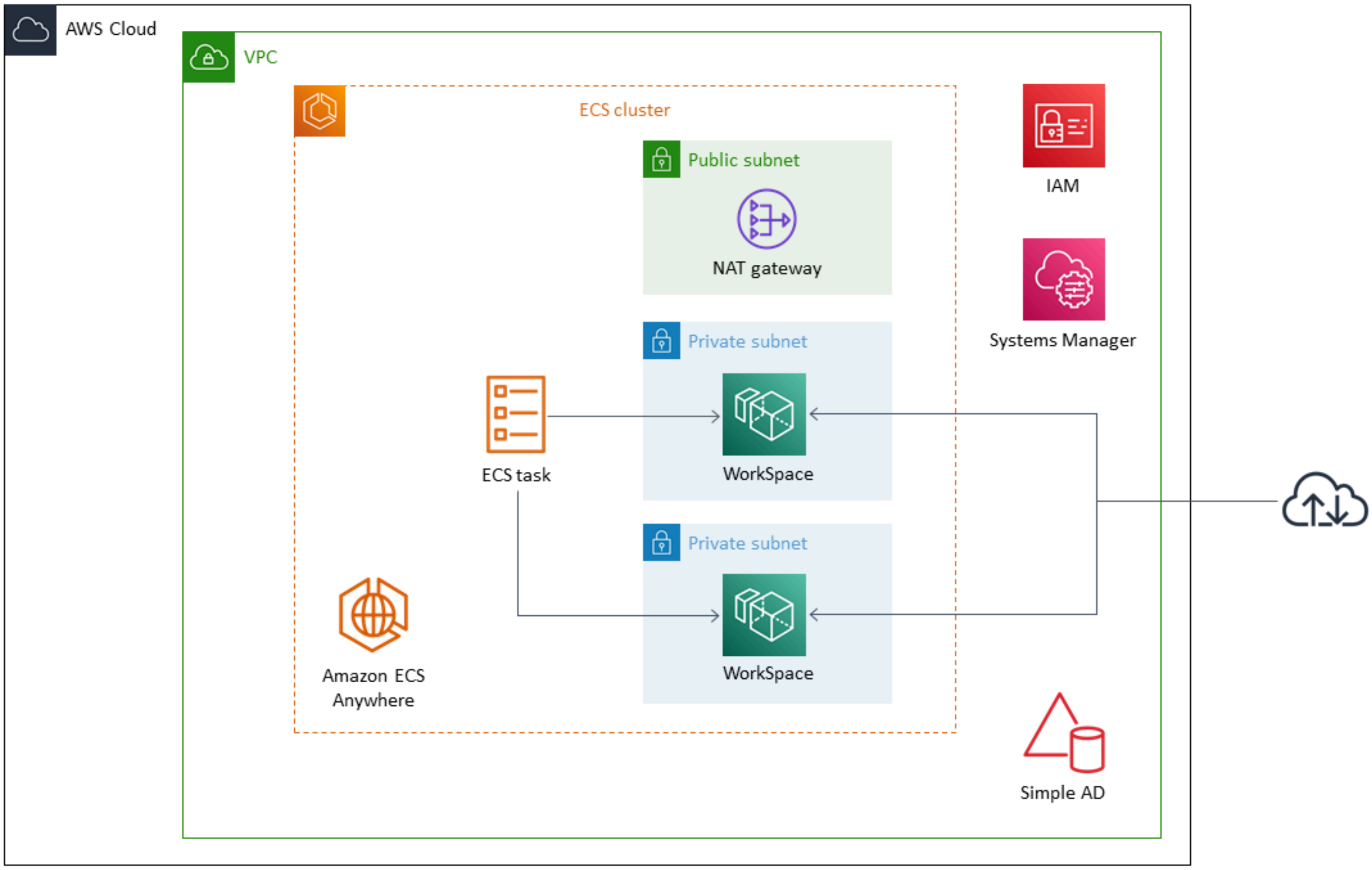
- An active AWS account
- AWS Command Line Interface (AWS CLI)
- AWS credentials configured on your machine

Architecture

Target technology stack

- A virtual private cloud (VPC)
- An Amazon ECS cluster
- Amazon WorkSpaces
- AWS Directory Service with Simple AD

Target architecture



The architecture includes the following services and resources:

- An ECS cluster with public and private subnets in a custom VPC
- Simple AD in the VPC to provide user access to Amazon WorkSpaces
- Amazon WorkSpaces provisioned in the VPC using Simple AD
- AWS Systems Manager activated for adding Amazon WorkSpaces as managed instances
- Using Amazon ECS and AWS Systems Manager Agent (SSM Agent), Amazon WorkSpaces added to Systems Manager and the ECS cluster
- An example ECS task to run in the WorkSpaces in the ECS cluster

Tools

- [AWS Directory Service Simple Active Directory (Simple AD)](#) is a standalone managed directory powered by a Samba 4 Active Directory Compatible Server. Simple AD provides a subset of the features offered by AWS Managed Microsoft AD, including the ability to manage users and to securely connect to Amazon EC2 instances.

- [Amazon Elastic Container Service (Amazon ECS)] is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.
- [AWS Identity and Access Management (IAM)] helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Systems Manager] helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.
- [Amazon WorkSpaces] helps you provision virtual, cloud-based Microsoft Windows or Amazon Linux desktops for your users, known as *WorkSpaces*. WorkSpaces eliminates the need to procure and deploy hardware or install complex software.

Epics

Set up the ECS cluster

Task	Description	Skills required
Create and configure the ECS cluster.	To create the ECS cluster, follow the instructions in the [AWS documentation], including the following steps: • For **Select cluster compatibility**, choose **Networking only**, which will support an Amazon WorkSpace as an external instance to the ECS cluster. • Choose to create a new VPC.	Cloud architect

Launch Amazon WorkSpaces

Task	Description	Skills required
Set up Simple AD and launch Amazon WorkSpaces.	To provision a Simple AD directory for your newly created VPC and launch Amazon WorkSpaces, follow the instructions in the [AWS documentation]().	Cloud architect

Set up AWS Systems Manager for a hybrid environment

Task	Description	Skills required
Download the attached scripts.	On your local machine, download the `ssm-trust-policy.json` and `ssm-activation.json` files that are in the *Attachments* section.	Cloud architect

Add the IAM role.	Add environment variables based on your business requirements.
	`export AWS_DEFAULT_REGION=${AWS_REGION_ID}`
	`export ROLE_NAME=${ECS_TASK_ROLE}`
	`export CLUSTER_NAME=${ECS_CLUSTER_NAME}`
	`export SERVICE_NAME=${ECS_CLUSTER_SERVICE_NAME}`
	Run the following command.
		Cloud architect

Task	Description	Skills required
	`aws iam create-role --role-name $ROLE_NAME --assume-role-policy-document file://ssm-trust-policy.json`	
Add the AmazonSSM ManagedInstanceCore policy to the IAM role.	Run the following command. `aws iam attach-role-policy --role-name $ROLE_NAME --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore`	Cloud architect
Add the AmazonEC2Container ServiceforEC2Role policy to IAM role.	Run the following command. `aws iam attach-role-policy --role-name $ROLE_NAME --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEC2ContainerServiceforEC2Role`	Cloud architect
Verify the IAM role.	To verify the IAM role, run the following command. `aws iam list-attached-role-policies --role-name $ROLE_NAME`	Cloud architect

Task	Description	Skills required
Activate Systems Manager.	Run the following command. `aws ssm create-activation --iam-role $ROLE_NAME | tee ssm-activation.json`	Cloud architect

Add WorkSpaces to the ECS cluster

Task	Description	Skills required
Connect to your WorkSpaces.	To connect to and set up your Workspaces, follow the instructions in the [AWS documentation]().	App developer
Download the ecs-anywhere install script.	At the command prompt, run the following command. `curl -o "ecs-anywhere-install.sh" "https://amazon-ecs-agent-packages-preview.s3.us-east-1.amazonaws.com/ecs-anywhere-install.sh" && sudo chmod +x ecs-anywhere-install.sh`	App developer
Check integrity of the shell script.	(Optional) Run the following command. `curl -o "ecs-anywhere-install.sh.sha256" "https://amazon-ecs-agent-packages-preview.s3.us-east-`	App developer

Task	Description	Skills required
	`1.amazonaws.com/ecs-anywhere-install.sh.sha256" && sha256sum -c ecs-anywhere-install.sh.sha256`	
Add an EPEL repository on Amazon Linux.	To add an Extra Packages for Enterprise Linux (EPEL) repository, run the command `sudo amazon-linux-extras install epel -y`.	App developer
Install Amazon ECS Anywhere.	To run the install script, use the following command. `sudo ./ecs-anywhere-install.sh --cluster $CLUSTER_NAME --activation-id $ACTIVATION_ID --activation-code $ACTIVATION_CODE --region $AWS_REGION`	

Task	Description	Skills required
Check instance information from the ECS cluster.	To check the Systems Manager and ECS cluster instance information and validate that WorkSpaces were added on the cluster, run the following command from your local machine. `aws ssm describe-instance-informati on" && "aws ecs list-container-instances --cluster $CLUSTER_NAME`	App developer

Add an ECS task for the WorkSpaces

Task	Description	Skills required
Create a task execution IAM role.	Download `task-execution-assume-role.json` and `external-task-definition.json` from the *Attachments* section. On your local machine, run the following command. `aws iam --region $AWS_DEFAULT_REGIO N create-role --role-name $ECS_TASK _EXECUTION_ROLE --assume-role-policy-document file://ta`	Cloud architect

Task	Description	Skills required
	```sk-execution-assume-role.json```	
Add the policy to the execution role.	Run the following command. ```aws iam --region $AWS_DEFAULT_REGION attach-role-policy --role-name $ECS_TASK_EXECUTION_ROLE --policy-arn arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy```	Cloud architect
Create a task role.	Run the following command. ```aws iam --region $AWS_DEFAULT_REGION create-role --role-name $ECS_TASK_EXECUTION_ROLE --assume-role-policy-document file://task-execution-assume-role.json```	Cloud architect
Register the task definition to the cluster.	On your local machine, run the following command. ```aws ecs register-task-definition --cli-input-json file://external-task-definition.json```	Cloud architect

Task	Description	Skills required
Run the task.	On your local machine, run the following command. `aws ecs run-task --cluster $CLUSTER_NAME --launch-type EXTERNAL --task-definition nginx`	Cloud architect

Validate the task running state.	To fetch the task ID, run the following command.
	`export TEST_TASKID=$(aws ecs list-tasks --cluster $CLUSTER_NAME | jq -r '.taskArns[0]')`
	With the task ID, run the following command.
	`aws ecs describe-tasks --cluster $CLUSTER_NAME --tasks ${TEST_TASKID}`
		Cloud architect

Verify the task on the WorkSpace.	To check that NGINX is running on the WorkSpace, run the command `curl http://localhost:8080`.	App developer

Related resources

- ECS clusters
- Setting up a hybrid environment

- [Amazon WorkSpaces]
- [Simple AD]

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance

Vijai Anand Ramalingam and Sreelaxmi Pai, Amazon Web Services

Summary

This pattern is for people who are starting to containerize their applications on the Amazon Web Services (AWS) Cloud. When you begin to containerize apps on cloud, usually there are no container orchestrating platforms set up. This pattern helps you quickly set up infrastructure on AWS to test your containerized applications without needing an elaborate container orchestrating infrastructure.

The first step in the modernization journey is to transform the application. If it's a legacy .NET Framework application, you must first change the runtime to ASP.NET Core. Then do the following:

- Create the Docker container image
- Run the Docker container using the built image
- Validate the application before deploying it on any container orchestration platform, such as Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS).

This pattern covers the build, run, and validate aspects of modern application development on an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- An AWS Identity and Access Management (IAM) role with sufficient access to create AWS resources for this pattern
- Visual Studio Community 2022 or later downloaded and installed
- A .NET Framework project modernized to ASP.NET Core
- A GitHub repository

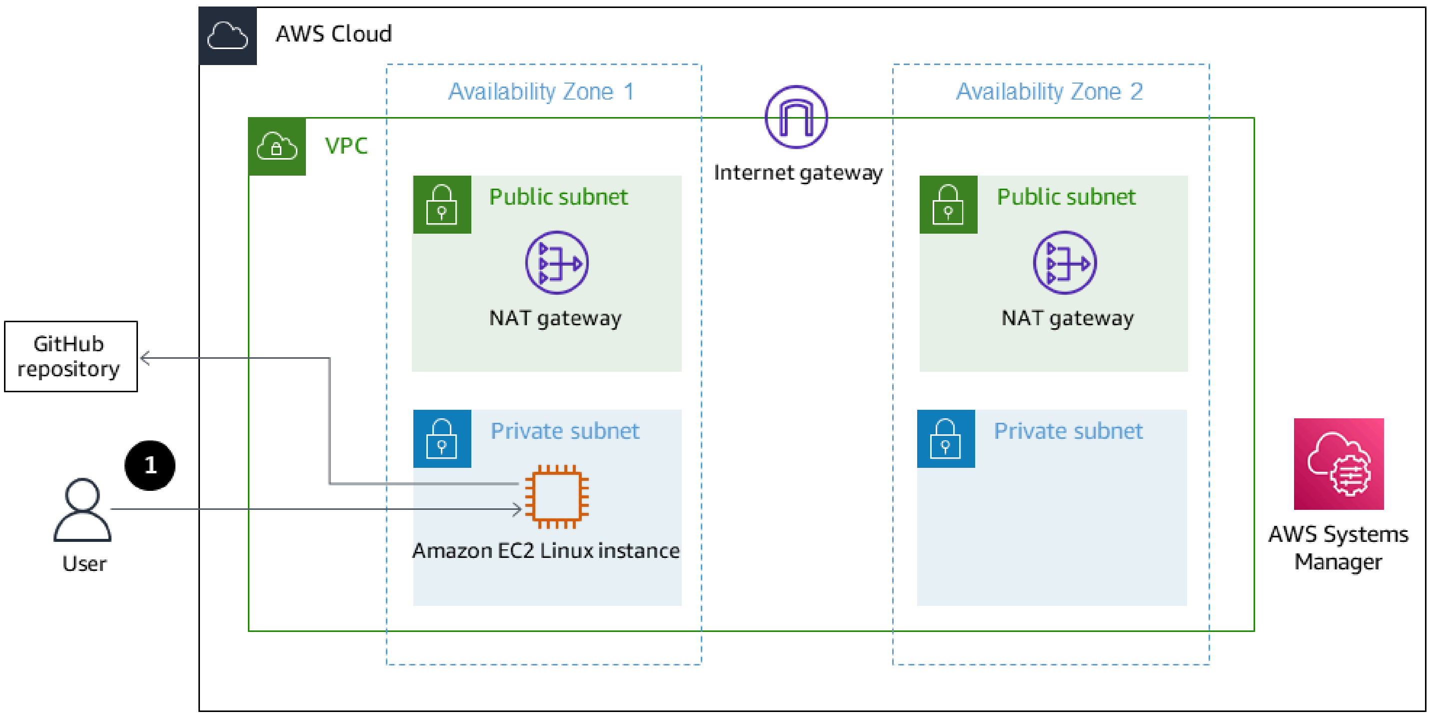
Product versions

- Visual Studio Community 2022 or later

Architecture

Target architecture

This pattern uses an [AWS CloudFormation template](#) to create the highly available architecture shown in the following diagram. An Amazon EC2 Linux instance is launched in a private subnet. AWS Systems Manager Session Manager is used to access the private Amazon EC2 Linux instance and to test the API running in the Docker container.



1. Access to the Linux instance through Session Manager

Tools

AWS services

- [AWS Command Line Interface](#) – AWS Command Line Interface (AWS CLI) is an open source tool for interacting with AWS services through commands in your command line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console.

- AWS Management Console – The AWS Management Console is a web application that comprises and refers to a broad collection of service consoles for managing AWS resources. When you first sign in, you see the console home page. The home page provides access to each service console and offers a single place to access the information you need to perform your AWS related tasks.
- AWS Systems Manager Session Manager – Session Manager is a fully managed AWS Systems Manager capability. With Session Manager, you can manage your Amazon Elastic Compute Cloud (Amazon EC2) instances. Session Manager provides secure and auditable node management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys.

Other tools

- Visual Studio 2022 – Visual Studio 2022 is an integrated development environment (IDE).
- Docker – Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.

Code

```
FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base
 WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /src
COPY ["DemoNetCoreWebAPI/DemoNetCoreWebAPI.csproj", "DemoNetCoreWebAPI/"]
RUN dotnet restore "DemoNetCoreWebAPI/DemoNetCoreWebAPI.csproj"
COPY . .
WORKDIR "/src/DemoNetCoreWebAPI"
RUN dotnet build "DemoNetCoreWebAPI.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "DemoNetCoreWebAPI.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "DemoNetCoreWebAPI.dll"]
```

Epics

Develop the ASP.NET Core web API

Task	Description	Skills required
Create an example ASP.NET Core web API using Visual Studio.	To create an example ASP.NET Core web API, do the following: 1. Open Visual Studio 2022. 2. Choose **Create a new project**. 3. Select the **ASP.NET Core Web API** project template, and choose **Next**. 4. For the project name, enter **DemoNetCoreWebAPI**, and choose **Next**. 5. Choose **Create**. 6. To run the project locally, press F5. 7. Verify that the default **WeatherForecast** API endpoint is returning the results using [Swagger](https://swagger.io/). 8. Open the command prompt, navigate to the .csproj project folder, and run the following commands to push the new web API to your GitHub repository. `git add --all`	App developer

Task	Description	Skills required
	```git commit -m "Initial Version"``` ```git push```	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a Dockerfile.</td><td>To create a Dockerfile, do one of the following:
Create the Dockerfile manually using the sample Dockerfile in the Code section. Based on the requirements, select the appropriate .NET base image. For information about .NET and ASP.NET Core related images, see Docker hub.
Create the Dockerfile using Visual Studio and Docker Desktop. In the solution explorer, right click on the project, choose Add->Docker Support. For Target OS, select Linux. Ensure that the new Dockerfile is in the same path as the solution file (.sln).
To push the changes to your GitHub repository, run the following command.<pre>git add --all
git commit -m "Dockerfile added"
git push</pre></td><td>App developer</td></tr>
</table>

Set up the Amazon EC2 Linux instance

Task	Description	Skills required
Set up the infrastructure.	Launch the AWS CloudFormation template to create the infrastructure, which includes the following: • A virtual private cloud (VPC), using the AWS VPC Quick Start, with two public and two private subnets spanning two Availability Zones. • The required IAM role to enable AWS Systems Manager. • In one of the private subnets, an Amazon Linux 2 demo instance with the latest SSM Agent. Although this instance doesn't have any direct connectivity from the internet, it can be accessed securely by using AWS Systems Manager Session Manager without requiring a bastion host. **Note** Amazon Linux 2 is nearing end of support. For more information, see	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required
	the Amazon Linux 2 FAQs. To learn more about accessing a private Amazon EC2 instance using Session Manager without requiring a bastion host, see the Toward a bastion-less world blog post.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Log in to the Amazon EC2 Linux instance.</td><td>To connect to the Amazon EC2 Linux instance in the private subnet, do the following:

1. Open the Amazon EC2 console.
2. In the navigation pane, choose Instances.
3. Select the Amazon Linux 2 demo instance, and choose Connect.

Note
Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs.

4. Choose Session Manager.
5. Choose Connect to open a new terminal window.
6. Run the following command.

<code>sudo su</code></td><td>App developer</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install and start Docker.</td><td>To install and start Docker in the Amazon EC2 Linux instance, do the following:

1. To install Docker, run the following command.
<pre><code>yum install -y docker</code></pre>2. To start the Docker service, run the following command.
<pre><code>service docker start</code></pre>3. To verify Docker installat ion, run the following command.
<pre><code>docker info</code></pre></td><td>App developer, AWS administrator, AWS DevOps</td></tr>
</table>

Task	Description	Skills required

Install Git and clone the repository.	To install Git on the Amazon EC2 Linux instance and clone the repository from GitHub, do the following.
	
	1. To install Git, run the following command.
	
	`yum install git -y`
	
	2. To clone the repository, run the following command.
	
	`git clone https://github.com/<username>/<repo-name>.git`
	
	3. To navigate to the Dockerfile, run the following command.
	
	`cd <repo-name>/DemoNetCoreWebAPI/`
		App developer, AWS administrator, AWS DevOps

Task	Description	Skills required

Build and run the Docker container.	To build the Docker image and run the container inside the Amazon EC2 Linux instance, do the following:
	
	1. To create the Docker image, run the following command.
	`docker build -t aspnetcorewebapiimage -f Dockerfile .`
	2. To view all the Docker images, run the following command.
	`docker images`
	3. To create and run the container, run the following command.
	`docker run -d -p 80:80 --name aspnetcorewebapicontainer aspnetcorewebapiimage`
		App developer, AWS administrator, AWS DevOps

Test the web API

Task	Description	Skills required
Test the web API using the curl command.	To test the web API, run the following command.	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

```
curl -X GET "http://l
ocalhost/WeatherFo
recast" -H  "accept:
 text/plain"
```

Verify the API response.

Note

You can get the curl commands for each endpoint from Swagger when you are running it locally.
</td><td></td></tr>
</table>

Clean up resources

Task	Description	Skills required
Delete all resources.	Delete the stack to remove all the resources. This ensures that you aren’t charged for any services that you aren’t using.	AWS administrator, AWS DevOps

Related resources

- Connect to your Linux instance from Windows using PuTTY
- Create a web API with ASP.NET Core
- Toward a bastion-less world

Run message-driven workloads at scale by using AWS Fargate

Stan Zubarev, Amazon Web Services

Summary

This pattern shows how to run message-driven workloads at scale in the AWS Cloud by using containers and AWS Fargate.

Using containers to process data can be helpful when the amount of data an application processes exceeds the limitations of function-based serverless compute services. For example, if an application requires more compute capacity or processing time than what AWS Lambda offers, using Fargate can improve performance.

The following example setup uses the [AWS Cloud Development Kit (AWS CDK) in TypeScript] to configure and deploy the following resources in the AWS Cloud:

- A Fargate service
- An Amazon Simple Queue Service (Amazon SQS) queue
- An Amazon DynamoDB table.
- An Amazon CloudWatch dashboard

The Fargate service receives and processes messages from the Amazon SQS queue, then stores them in the Amazon DynamoDB table. You can monitor how many Amazon SQS messages are processed and how many DynamoDB items are created by Fargate by using the CloudWatch dashboard.

Note

You can also use this pattern's example code to build more complex data processing workloads in event-driven serverless architectures. For more information, see [Run event-driven and scheduled workloads at scale with AWS Fargate].

Prerequisites and limitations

Prerequisites

- An active AWS account

- The latest version of the [AWS Command Line Interface (AWS CLI)](), installed and configured on your local machine
- [Git](), installed and configured on your local machine
- The [AWS CDK](), installed and configured on your local machine
- [Go](), installed and configured on your local machine
- [Docker](), installed and configured on your local machine

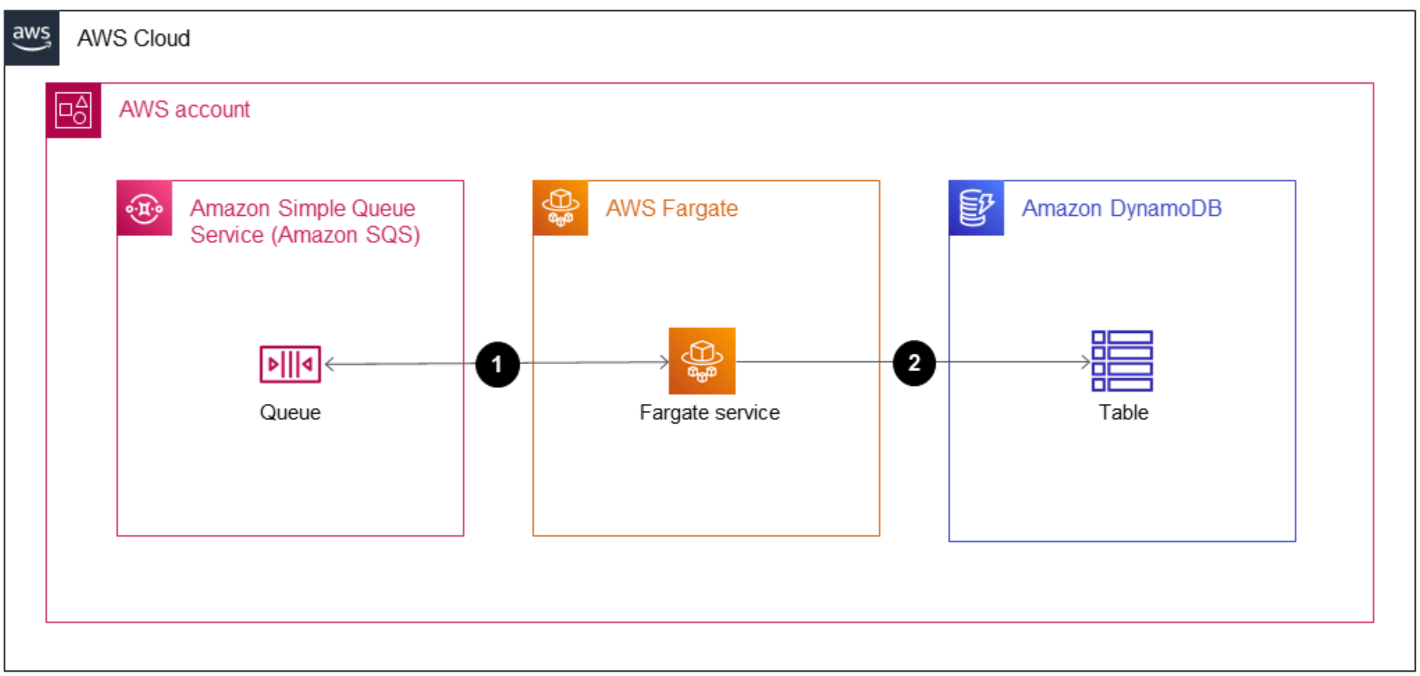
Architecture

Target technology stack

- Amazon SQS
- AWS Fargate
- Amazon DynamoDB

Target architecture

The following diagram shows an example workflow for running message-driven workloads at scale in the AWS Cloud by using Fargate:



The diagram shows the following workflow:

1. The Fargate service uses Amazon SQS long polling to receive messages from an Amazon SQS queue.
2. The Fargate service then processes the Amazon SQS messages and stores them in a DynamoDB table.

Automation and scale

To automate scaling your Fargate task count, you can configure Amazon Elastic Container Service (Amazon ECS) Service Auto Scaling. It's a best practice to configure the scaling policy based on the number of visible messages in your application's Amazon SQS queue.

For more information, see Scaling based on Amazon SQS in the *Amazon EC2 Auto Scaling User Guide*.

Tools

AWS services

- AWS Fargate helps you run containers without needing to manage servers or Amazon Elastic Compute Cloud (Amazon EC2) instances. It's used in conjunction with Amazon Elastic Container Service (Amazon ECS).
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.

Code

The code for this pattern is available in the GitHub sqs-fargate-ddb-cdk-go repository.

Epics

Create and deploy the resources by using the AWS CDK

Task	Description	Skills required
Clone the GitHub repository.	Clone the GitHub [sqs-fargate-ddb-cdk-go](https://github.com/aws-samples/sqs-fargate-ddb-cdk-go) repository to your local machine by running the following command: `git clone https://github.com/aws-samples/sqs-fargate-ddb-cdk-go.git`	App developer
Verify that the AWS CLI is configured to the correct AWS account and that the AWS CDK has the required permissions.	To check if your AWS CLI configuration settings are correct, you can run the following Amazon Simple Storage Service (Amazon S3) ls command: `aws s3 ls` This procedure also requires the AWS CDK to have permissions to provision infrastructure within your AWS account. To grant the required permissions, you must create named AWS profile in AWS CLI and export it as an **AWS_PROFILE** environment variable.	App developer

Task	Description	Skills required
	Note If you haven't used the AWS CDK in your AWS account before, you must first provision the required AWS CDK resources . For more informati on, see [Bootstrapping](#) in the *AWS CDK v2 Developer Guide*.	

Task	Description	Skills required

Deploy the AWS CDK stack to your AWS account.	1. Build a container image by running the following AWS CLI command:
	`docker build -t go-fargate .`
	2. Open the AWS CDK directory by running the following command:
	`cd cdk`
	3. Install the required npm modules by running the following command:
	`npm i`
	4. Deploy the AWS CDK pattern to your AWS account by running the following command:
	`cdk deploy --profile ${AWS_PROFILE}`
		App developer

Test the setup

Task	Description	Skills required
Send a test message to the Amazon SQS queue.	For instructions, see Sending messages to a queue (console) in the *Amazon SQS Developer Guide*. **Test Amazon SQS message example**	App developer

Task	Description	Skills required

	``
	{
	    "message": "hello,
	 Fargate"
	}
	``	

Verify that the test message appears in the Fargate service's CloudWatch logs.	Follow the instructions in Viewing CloudWatch Logs in the *Amazon ECS Developer Guide*. Make sure that you review the logs for the **go-fargate-service** log group in the **go-service-cluster** ECS cluster.	App developer
Verify that the test message appears in the DynamoDB table.	1. Open the DynamoDB console. 2. In the left navigation pane, choose Tables. Then, select the following table from the list: **sqs-fargate-ddb-table**. 3. Choose **Explore table items**. 4. Verify that the test message appears in the **Items returned** list.	App developer

Task	Description	Skills required
Verify that the Fargate service is sending messages to CloudWatch Logs.	1. Open the [CloudWatch console](https://console.aws.amazon.com/cloudwatch/). 2. In the left navigation pane, choose **Dashboards**. 3. In the Custom Dashboard s list, select the dashboard named **go-service-dashboard**. 4. Verify that the test message appears in the logs. **Note** The AWS CDK creates the CloudWatc h dashboard in your AWS account automatically.	App developer

Clean up

Task	Description	Skills required
Delete the AWS CDK stack.	1. Open your AWS CDK directory in the AWS CLI by running the following command: `cd cdk`	App developer

Task	Description	Skills required
	2. Delete the AWS CDK stack by running the following command: `cdk destroy --profile ${AWS_PROFILE}`	
Verify that the AWS CDK stack is deleted.	To make sure that the stack was deleted, run the following command: `aws cloudformation list-stacks --query \ "StackSummaries[?contains(StackName,'SqsFargate')].StackStatus" \ --profile ${AWS_PROFILE}` The `StackStatus` value returned in the command output is `DELETE_COMPLETE` if the stack is deleted. For more information, see [Examples of CloudFormation stack operation commands for the AWS CLI and PowerShell]() in the *AWS CloudFormation User Guide*.	App developer

Related resources

- Configuring AWS CLI (*AWS CLI User Guide for Version 2*)
- API reference (*AWS CDK API Reference*)
- AWS SDK for Go v2 (Go documentation)

Run stateful workloads with persistent data storage by using Amazon EFS on Amazon EKS with AWS Fargate

Ricardo Morais, Rodrigo Bersa, and Lucio Pereira, Amazon Web Services

Summary

This pattern provides guidance for enabling Amazon Elastic File System (Amazon EFS) as a storage device for containers that are running on Amazon Elastic Kubernetes Service (Amazon EKS) by using AWS Fargate to provision your compute resources.

The setup described in this pattern follows security best practices and provides security at rest and security in transit by default. To encrypt your Amazon EFS file system, it uses an AWS Key Management Service (AWS KMS) key, but you can also specify a key alias that dispatches the process of creating a KMS key.

You can follow the steps in this pattern to create a namespace and Fargate profile for a proof-of-concept (PoC) application, install the Amazon EFS Container Storage Interface (CSI) driver that is used to integrate the Kubernetes cluster with Amazon EFS, configure the storage class, and deploy the PoC application. These steps result in an Amazon EFS file system that is shared among multiple Kubernetes workloads, running over Fargate. The pattern is accompanied by scripts that automate these steps.

You can use this pattern if you want data persistence in your containerized applications and want to avoid data loss during scaling operations. For example:

- **DevOps tools** – A common scenario is to develop a continuous integration and continuous delivery (CI/CD) strategy. In this case, you can use Amazon EFS as a shared file system to store configurations among different instances of the CI/CD tool or to store a cache (for example, an Apache Maven repository) for pipeline stages among different instances of the CI/CD tool.
- **Web servers** – A common scenario is to use Apache as an HTTP web server. You can use Amazon EFS as a shared file system to store static files that are shared among different instances of the web server. In this example scenario, modifications are applied directly to the file system instead of static files being baked into a Docker image.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An existing Amazon EKS cluster with Kubernetes version 1.17 or later (tested up to version 1.27)
- An existing Amazon EFS file system to bind a Kubernetes StorageClass and provision file systems dynamically
- Cluster administration permissions
- Context configured to point to the desired Amazon EKS cluster

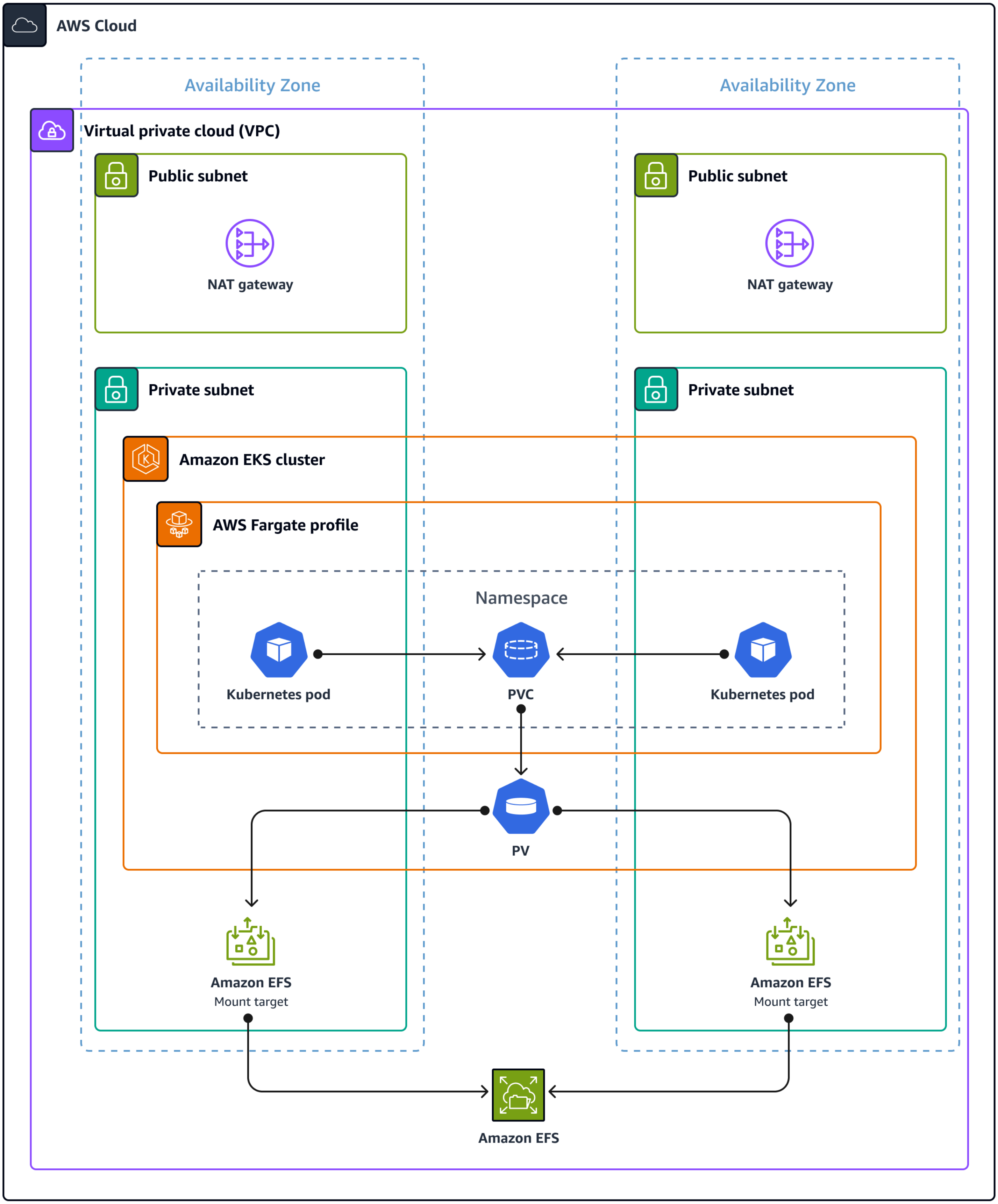
Limitations

- There are some limitations to consider when you're using Amazon EKS with Fargate. For example, the use of some Kubernetes constructs, such as DaemonSets and privileged containers, aren't supported. For more information, about Fargate limitations, see the AWS Fargate considerations in the Amazon EKS documentation.
- The code provided with this pattern supports workstations that are running Linux or macOS.

Product versions

- AWS Command Line Interface (AWS CLI) version 2 or later
- Amazon EFS CSI driver version 1.0 or later (tested up to version 2.4.8)
- eksctl version 0.24.0 or later (tested up to version 0.158.0)
- jq version 1.6 or later
- kubectl version 1.17 or later (tested up to version 1.27)
- Kubernetes version 1.17 or later (tested up to version 1.27)

Architecture



The target architecture is comprised of the following infrastructure:

- A virtual private cloud (VPC)
- Two Availability Zones
- A public subnet with a NAT gateway that provides internet access
- A private subnet with an Amazon EKS cluster and Amazon EFS mount targets (also known as *mount points*)
- Amazon EFS at the VPC level

The following is the environment infrastructure for the Amazon EKS cluster:

- AWS Fargate profiles that accommodate the Kubernetes constructs at the namespace level
- A Kubernetes namespace with:
  - Two application pods distributed across Availability Zones
  - One persistent volume claim (PVC) bound to a persistent volume (PV) at the cluster level
- A cluster-wide PV that is bound to the PVC in the namespace and that points to the Amazon EFS mount targets in the private subnet, outside of the cluster

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that you can use to interact with AWS services from the command line.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud. In this pattern, it provides a simple, scalable, fully managed, and shared file system for use with Amazon EKS.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or operate your own clusters.
- AWS Fargate is a serverless compute engine for Amazon EKS. It creates and manages compute resources for your Kubernetes applications.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- eksctl is a command-line utility for creating and managing Kubernetes clusters on Amazon EKS.
- kubectl is a command-line interface that helps you run commands against Kubernetes clusters.
- jq is a command-line tool for parsing JSON.

Code

The code for this pattern is provided in the GitHub Persistence Configuration with Amazon EFS on Amazon EKS using AWS Fargate repo. The scripts are organized by epic, in the folders `epic01` through `epic06`, corresponding to the order in the Epics section in this pattern.

Best practices

The target architecture includes the following services and components, and it follows AWS Well-Architected Framework best practices:

- Amazon EFS, which provides a simple, scalable, fully managed elastic NFS file system. This is used as a shared file system among all replications of the PoC application that are running in pods, which are distributed in the private subnets of the chosen Amazon EKS cluster.
- An Amazon EFS mount target for each private subnet. This provides redundancy per Availability Zone within the virtual private cloud (VPC) of the cluster.
- Amazon EKS, which runs the Kubernetes workloads. You must provision an Amazon EKS cluster before you use this pattern, as described in the Prerequisites section.
- AWS KMS, which provides encryption at rest for the content that's stored in the Amazon EFS file system.
- Fargate, which manages the compute resources for the containers so that you can focus on business requirements instead of infrastructure burden. The Fargate profile is created for all private subnets. It provides redundancy per Availability Zone within the virtual private cloud (VPC) of the cluster.
- Kubernetes Pods, for validating that content can be shared, consumed, and written by different instances of an application.

Epics

Provision an Amazon EKS cluster (optional)

Task	Description	Skills required
Create an Amazon EKS cluster.	**Note** If you already have a cluster deployed, skip to the next epic. Create an Amazon EKS cluster in your existing AWS account. In the [GitHub directory](), use one of the patterns to deploy an Amazon EKS cluster by using Terraform or eksctl. For more information, see [Creating an Amazon EKS cluster]() in the Amazon EKS documentation. In the Terraform pattern, there are also examples showing how to: link Fargate profiles to your Amazon EKS cluster, create an Amazon EFS file system, and deploy Amazon EFS CSI driver in your Amazon EKS cluster.	AWS administrator, Terraform or eksctl administrator, Kubernetes administrator

Task	Description	Skills required

Export environment variables.	Run the env.sh script. This provides the information required in the next steps.
	
	`source ./scripts/env.sh`
	`Inform the AWS Account ID:`
	`<13-digit-account-id>`
	`Inform your AWS Region:`
	`<aws-Region-code>`
	`Inform your Amazon EKS Cluster Name:`
	`<amazon-eks-cluster-name>`
	`Inform the Amazon EFS Creation Token:`
	`<self-genereated-uuid>`
	
	If not noted yet, you can get all the information requested above with the following CLI commands.
	
	`# ACCOUNT ID`
	`aws sts get-caller-identity --query "Account" --output text`
	
	`# REGION CODE`
	`aws configure get region`
	
	`# CLUSTER EKS NAME`
	`aws eks list-clusters --query "clusters" --output text`
		AWS systems administrator

Task	Description	Skills required
	``` # GENERATE EFS TOKEN uuidgen ```	

Create a Kubernetes namespace and a linked Fargate profile

Task	Description	Skills required

Create a Kubernetes namespace and Fargate profile for application workloads.	Create a namespace for receiving the application workloads that interact with Amazon EFS. Run the `create-k8s-ns-and-linked-fargate-profile.sh` script. You can choose to use a custom namespace name or the default provided namespace `poc-efs-eks-fargate` .
	
	With a custom application namespace name:
	
	`export $APP_NAMESPACE=<CUSTOM_NAME>`
	`./scripts/epic01/create-k8s-ns-and-linked-fargate-profile.sh \`
	`-c "$CLUSTER_NAME" -n "$APP_NAMESPACE"`
	
	Without a custom application namespace name:
		Kubernetes user with granted permissions

Task	Description	Skills required
	`./scripts/epic01/create-k8s-ns-and-linked-fargate-profile.sh \ -c "$CLUSTER_NAME"` where `$CLUSTER_NAME` is the name of your Amazon EKS cluster. The `-n <NAMESPACE>` parameter is optional; if not informed, a default generated namespace name will be provided.	

Create an Amazon EFS file system

Task	Description	Skills required
Generate a unique token.	Amazon EFS requires a creation token to ensure idempotent operation (calling the operation with the same creation token has no effect). To meet this requirement, you must generate a unique token through an available technique. For example, you can generate a universally unique identifier (UUID) to use as a creation token.	AWS systems administrator
Create an Amazon EFS file system.	Create the file system for receiving the data files that are read and written by the application workloads. You	AWS systems administrator

Task	Description	Skills required
	can create an encrypted or non-encrypted file system. (As a best practice, the code for this pattern creates an encrypted system to enable encryption at rest by default.) You can use a unique, symmetric AWS KMS key to encrypt your file system. If a custom key is not specified, an AWS managed key is used. Use the create-efs.sh script to create an encrypted or non-encrypted Amazon EFS file system, after you generate a unique token for Amazon EFS. **With encryption at rest, without a KMS key:** `./scripts/epic02/create-efs.sh \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN"` where `$CLUSTER_NAME` is the name of your Amazon EKS cluster and `$EFS_CREATION_TOKEN` is a unique creation token for the file system.	

Task	Description	Skills required

	With encryption at rest, with a KMS key:
	`./scripts/epic02/create-efs.sh \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN" \ -k "$KMS_KEY_ALIAS"`
	where `$CLUSTER_NAME` is the name of your Amazon EKS cluster, `$EFS_CREATION_TOKEN` is a unique creation token for the file system, and `$KMS_KEY_ALIAS` is the alias for the KMS key.
	Without encryption:
	`./scripts/epic02/create-efs.sh -d \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN"`
	where `$CLUSTER_NAME` is the name of your Amazon EKS cluster, `$EFS_CREATION_TOKEN` is a unique creation token for the file system, and –d disables encryption at rest.	

Task	Description	Skills required
Create a security group.	Create a security group to allow the Amazon EKS cluster to access the Amazon EFS file system.	AWS systems administrator
Update the inbound rule for the security group.	Update the inbound rules of the security group to allow incoming traffic for the following settings: • **TCP protocol** – port 2049 • **Source** – CIDR block ranges for the private subnets in the VPC that contains the Kubernetes cluster	AWS systems administrator
Add a mount target for each private subnet.	For each private subnet of the Kubernetes cluster, create a mount target for the file system and the security group.	AWS systems administrator

Install Amazon EFS components into the Kubernetes cluster

Task	Description	Skills required
Deploy the Amazon EFS CSI driver.	Deploy the Amazon EFS CSI driver into the cluster. The driver provisions storage according to persisten t volume claims created by applications. Run the `create-k8s-efs-csi-sc.sh` script to deploy the Amazon EFS CSI driver and	Kubernetes user with granted permissions

Task	Description	Skills required
	the storage class into the cluster. `./scripts/epic03/create-k8s-efs-csi-sc.sh` This script uses the `kubectl` utility, so make sure that the context has been configured and point to the desired Amazon EKS cluster.	
Deploy the storage class.	Deploy the storage class into the cluster for the Amazon EFS provisioner (efs.csi.aws.com).	Kubernetes user with granted permissions

Install the PoC application into the Kubernetes cluster

Task	Description	Skills required
Deploy the persistent volume.	Deploy the persistent volume, and link it to the created storage class and to the ID of the Amazon EFS file system. The application uses the persistent volume to read and write content. You can specify any size for the persistent volume in the storage field. Kubernetes requires this field, but because Amazon EFS is an elastic file system, it does not enforce any file	Kubernetes user with granted permissions

Task	Description	Skills required

	system capacity. You can deploy the persistent volume with or without encryption. (The Amazon EFS CSI driver enables encryption by default, as a best practice.) Run the `deploy-poc-app.sh` script to deploy the persistent volume, the persistent volume claim, and the two workloads.
	
	With encryption in transit:
	
	`./scripts/epic04/deploy-poc-app.sh \ -t "$EFS_CREATION_TOKEN"`
	
	where `$EFS_CREATION_TOKEN` is the unique creation token for the file system.
	
	Without encryption in transit:
	
	`./scripts/epic04/deploy-poc-app.sh -d \ -t "$EFS_CREATION_TOKEN"`
	
	where `$EFS_CREATION_TOKEN` is the unique creation token for the file	

Task	Description	Skills required
	system, and `-d` disables encryption in transit.	
Deploy the persistent volume claim requested by the application.	Deploy the persistent volume claim requested by the application, and link it to the storage class. Use the same access mode as the persistent volume you created previousl y. You can specify any size for the persistent volume claim in the storage field. Kubernetes requires this field, but because Amazon EFS is an elastic file system, it does not enforce any file system capacity.	Kubernetes user with granted permissions
Deploy workload 1.	Deploy the pod that represent s workload 1 of the applicati on. This workload writes content to the file `/data/out1.txt` .	Kubernetes user with granted permissions
Deploy workload 2.	Deploy the pod that represent s workload 2 of the applicati on. This workload writes content to the file `/data/out2.txt` .	Kubernetes user with granted permissions

Validate file system persistence, durability, and shareability

Task	Description	Skills required
Check the status of the `PersistentVolume` .	Enter the following command to check the status of the `PersistentVolume` . `kubectl get pv` For an example output, see the Additional information section.	Kubernetes user with granted permissions
Check the status of the `PersistentVolumeClaim` .	Enter the following command to check the status of the `PersistentVolumeClaim` . `kubectl -n poc-efs-eks-fargate get pvc` For an example output, see the Additional information section.	Kubernetes user with granted permissions
Validate that workload 1 can write to the file system.	Enter the following command to validate that workload 1 is writing to `/data/out1.txt` . `kubectl exec -ti poc-app1 -n poc-efs-eks-fargate -- tail -f /data/out1.txt` The results are similar to the following:	Kubernetes user with granted permissions

Task	Description	Skills required

	`...`
	`Thu Sep  3 15:25:07 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:25:12 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:25:17 UTC 2023 - PoC APP 1`
	`...`	

Validate that workload 2 can write to the file system.	Enter the following command to validate that workload 2 is writing to `/data/out2.txt` .
	`kubectl -n $APP_NAMESPACE exec -ti poc-app2 -- tail -f /data/out2.txt`
	The results are similar to the following:
	`...`
	`Thu Sep  3 15:26:48 UTC 2023 - PoC APP 2`
	`Thu Sep  3 15:26:53 UTC 2023 - PoC APP 2`
	`Thu Sep  3 15:26:58 UTC 2023 - PoC APP 2`
	`...`
		Kubernetes user with granted permissions

Task	Description	Skills required

Validate that workload 1 can read the file written by workload 2.	Enter the following command to validate that workload 1 can read the `/data/out2.txt` file written by workload 2.
	`kubectl exec -ti poc-app1 -n poc-efs-eks-fargate -- tail -n 3 /data/out2.txt`
	The results are similar to the following:
	`...`
	`Thu Sep  3 15:26:48 UTC 2023 - PoC APP 2`
	`Thu Sep  3 15:26:53 UTC 2023 - PoC APP 2`
	`Thu Sep  3 15:26:58 UTC 2023 - PoC APP 2`
	`...`
		Kubernetes user with granted permissions

Task	Description	Skills required

Validate that workload 2 can read the file written by workload 1.	Enter the following command to validate that workload 2 can read the `/data/out1.txt` file written by workload 1.
	
	`kubectl -n $APP_NAMESPACE exec -ti poc-app2 -- tail -n 3 /data/out1.txt`
	
	The results are similar to the following:
	
	`...`
	`Thu Sep  3 15:29:22 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:29:27 UTC 2023 - PoC APP 1`
	`Thu Sep  3 15:29:32 UTC 2023 - PoC APP 1`
	`...`
		Kubernetes user with granted permissions

Task	Description	Skills required

Validate that files are retained after you remove application components.	Next, you use a script to remove the application components (persistent volume, persistent volume claim, and pods), and validate that the files `/data/out1.txt` and `/data/out2.txt` are retained in the file system. Run the `validate-efs-content.sh` script by using the following command.
	`./scripts/epic05/validate-efs-content.sh \ -t "$EFS_CREATION_TOKEN"`
	where `$EFS_CREATION_TOKEN` is the unique creation token for the file system.
	The results are similar to the following:
	`pod/poc-app-validation created`
	`Waiting for pod get Running state...`
	`Waiting for pod get Running state...`
	`Waiting for pod get Running state...`
	`Results from execution of 'find /data' on`
		Kubernetes user with granted permissions, System administrator

Task	Description	Skills required

	`validation process pod:`
	`/data`
	`/data/out2.txt`
	`/data/out1.txt`	

Monitor operations

Task	Description	Skills required
Monitor application logs.	As part of a day-two operation, ship the applicati on logs to Amazon CloudWatc h for monitoring.	AWS systems administrator, Kubernetes user with granted permissions
Monitor Amazon EKS and Kubernetes containers with Container Insights.	As part of a day-two operation, monitor the Amazon EKS and Kubernete s systems by using Amazon CloudWatch Container Insights. This tool collects, aggregates, and summarize s metrics from containerized applications at different levels and dimensions. For more information, see the Related resources section.	AWS systems administrator, Kubernetes user with granted permissions
Monitor Amazon EFS with CloudWatch.	As part of a day-two operation, monitor the file systems using Amazon CloudWatch, which collects and processes raw data from Amazon EFS into readable, near real-time metrics. For	AWS systems administrator

Task	Description	Skills required
	more information, see the Related resources section.	

Clean up resources

Task	Description	Skills required
Clean up all created resources for the pattern.	After you complete this pattern, clean up all resources , to avoid incurring AWS charges. Run the `clean-up-resources.sh` script to remove all resources after you have finished using the PoC application. Complete one of the following options. **With encryption at rest, with a KMS key:** `./scripts/epic06/clean-up-resources.sh \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN" \ -k "$KMS_KEY_ALIAS"` where `$CLUSTER_NAME` is the name of your Amazon EKS cluster, `$EFS_CREATION_TOKEN` is the creation token for the file system, and `$KMS_KEY_`	Kubernetes user with granted permissions, System administrator

Task	Description	Skills required
	`ALIAS` is the alias for the KMS key. **Without encryption at rest:** `./scripts/epic06/clean-up-resources.sh \ -c "$CLUSTER_NAME" \ -t "$EFS_CREATION_TOKEN"` where `$CLUSTER_NAME` is the name of your Amazon EKS cluster and `$EFS_CREATION_TOKEN` is the creation token for the file system.	

Related resources

References

- AWS Fargate for Amazon EKS now supports Amazon EFS (announcement)
- How to capture application logs when using Amazon EKS on AWS Fargate (blog post)
- Using Container Insights (Amazon CloudWatch documentation)
- Setting Up Container Insights on Amazon EKS and Kubernetes (Amazon CloudWatch documentation)
- Amazon EKS and Kubernetes Container Insights metrics (Amazon CloudWatch documentation)
- Monitoring Amazon EFS with Amazon CloudWatch (Amazon EFS documentation)

GitHub tutorials and examples

- Static provisioning

- Encryption in transit
- Accessing the file system from multiple pods
- Consuming Amazon EFS in StatefulSets
- Mounting subpaths
- Using Amazon EFS access points
- Amazon EKS Blueprints for Terraform

Required tools

- Installing the AWS CLI version 2
- Installing eksctl
- Installing kubectl
- Installing jq

Additional information

The following is an example output of the `kubectl get pv` command.

```
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
            STORAGECLASS   REASON   AGE
poc-app-pv   1Mi         RWX            Retain           Bound    poc-efs-eks-fargate/
poc-app-pvc   efs-sc                   3m56s
```

The following is an example output of the `kubectl -n poc-efs-eks-fargate get pvc` command.

```
NAME          STATUS   VOLUME        CAPACITY   ACCESS MODES   STORAGECLASS   AGE
poc-app-pvc   Bound    poc-app-pv    1Mi        RWX            efs-sc         4m34s
```

Set up event-driven auto scaling in Amazon EKS by using Amazon EKS Pod Identity and KEDA

Dipen Desai, Abhay Diwan, Kamal Joshi, and Mahendra Revanasiddappa, Amazon Web Services

Summary

Orchestration platforms, such as Amazon Elastic Kubernetes Service (Amazon EKS), have streamlined the lifecycle management of container-based applications. This helps organizations focus on building, securing, operating, and maintaining container-based applications. As event-driven deployments become more common, organizations are more frequently scaling Kubernetes deployments based on various event sources. This method, combined with auto scaling, can result in significant cost savings by providing on-demand compute resources and efficient scaling that is tailored to application logic.

KEDA is a Kubernetes-based event-driven autoscaler. KEDA helps you scale any container in Kubernetes based on the number of events that need to be processed. It is lightweight and integrates with any Kubernetes cluster. It also works with standard Kubernetes components, such as Horizontal Pod Autoscaling (HPA). KEDA also offers TriggerAuthentication, which is a feature that helps you delegate authentication. It allows you to describe authentication parameters that are separate from the ScaledObject and the deployment containers.

AWS provides AWS Identity and Access Management (IAM) roles that support diverse Kubernetes deployment options, including Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on AWS (ROSA), and self-managed Kubernetes clusters on Amazon Elastic Compute Cloud (Amazon EC2). These roles use IAM constructs, such as OpenID Connect (OIDC) identity providers and IAM trust policies, to operate across different environments without relying directly on Amazon EKS services or APIs. For more information, see IAM roles for service accounts in the Amazon EKS documentation.

Amazon EKS Pod Identity simplifies the process for Kubernetes service accounts to assume IAM roles without requiring OIDC providers. It provides the ability to manage credentials for your applications. Instead of creating and distributing your AWS credentials to the containers or using the Amazon EC2 instance’s role, you associate an IAM role with a Kubernetes service account and configure your Pods to use the service account. This helps you use an IAM role across multiple clusters and simplifies policy management by enabling the reuse of permission policies across IAM roles.

By implementing KEDA with Amazon EKS Pod Identity, businesses can achieve efficient event-driven auto scaling and simplified credential management. Applications scale based on demand, which optimizes resource utilization and reduces costs.

This pattern helps you integrate Amazon EKS Pod Identity with KEDA. It showcases how you can use the `keda-operator` service account and delegate authentication with `TriggerAuthentication`. It also describes how to set up a trust relationship between an IAM role for the KEDA operator and an IAM role for the application. This trust relationship allows KEDA to monitor messages in the event queues and adjust scaling for the destination Kubernetes objects.

Prerequisites and limitations

Prerequisites

- AWS Command Line Interface (AWS CLI) version 2.13.17 or later, installed
- Python version 3.11.5 or later, installed
- AWS SDK for Python (Boto3) version 1.34.135 or later, installed
- Helm version 3.12.3 or later, installed
- kubectl version 1.25.1 or later, installed
- Docker Engine version 26.1.1 or later, installed
- An Amazon EKS cluster version 1.24 or later, created
- Prerequisites for creating the Amazon EKS Pod Identity agent, met

Limitations

- It is required that you establish a trust relationship between the `keda-operator` role and the `keda-identity` role. Instructions are provided in the Epics section of this pattern.

Architecture

In this pattern, you create the following AWS resources:

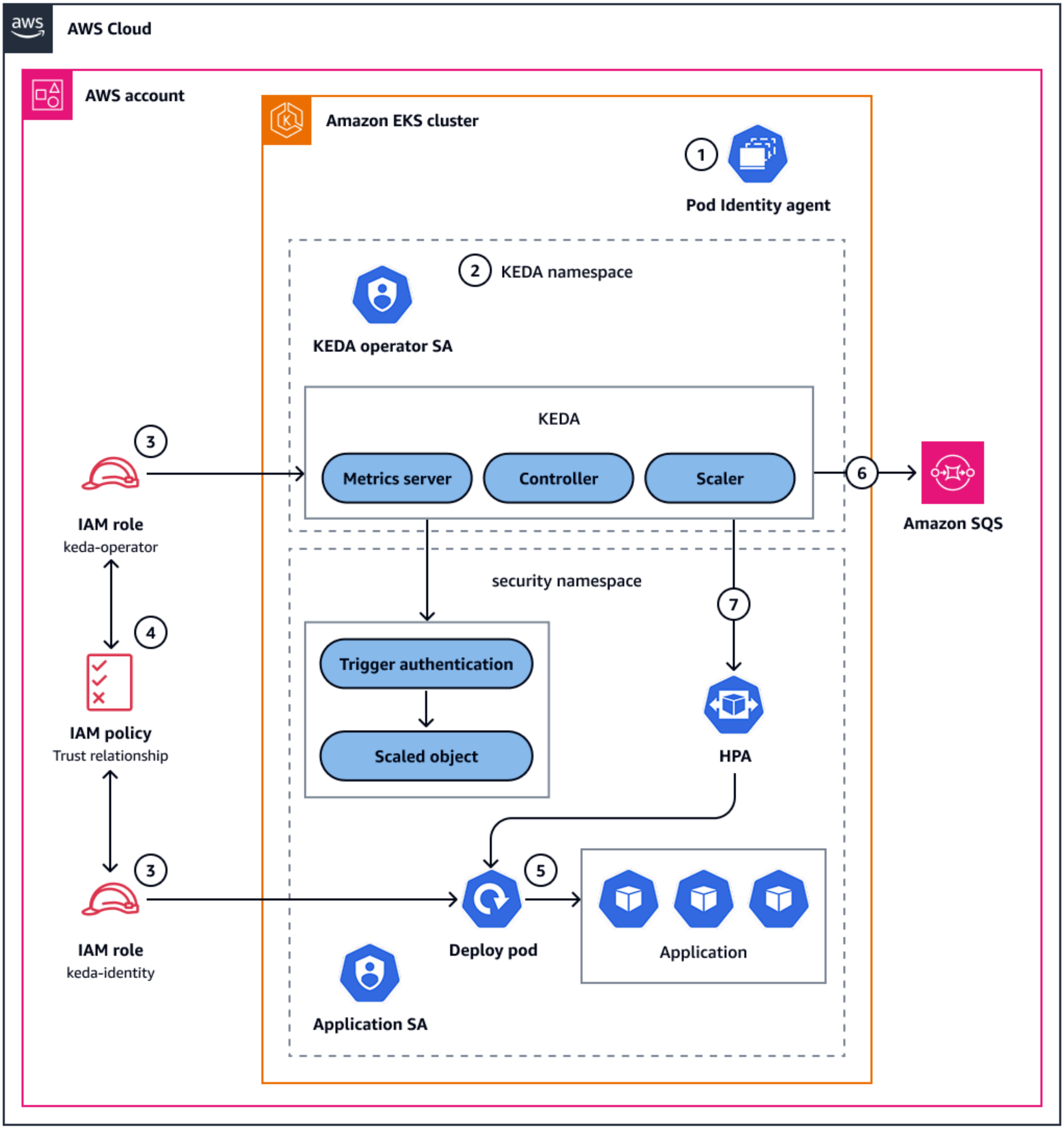
- **Amazon Elastic Container Registry (Amazon ECR) repository** – In this pattern, this repo is named `keda-pod-identity-registry`. This private repo is used to store Docker images of the sample application.
- **Amazon Simple Queue Service (Amazon SQS) queue** – In this pattern, this queue is named `event-messages-queue`. The queue acts as a message buffer that collects and stores incoming

messages. KEDA monitors the queue metrics, such as message count or queue length, and it automatically scales the application based on these metrics.

- **IAM role for the application** – In this pattern, this role is named `keda-identity`. The `keda-operator` role assumes this role. This role allows access to the Amazon SQS queue.
- **IAM role for the KEDA operator** – In this pattern, this role is named `keda-operator`. The KEDA operator uses this role to make the required AWS API calls. This role has permissions to assume the `keda-identity` role. Because of the trust relationship between the `keda-operator` and the `keda-identity` roles, the `keda-operator` role has Amazon SQS permissions.

Through the `TriggerAuthentication` and `ScaledObject` Kubernetes custom resources, the operator uses the `keda-identity` role to connect with an Amazon SQS queue. Based on the queue size, KEDA automatically scales the application deployment. It adds 1 pod for every 5 unread messages in the queue. In the default configuration, if there are no unread messages in the Amazon SQS queue, the application scales down to 0 pods. The KEDA operator monitors the queue at an interval that you specify.

The following image shows how you use Amazon EKS Pod Identity to provide the `keda-operator` role with secure access to the Amazon SQS queue.



The diagram shows the following workflow:

1. You install the Amazon EKS Pod Identity agent in the Amazon EKS cluster.
2. You deploy KEDA operator in the KEDA namespace in the Amazon EKS cluster.

3. You create the `keda-operator` and `keda-identity` IAM roles in the target AWS account.
4. You establish a trust relationship between the IAM roles.
5. You deploy the application in the `security` namespace.
6. The KEDA operator polls messages in an Amazon SQS queue.
7. KEDA initiates HPA, which automatically scales the application based on the queue size.

Tools

AWS services

- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.

Other tools

- KEDA is a Kubernetes-based event-driven autoscaler.

Code repository

The code for this pattern is available in the GitHub Event-driven auto scaling using EKS Pod Identity and KEDA repository.

Best practices

We recommend that you adhere to the following best practices:

- Amazon EKS best practices
- Security best practices in IAM
- Amazon SQS best practices

Epics

Create AWS resources

Task	Description	Skills required
Create the IAM role for the KEDA operator.	1. Sign in to the AWS Management Console, and then open the [IAM console](). 2. In the navigation pane, choose **Roles**. 3. Choose **Create role**. 4. Choose the **Custom trust policy** role type. 5. In the **Custom trust policy** section, enter the following custom trust policy for this role: `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "pods.eks.amazonaws.com" }, "Action": [ "sts:AssumeRole", "sts:TagSession" ] }`	AWS administrator

Task	Description	Skills required

	``
	    ]
	}
	``
	6. On the **Add permissions** page, choose **Next**. You do not add any policies to this role.
	7. For **Role name**, enter `keda-operator` .
	8. Choose **Create role**.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create the IAM role for the sample application.</td><td>

In the IAM console, in the navigation pane, choose Roles.
Choose Create role.
Choose the Custom trust policy role type.
In the Custom trust policy section, enter the following custom trust policy for this role. Replace <code><account number></code> with your target account number:

<pre><code>{
    "Version":
 "2012-10-17",
    "Statement": [
        {
            "Effect":
 "Allow",
            "Principa
l": {

 "Service": "pods.eks
.amazonaws.com",
                "AWS":
 "arn:aws:iam::<acc
ount number>:role/
keda-operator"
            },
            "Action":
 [

 "sts:AssumeRole",

 "sts:TagSession"
            ]
        }</code></pre>
</td><td>AWS administrator</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>
<pre><code>   ]
}</code></pre>
<ol start="5">
On the Add permissions page, add the following AWS managed polices to the role:

<code>AmazonSQSReadOnlyAccess</code>
<code>AWSLambdaSQSQueueExecutionRole</code>

Choose Next.
For Role name, enter <code>keda-identity</code> .
Choose Create role.

</td><td></td></tr>
<tr><td>Create an Amazon SQS queue.</td><td>

Open the Amazon SQS console.
Choose Create queue.
For Type, choose Standard.
On Create queue page, for Name, enter <code>event-messages-queue</code> .
Choose Create queue. You do not change any of the default settings for this queue.

</td><td>General AWS</td></tr>
</table>

Task	Description	Skills required
Create an Amazon ECR repository.	1. Open the [Amazon ECR console](). 2. Choose **Create repository**. 3. For **Repository name**, enter `keda-pod-identity-registry` . 4. Choose **Create repository**. You do not change any of the default settings for this repository.	General AWS

Set up the Amazon EKS cluster

Task	Description	Skills required
Deploy the Amazon EKS Pod Identity agent.	For the target Amazon EKS cluster, set up the Amazon EKS Pod Identity agent. Follow the instructions in [Set up the Amazon EKS Pod Identity Agent]() in the Amazon EKS documentation.	AWS DevOps

Deploy KEDA.	1. Enter the following commands to deploy KEDA on the target Amazon EKS cluster:
	```# Add Helm Repo for Keda```
	```helm repo add kedacore https://kedacore.github.io/charts```
	```# Update Helm repo```
	```helm repo update```
		DevOps engineer

Task	Description	Skills required

	<pre># Install Keda
	helm install keda
	 kedacore/keda --
	namespace keda --
	create-namespace</pre> For more information, see [Deploying with Helm](#) in the KEDA documentation.
	2. After successful deploymen t, in the output, validate that three deployments are created for the KEDA operator. The following is a sample output:<pre>NAME
	               READY
	   UP-TO-DATE
	 AVAILABLE AGE
	keda-admission-
	webhooks 1/1
	     1 1
	        89s
	keda-operator
	             1/1 1
	            1
	  89s
	keda-operator-
	metrics-apiserver
	 1/1 1
	 1 89s</pre>	

Task	Description	Skills required
Assign the IAM role to the Kubernetes service account.	Follow the instructions in [Assign an IAM role to a Kubernetes service account](#) in the Amazon EKS documentation. Use the following values: • For **IAM role**, enter `keda-operator` . • For **Kubernetes namespace** , enter `keda`. • For **Kubernetes service account**, enter `keda-operator` .	AWS DevOps
Create a namespace.	Enter the following command to create a `security` namespace in the target Amazon EKS cluster: `kubectl create ns security`	DevOps engineer

Deploy the sample application

Task	Description	Skills required
Clone the application files.	Enter the following command to clone the [Event-driven auto scaling using EKS Pod Identity and KEDA repository](#) from GitHub: `git clone https://github.com/aws-samp`	DevOps engineer

Task	Description	Skills required
	```les/event-driven-autoscaling-using-podidentity-and-keda.git```	
Build the Docker image.	1. Enter the following command to navigate into the cloned repository: ```cd event-driven-autoscaling-using-podidentity-and-keda``` 2. Enter the following command to build the Docker image for the sample application: ```docker build -t keda-pod-identity-registry .```	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Push the Docker image to Amazon ECR.</td><td>

1. In the terminal where you built the Docker image, enter the following command to log in to Amazon ECR. Replace `<AWS_REGION>` and `<AWS_ACCOUNT_ID>` with values from your AWS environment:

```
aws ecr get-login-password \
   --region <AWS_REGION> | docker login \
   --username AWS \
   --password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com
```

2. Enter the following command to tag the image. Replace `<AWS_REGION>` and `<AWS_ACCOUNT_ID>` with values from your AWS environment:

```
docker tag keda-pod-identity-registry:latest <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/keda-pod-identity-registry:latest
```

</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	3. Enter the following command to push the image to Amazon ECR. Replace `<AWS_REGION>` and `<AWS_ACCOUNT_ID>` with values from your AWS environment: `docker push <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/keda-pod-identity-registry:latest` **Note** You can find push commands by navigating to the Amazon ECR repository page and then choosing **View push commands**.	

Task	Description	Skills required
Deploy the sample application.	1. In the cloned repository, open the **deploy.yaml** file. 2. Replace `<AWS_ACCOUNT_ID>` and `<AWS_REGION>` with values from your environment. 3. Save and close the **deploy.yaml** file. 4. Enter the following command to deploy the sample application on the target Amazon EKS cluster: `kubectl apply -f deploy.yaml` This command creates a deployment and service account in the cluster.	DevOps engineer

Task	Description	Skills required

Assign the IAM role to the application service account.	Do one of the following to associate the `keda-identity` IAM role with the service account for the sample application:
	
	• Follow the instructions in Assign an IAM role to a Kubernetes service account in the Amazon EKS documentation. Use the following values:
	  ◦ For **IAM role**, enter `keda-identity` .
	  ◦ For **Kubernetes namespace**, enter `security`.
	  ◦ For **Kubernetes service account**, enter `my-sqs-read-msgs` .
	• Enter the following AWS CLI command. Replace `<cluster-name>` with the name of the target Amazon EKS cluster, and replace `<role-ARN>` with the Amazon Resource Name (ARN) of the `keda-identity` role:
	
	`aws eks create-pod-identity-association \`
	`  --cluster-name <cluster-name> \`
		DevOps engineer

Task	Description	Skills required

	``
	  --role-arn <role-ARN> \
	  --namespace security \
	  --service-account my-sqs-read-msgs
	``	

Task	Description	Skills required
Deploy `ScaledObject` and `TriggerAuthentication` .	1. In the cloned repository, open the **keda.yaml** file. 2. Replace `{{AWS_ACCOUNT_ID}}` with the ID of your target AWS account. 3. Replace `{{AWS_REGION}}` with your target AWS Region. 4. (Optional) In lines 21–24, update the parameters for the `ScaledObject` scaling policy. See the following for more information about these parameters: • pollingInterval • cooldownPeriod • idleReplicaCount • minReplicaCount 5. Save and close the **keda.yaml** file. 6. Enter the following command to deploy the `ScaledObject` and `TriggerAuthentication` resources: `kubectl -n security apply -f keda.yaml`	DevOps engineer

Test auto scaling

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Send messages to the Amazon SQS queue.</td><td>1. Enter the following command to navigate into the cloned repository:
<pre>cd event-driven-autoscaling-using-podidentity-and-keda</pre>2. Enter the following command to send test messages to the Amazon SQS queue:
<pre>python sqs_send_msg.py</pre>The sqs_send_msg.py script acts as an application that generates messages for testing auto scaling.

Note
If you're running Python 3, enter <code>python3 sqs_send_msg.py</code> .</td><td>DevOps engineer</td></tr>
<tr><td>Monitor the application pods.</td><td>1. In different terminal, enter the following command to monitor the pods:</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required

	`kubectl -n security get po`
	2. For every 5 unread messages in the Amazon SQS queue, KEDA adds one pod. In the output of the previous command, confirm that new pods are being added. The following is a sample output:
	`kubectl -n security get po`
	`NAME READY STATUS RESTARTS AGE`
	`q-read-797f4c7589-2bj76 1/1 Running 0 2s`
	`q-read-797f4c7589-4zxph 1/1 Running 0 49s`
	`q-read-797f4c7589-cg9dt 1/1 Running 0 18s`
	`q-read-797f4c7589-slc69 1/1 Running 0 33s`
	3. When you are finished testing, in the original terminal, enter CTRL + C (Windows) or CMD + C	

Task	Description	Skills required
	(macOS). This stops the **python sqs_send_msg.py** script.	

Troubleshooting

Issue	Solution

The KEDA operator cannot scale the applicati on.	Enter the following command to check the logs of the `keda-operator` IAM role:
	`kubectl logs -n keda -l app=keda-operator -c keda-operator`
	
	If there is an HTTP `403` response code, then the application and the KEDA scaler do not have sufficient permissions to access the Amazon SQS queue. Complete the following steps:
	1. Check the IAM policies and statements for the `keda-identity` role to confirm that that queue read access is granted.
	2. Validate the trust relationship between the IAM roles. The following is an example:
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Effect": "Allow",`
	`            "Principal": {`
	`                "Service": "pods.eks.amazonaws.com"`

Issue	Solution

	``
	            },
	            "Action": [
	                "sts:AssumeRole",
	                "sts:TagSession"
	            ]
	        }
	    ]
	}
	``
	
	If there is an `Assume-Role` error, then an [Amazon EKS node IAM role](#) is unable to assume the IAM role that is defined for `TriggerAuthentication`. Complete the following steps:
	
	1. Enter the following command to delete the `keda-operator` pod and create a new one:
	
	``
	kubectl delete pod keda-operator-<alphenumeric-value> --namespace keda
	``
	
	2. Enter the following command to check the identity that the pod assumes:
	
	``
	kubectl describe pod <keda-operator-pod-name> --namespace keda
	``
	
	3. When the pod successfully restarts, confirm that the following two variables are added to the pod description:
	
	• `AWS_CONTAINER_CREDENTIALS_FULL_URI`
	• `AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE`

Related resources

- Set up the Amazon EKS Pod Identity Agent (Amazon EKS documentation)
- Deploying KEDA (KEDA documentation)
- ScaledObject specification (KEDA documentation)
- Authentication with TriggerAuthentication (KEDA documentation)

Streamline PostgreSQL deployments on Amazon EKS by using PGO

Shalaka Dengale, Amazon Web Services

Summary

This pattern integrates the Postgres Operator from Crunchy Data (PGO) with Amazon Elastic Kubernetes Service (Amazon EKS) to streamline PostgreSQL deployments in cloud-native environments. PGO provides automation and scalability for managing PostgreSQL databases in Kubernetes. When you combine PGO with Amazon EKS, it forms a robust platform for deploying, managing, and scaling PostgreSQL databases efficiently.

This integration provides the following key benefits:

- Automated deployment: Simplifies PostgreSQL cluster deployment and management.
- Custom resource definitions (CRDs): Uses Kubernetes primitives for PostgreSQL management.
- High availability: Supports automatic failover and synchronous replication.
- Automated backups and restores: Streamlines backup and restore processes.
- Horizontal scaling: Enables dynamic scaling of PostgreSQL clusters.
- Version upgrades: Facilitates rolling upgrades with minimal downtime.
- Security: Enforces encryption, access controls, and authentication mechanisms.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows.
- AWS CLI Config, to connect AWS resources from the command line.
- eksctl, installed and configured on Linux, macOS, or Windows.
- `kubectl`, installed and configured to access resources on your Amazon EKS cluster. For more information, see Set up kubectl and eksctl in the Amazon EKS documentation.
- Your computer terminal configured to access the Amazon EKS cluster. For more information, see Configure your computer to communicate with your cluster in the Amazon EKS documentation.

Product versions

- Kubernetes versions 1.21–1.24 or later (see the PGO documentation).
- PostgreSQL version 10 or later. This pattern uses PostgreSQL version 16.

Limitations

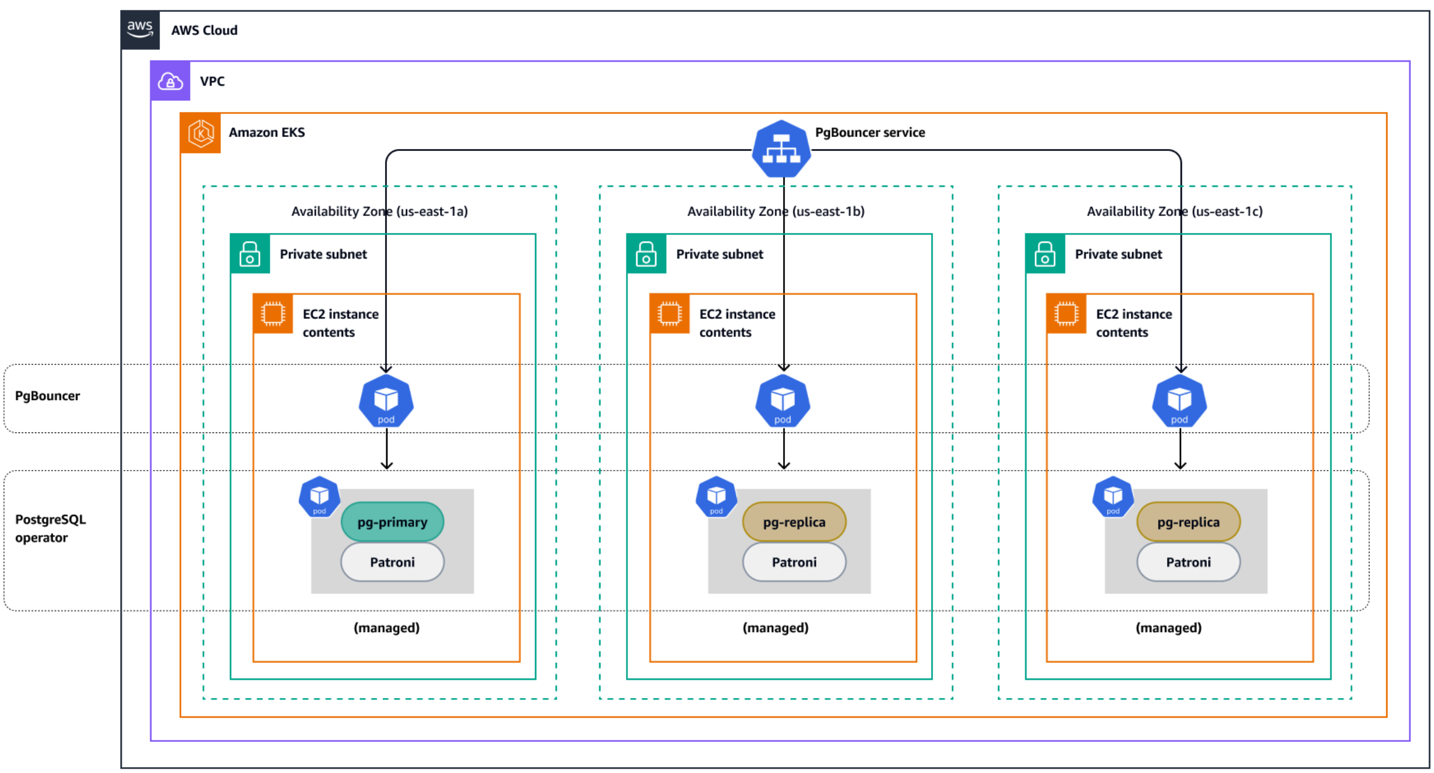
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

Target technology stack

- Amazon EKS
- Amazon Virtual Private Cloud (Amazon VPC)
- Amazon Elastic Compute Cloud (Amazon EC2)

Target architecture



This pattern builds an architecture that contains an Amazon EKS cluster with three nodes. Each node runs on a set of EC2 instances in the backend. This PostgreSQL setup follows a primary replica architecture, which is particularly effective for read-heavy use cases. The architecture includes the following components:

- **Primary database container (pg-primary)** hosts the main PostgreSQL instance where all write operations are directed.
- **Secondary replica containers (pg-replica)** host the PostgreSQL instances that replicate the data from the primary database and handle read operations.
- **PgBouncer** is a lightweight connection pooler for PostgreSQL databases that's included with PGO. It sits between the client and the PostgreSQL server, and acts as an intermediary for database connections.
- **PGO** automates the deployment and management of PostgreSQL clusters in this Kubernetes environment.
- **Patroni** is an open-source tool that manages and automates high availability configurations for PostgreSQL. It's included with PGO. When you use Patroni with PGO in Kubernetes, it plays a crucial role in ensuring the resilience and fault tolerance of a PostgreSQL cluster. For more information, see the Patroni documentation.

The workflow includes these steps:

- **Deploy the PGO operator**. You deploy the PGO operator on your Kubernetes cluster that runs on Amazon EKS. This can be done by using Kubernetes manifests or Helm charts. This pattern uses Kubernetes manifests.
- **Define PostgreSQL instances**. When the operator is running, you create custom resources (CRs) to specify the desired state of PostgreSQL instances. This includes configurations such as storage, replication, and high availability settings.
- **Operator management**. You interact with the operator through Kubernetes API objects such as CRs to create, update, or delete PostgreSQL instances.
- **Monitoring and maintenance**. You can monitor the health and performance of the PostgreSQL instances running on Amazon EKS. Operators often provide metrics and logging for monitoring purposes. You can perform routine maintenance tasks such as upgrades and patching as necessary. For more information, see Monitor your cluster performance and view logs in the Amazon EKS documentation.

- **Scaling and backup**: You can use the features provided by the operator to scale PostgreSQL instances and manage backups.

This pattern doesn't cover monitoring, maintenance, and backup operations.

Automation and scale

- You can use AWS CloudFormation to automate the infrastructure creation. For more information, see Create Amazon EKS resources with AWS CloudFormation in the Amazon EKS documentation.
- You can use GitVersion or Jenkins build numbers to automate the deployment of database instances.

Tools

AWS services

- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command line shell.

Other tools

- eksctl is a simple command line tool for creating clusters on Amazon EKS.
- kubectl is a command line utility for running commands against Kubernetes clusters.
- PGO automates and scales the management of PostgreSQL databases in Kubernetes.

Best practices

Follow these best practices to ensure a smooth and efficient deployment:

- **Secure your EKS cluster**. Implement security best practices for your EKS cluster, such as using AWS Identity and Access Management (IAM) roles for service accounts (IRSA), network policies, and VPC security groups. Limit access to the EKS cluster API server, and encrypt communications between nodes and the API server by using TLS.

- **Ensure version compatibility** between PGO and Kubernetes running on Amazon EKS. Some PGO features might require specific Kubernetes versions or introduce compatibility limitations. For more information, see Components and Compatibility in the PGO documentation.
- **Plan resource allocation** for your PGO deployment, including CPU, memory, and storage. Consider the resource requirements of both PGO and the PostgreSQL instances it manages. Monitor resource usage and scale resources as needed.
- **Design for high availability**. Design your PGO deployment for high availability to minimize downtime and ensure reliability. Deploy multiple replicas of PGO across multiple Availability Zones for fault tolerance.
- **Implement backup and restore procedures** for your PostgreSQL databases that PGO manages. Use features provided by PGO or third-party backup solutions that are compatible with Kubernetes and Amazon EKS.
- **Set up monitoring and logging** for your PGO deployment to track performance, health, and events. Use tools such as Prometheus for monitoring metrics and Grafana for visualization. Configure logging to capture PGO logs for troubleshooting and auditing.
- **Configure networking** properly to allow communications between PGO, PostgreSQL instances, and other services in your Kubernetes cluster. Use Amazon VPC networking features and Kubernetes networking plugins such as Calico or Amazon VPC CNI for network policy enforcement and traffic isolation.
- **Choose appropriate storage options** for your PostgreSQL databases, considering factors such as performance, durability, and scalability. Use Amazon Elastic Block Store (Amazon EBS) volumes or AWS managed storage services for persistent storage. For more information, see Store Kubernetes volumes with Amazon EBS in the Amazon EKS documentation.
- **Use infrastructure as code (IaC) tools** such as AWS CloudFormation to automate the deployment and configuration of PGO on Amazon EKS. Define infrastructure components—including the EKS cluster, networking, and PGO resources—as code for consistency, repeatability, and version control.

Epics

Create an IAM role

Task	Description	Skills required

Create an IAM role.	1. Create an IAM role by using the following command in the AWS CLI:
	``
	aws iam create-role \
	    --role-name
	 {YourRoleName} \
	    --assume-role-
	policy-document '{
	        "Version":
	 "2012-10-17",
	        "Statement":
	 [
	            {
	
	 "Effect": "Allow",
	
	 "Principal": {
	
	 "Service": "eks.amaz
	onaws.com"
	                },
	
	 "Action": "sts:Assu
	meRole"
	            }
	        ]
	    }' && \
	aws iam attach-role-
	policy \
	    --role-name
	 {YourRoleName}\
	    --policy-arn
	 arn:aws:iam::aws:p
	olicy/AmazonEKSClu
	sterPolicy && \
	``
		AWS administrator

Task	Description	Skills required

	``
	aws iam attach-role-policy \
	    --role-name {YourRoleName}\
	    --policy-arn arn:aws:iam::aws:policy/AmazonEKSServicePolicy && \
	aws iam attach-role-policy \
	    --role-name {YourRoleName}\
	     --policy-arn arn:aws:iam::aws:policy/CloudWatchFullAccess
	``
	2. Review the role in the AWS Management Console:
	a. Open the [IAM console](https://console.aws.amazon.com/iam/).
	b. Choose **Roles**, and search for the role name you created.
	c. Validate that the following policies are attached:
	`AmazonEKSClusterPolicy`
	`AmazonEKSServicePolicy`
	`CloudWatchFullAccess`	

Create an Amazon EKS cluster

Task	Description	Skills required

Create an Amazon EKS cluster.	If you've already deployed a cluster, skip this step. Otherwise, deploy an Amazon EKS cluster in your current AWS account by using `eksctl`, Terraform, or AWS CloudFormation. This pattern uses `eksctl` for cluster deployment.
	
	Note
	This pattern uses Amazon EC2 as a node group for Amazon EKS. If you want to use AWS Fargate, see the `managedNodeGroups` configuration in the [eksctl documentation](https://eksctl.io).
	
	1. Use the following `eksctl` input file to generate the cluster.
	
	`sample-cluster.yaml` :
	
	`apiVersion: eksctl.io/v1alpha5`
	`kind: ClusterConfig`
		AWS administrator, Terraform or eksctl administrator, Kubernetes administrator

Task	Description	Skills required
	(continued below)	

```
metadata:
  name: postgresql
  region: us-east-1
  version: "1.29"
accessConfig:
  authenticationMode
: API_AND_C
ONFIG_MAP
availabilityZones:
- us-east-1a
- us-east-1b
- us-east-1c
nodeGroups:
  - name: ng-1
    instanceType:
 m5.16xlarge
    desiredCapacity:
 2
  - name: ng-2
    instanceType:
 m5.16xlarge
    desiredCapacity:
 2
  - name: ng-3
    instanceType:
 m5.16xlarge
    desiredCapacity:
 2
vpc:
  cidr: 192.168.0
.0/16
  clusterEndpoints:
    publicAccess:
 true
  nat:
    gateway: HighlyAva
ilable
iamIdentity
Mappings:
  - arn: arn:aws:i
am::<account-id>:r
ole/<role-name> #
```

Task	Description	Skills required

	`update the IAM role ARN created in step 1`
	`    username: <user-name> # Enter the user name per your choice`
	`    noDuplicateARNs: false`
	
	2. Run the following command to create the cluster (provide the file path to your `sample-cluster.yaml` file):
	
	`eksctl create cluster -f sample-cluster.yaml`	

Validate the status of the cluster.	Run the following command to see the current status of nodes in the cluster: `kubectl get nodes` If you encounter errors, see the troubleshooting section of the Amazon EKS documentation.	AWS administrator, Terraform or eksctl administrator, Kubernetes administrator

Create an OIDC identity provider

Task	Description	Skills required
Enable the IAM OIDC provider.	As a prerequisite for the Amazon EBS Container	AWS administrator

Task	Description	Skills required
	Storage Interface (CSI) driver, you must have an existing IAM OpenID Connect (OIDC) provider for your cluster. Enable the IAM OIDC provider by using the following command: `eksctl utils associate -iam-oidc-provider --region={region} --cluster={YourClusterNameHere} --approve` For more information about this step, see the Amazon EKS documentation.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create an IAM role for the Amazon EBS CSI driver.</td><td>Use the following <code>eksctl</code> command to create the IAM role for the CSI driver:

<pre><code>eksctl create iamserviceaccount \
  --region {RegionName} \
  --name ebs-csi-controller-sa \
  --namespace kube-system \
  --cluster {YourClusterNameHere} \
  --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --approve \
  --role-only \
  --role-name AmazonEKS_EBS_CSI_DriverRole</code></pre>

If you use encrypted Amazon EBS drives, you have to configure the policy further. For instructions, see the Amazon EBS SCI driver documentation.</td><td>AWS administrator</td></tr>
</table>

Task	Description	Skills required
Add the Amazon EBS CSI driver.	Use the following `eksctl` command to add the Amazon EBS CSI driver: `eksctl create addon \ --name aws-ebs-csi-driver \ --cluster <YourClusterName> service-account-role-arn arn:aws:iam::$(aws sts get-caller-identity \ --query Account \ --output text):role/AmazonEKS_EBS_CSI_DriverRole \ --force`	AWS administrator

Install PGO

Task	Description	Skills required
Clone the PGO repository.	Clone the GitHub repository for PGO: `git clone https://github.com/CrunchyData/postgres-operator-examples.git`	AWS DevOps
Provide the role details for service account creation.	To grant the Amazon EKS cluster access to the required AWS resources, specify the Amazon Resource Name (ARN) of the OIDC role that you created earlier in the	AWS administrator, Kubernetes administrator

Task	Description	Skills required

	`service_account.yaml` file. This file is located in the [namespace folder](#) of the repository.
	`cd postgres-operator-examples`
	`---`
	`metadata:`
	`  annotations:`
	`    eks.amazonaws.com/role-arn: arn:aws:iam::<accountId>:role/<role_name> # Update the OIDC role ARN created earlier`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create the namespace and PGO prerequisites.</td><td>1. Run the following command to create the namespace:
<pre>kubectl apply -k
 kustomize/install/
namespace</pre>This establishes a dedicated namespace for PGO. If necessary, you can modify the <code>namespace.yml</code> file and assign the namespace a different name.
2. Run the following command to apply the default configuration to the cluster:
<pre>kubectl apply --
server-side -k
 kustomize/install/
default</pre><code>kustomize/install/default</code> provides the default configuration for the Kubernetes role-based access control (RBAC), custom resource definition (CRD), and Kubernetes Manager files.</td><td>Kunernetes administrator</td></tr>
</table>

Task	Description	Skills required
Verify the creation of pods.	Verify that the namespace and default configuration were created: `kubectl get pods -n postgres-operator`	AWS administrator, Kubernetes administrator
Verify PVCs.	Use the following command to verify persistent volume claims (PVCs): `kubectl describe pvc -n postgres-operator`	AWS administrator, Kubernetes administrator

Create and deploy an operator

Task	Description	Skills required

Create an operator.	Revise the contents of the file located at `/kustomize/postgres/postgres.yaml` to match the following:
	<pre>spec:
	  instances:
	    - name: pg-1
	      replicas: 3
	  patroni:
	    dynamicConfiguration:
	      postgresql:
	      pg_hba:
	        - "host all all 0.0.0.0/0 trust" # this line enabled</pre>
		AWS administrator, DBA, Kubernetes administrator

Task	Description	Skills required

	<pre>logical replication
	with programmatic
	access
	        - "host all
	postgres 127.0.0.1/32
	md5"
	      synchronous_mode:
	true
	 users:
	 - name: replicator
	   databases:
	     - testdb
	   options: "REPLICAT
	ION"</pre> These updates do the following:
	• Adjust the PostgreSQL configuration settings to facilitate access to the PostgreSQL instance.
	• Include configurations for replication user, database user, and superuser to enable streaming replication, database access, and cluster management.	

Deploy the operator.	Deploy the PGO operator to enable the streamlined management and operation of PostgreSQL databases in Kubernetes environments:
	<pre>kubectl apply -k
	 kustomize/postgres</pre>
		AWS administrator, DBA, Kubernetes administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Verify the deployment.</td><td>1. Validate that the operator has been deployed:
<pre>kubectl get pods -n postgres-operator --selector=postgres-operator.crunchydata.com/instance-set \   -L postgres-operator.crunchydata.com/role</pre>2. Verify that the service resource associated with the operator pod has been created:
<pre>kubectl get svc -n postgres-operator</pre>From the command output, note the primary replica (<code>primary_pod_name</code>) and read replica (<code>read_pod_name</code>). You will uses these in the next steps.</td><td>AWS administrator, DBA, Kubernetes administrator</td></tr>
</table>

Verify streaming replication

Task	Description	Skills required
Write data to the primary replica.	Use the following commands to connect to the PostgreSQL primary replica and write data to the database:	AWS administrator, Kubernetes administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>
<pre><code>kubectl exec -it
 <primary_pod_name>
 bash -n postgres-
operator</code></pre>
<pre><code>psql</code></pre>
<pre><code>CREATE TABLE customers
 (firstname text,
 customer_id serial,
 date_created timestamp
);
\dt</code></pre>
</td><td></td></tr>
<tr><td>Confirm that the read replica has the same data.</td><td>
Connect to the PostgreSQL read replica and check whether the streaming replication is working correctly:
<pre><code>kubectl exec -it
 {read_pod_name} bash -
n postgres-operator</code></pre>
<pre><code>psql</code></pre>
<pre><code>\dt</code></pre>
The read replica should have the table that you created in the primary replica in the previous step.
</td><td>AWS administrator, Kubernetes administrator</td></tr>
</table>

Troubleshooting

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td>The pod doesn't start.</td><td>

- Use the following command to inspect pod status:

```
kubectl get pods -n your-namespace
```

- Inspect the logs for any errors:

```
kubectl logs your-pod-name -n your-namespace
```

- Check pod events for any abnormal events related to your pods:

```
kubectl describe pod your-pod-name -n your-namespace
```

</td></tr>
<tr><td>Replicas are significantly behind the primary database.</td><td>

- Check for replication lag:

```
SELECT * FROM pg_stat_replication;
```

- Make sure that replicas have enough CPU and memory resources. Check resource limits:

```
kubectl describe pod your-replica-pod -n your-namespace
```

- Verify that the storage backend is performing optimally. Slow disk I/O can cause replication lag.

</td></tr>
<tr><td>You don't have visibility into the performance and health of the PostgreSQL cluster.</td><td>

- Enable Amazon CloudWatch Logs and make sure that logs are being sent to Amazon CloudWatch for analysis. For more information, see the Amazon EKS documentation.

</td></tr>
</table>

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td></td><td>Check <code>pg_stat_activity</code> :<pre><code>SELECT * FROM pg_stat_activity;</code></pre></td></tr>
<tr><td>Replication doesn't work.</td><td>Check the primary configuration by viewing the replication settings in <code>postgresql.conf</code> :<pre><code>wal_level = replica</code></pre><pre><code>max_wal_senders = 10</code></pre><pre><code>wal_keep_size = 64 # or wal_keep_segments in older versions</code></pre>
Verify that <code>pg_hba.conf</code> includes replication permissions:<pre><code>host replication replica_user all md5</code></pre>
Check replica configuration. Make sure that <code>recovery.conf</code> or equivalent settings (<code>standby.signal</code> and <code>primary_conninfo</code>) are correctly set up on replicas.</td></tr>
</table>

Related resources

- Amazon Elastic Kubernetes Service (*Overview of Deployment Options on AWS* whitepaper)
- AWS CloudFormation (*Overview of Deployment Options on AWS* whitepaper)
- Get started with Amazon EKS – eksctl (*Amazon EKS User Guide*)
- Set up kubectl and eksctl (*Amazon EKS User Guide*)
- Create a role for OpenID Connect federation (*IAM User Guide*)
- Configuring settings for the AWS CLI (*AWS CLI User Guide*)

- Crunchy Postgres for Kubernetes documentation
- Crunch & Learn: Crunchy Postgres for Kubernetes 5.0 (video)

Simplify application authentication with mutual TLS in Amazon ECS by using Application Load Balancer

Olawale Olaleye and Shamanth Devagari, Amazon Web Services

Summary

This pattern helps you to simplify your application authentication and offload security burdens with mutual TLS in Amazon Elastic Container Service (Amazon ECS) by using Application Load Balancer (ALB). With ALB, you can authenticate X.509 client certificates from AWS Private Certificate Authority. This powerful combination helps to achieve secure communication between your services, reducing the need for complex authentication mechanisms within your applications. In addition, the pattern uses Amazon Elastic Container Registry (Amazon ECR) to store container images.

The example in this pattern uses Docker images from a public gallery to create the sample workloads initially. Subsequently, new Docker images are built to be stored in Amazon ECR. For the source, consider a Git-based system such as GitHub, GitLab, or Bitbucket, or use Amazon Simple Storage Service Amazon S3 (Amazon S3). For building the Docker images, consider using AWS CodeBuild for the subsequent images.

Prerequisites and limitations

Prerequisites

- An active AWS account with access to deploy AWS CloudFormation stacks. Make sure that you have AWS Identity and Access Management (IAM) user or role permissions to deploy CloudFormation.
- AWS Command Line Interface (AWS CLI) installed. Configure your AWS credentials on your local machine or in your environment by either using the AWS CLI or by setting the environment variables in the `~/.aws/credentials` file.
- OpenSSL installed.
- Docker installed.
- Familiarity with the AWS services described in Tools.
- Knowledge of Docker and NGINX.

Limitations

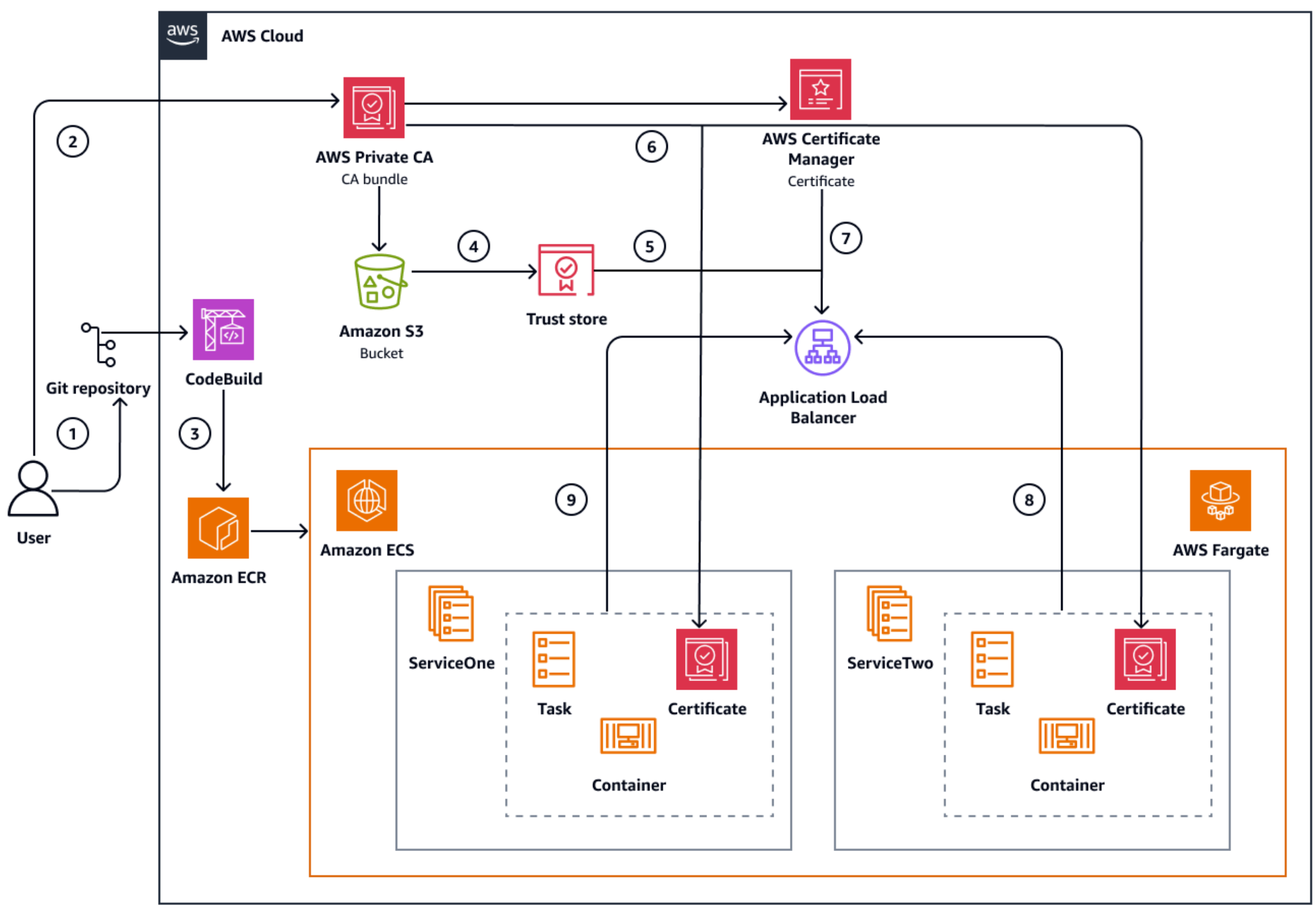
- Mutual TLS for Application Load Balancer only supports X.509v3 client certificates. X.509v1 client certificates are not supported.
- The CloudFormation template that is provided in this pattern's code repository doesn't include provisioning a CodeBuild project as part of the stack.
- Some AWS services aren't available in all AWS Regions. For Region availability, see [AWS Services by Region](). For specific endpoints, see [Service endpoints and quotas](), and choose the link for the service.

Product versions

- Docker version 27.3.1 or later
- AWS CLI version 2.14.5 or later

Architecture

The following diagram shows the architecture components for this pattern.



The diagram shows the following workflow:

1. Create a Git repository, and commit the application code to the repository.
2. Create a private certificate authority (CA) in AWS Private CA.
3. Create a CodeBuild project. The CodeBuildproject is triggered by commit changes and creates the Docker image and publishes the built image to Amazon ECR.
4. Copy the certificate chain and certificate body from the CA, and upload the certificate bundle to Amazon S3.
5. Create a trust store with the CA bundle that you uploaded to Amazon S3. Associate the trust store with the mutual TLS listeners on the Application Load Balancer (ALB).
6. Use the private CA to issue client certificates for the container workloads. Also create a private TLS certificate using AWS Private CA.
7. Import the private TLS certificate into AWS Certificate Manager (ACM), and use it with the ALB.
8. The container workload in `ServiceTwo` uses the issued client certificate to authenticate with the ALB when it communicates with the container workload in `ServiceOne`.
9. The container workload in `ServiceOne` uses the issued client certificate to authenticate with the ALB when it communicates with the container workload in `ServiceTwo`.

Automation and scale

This pattern can be fully automated by using CloudFormation, AWS Cloud Development Kit (AWS CDK) , or API operations from an SDK to provision the AWS resources.

You can use AWS CodePipeline to implement a continuous integration and continuous deployment (CI/CD) pipeline using CodeBuild to automate container image build process and deploying new releases to the Amazon ECS cluster services.

Tools

AWS services

- [AWS Certificate Manager (ACM)] helps you create, store, and renew public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications.
- [AWS CloudFormation] helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage.
- Amazon ECS Exec allows you to directly interact with containers without needing to first interact with the host container operating system, open inbound ports, or manage SSH keys. You can use ECS Exec to run commands in, or get a shell to, a container running on an Amazon EC2 instance or on AWS Fargate.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon EC2 instances, containers, and IP addresses, in one or more Availability Zones. ELB monitors the health of its registered targets, and routes traffic only to the healthy targets. ELB scales your load balancer as your incoming traffic changes over time. It can automatically scale to the majority of workloads.
- AWS Fargate helps you run containers without needing to manage servers or Amazon EC2 instances. Fargate is compatible with both Amazon ECS and Amazon Elastic Kubernetes Service (Amazon EKS). You can run your Amazon ECS tasks and services with the Fargate launch type or a Fargate capacity provider. To do so, package your application in containers, specify the CPU and memory requirements, define networking and IAM policies, and launch the application. Each Fargate task has its own isolation boundary and doesn't share the underlying kernel, CPU resources, memory resources, or elastic network interface with another task.
- AWS Private Certificate Authority enables creation of private certificate authority (CA) hierarchies, including root and subordinate CAs, without the investment and maintenance costs of operating an on-premises CA.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- GitHub, GitLab, and Bitbucket are some of the commonly used Git-based source control system to keep track of source code changes.

- NGINX Open Source is an open source load balancer, content cache, and web server. This pattern uses it as a web server.
- OpenSSL is an open source library that provides services that are used by the OpenSSL implementations of TLS and CMS.

Code repository

The code for this pattern is available in the GitHub mTLS-with-Application-Load-Balancer-in-Amazon-ECS repository.

Best practices

- Use Amazon ECS Exec to run commands or get a shell to a container running on Fargate. You can also use ECS Exec to help collect diagnostic information for debugging.
- Use security groups and network access control lists (ACLs) to control inbound and outbound traffic between the services. Fargate tasks receive an IP address from the configured subnet in your virtual private cloud (VPC).

Epics

Create the repository

Task	Description	Skills required
Download the source code.	To download this pattern's source code, fork or clone the GitHub mTLS-with-Applicat ion-Load-Balancer-in-Amazon -ECS repository.	DevOps engineer
Create a Git repository.	To create a Git repository to contain the Dockerfile and the `buildspec.yaml` files, use the following steps: 1. Create a folder in your virtual environment. Name it with your project name.	DevOps engineer

Task	Description	Skills required
	2. Open a terminal on your local machine, and navigate to this folder. 3. To clone the [mTLS-with-Application-Load-Balancer-in-Amazon-ECS](https://github.com/aws-samples/mTLS-with-Application-Load-Balancer-in-Amazon-ECS) repository to your project directory, enter the following command: `git clone https://github.com/aws-samples/mTLS-with-Application-Load-Balancer-in-Amazon-ECS.git`	

Create CA and generate certificates

Task	Description	Skills required

Create a private CA in AWS Private CA.	To create a private certificate authority (CA), run the following commands in your terminal. Replace the values in the example variables with your own values.
	
	`export AWS_DEFAULT_REGION="us-west-2"`
	`export SERVICES_DOMAIN="www.example.com"`
		DevOps engineer, AWS DevOps

Task	Description	Skills required
	```export ROOT_CA_ARN=`aws acm-pca create-certificate-authority \ --certificate-authority-type ROOT \ --certificate-authority-configuration \ "KeyAlgorithm=RSA_2048, SigningAlgorithm=SHA256WITHRSA, Subject={ Country=US, State=WA, Locality=Seattle, Organization=Build on AWS, OrganizationalUnit=mTLS Amazon ECS and ALB Example, CommonName=${SERVICES_DOMAIN}}" \ --query CertificateAuthorityArn --output text` ``` For more details, see [Create a private CA in AWS Private CA](#) in the AWS documentation.	

Task	Description	Skills required

Create and install your private CA certificate.	To create and install a certificate for your private root CA, run the following commands in your terminal:
	
	1. Generate a certificate signing request (CSR).
	
	`ROOT_CA_CSR=`aws acm-pca get-certificate-authority-csr \ --certificate-authority-arn ${ROOT_CA_ARN} \ --query Csr --output text``
	
	2. Issue the root certificate.
	
	`AWS_CLI_VERSION=$( aws --version 2>&1 | cut -d/ -f2 | cut -d. -f1)`
	`[[ ${AWS_CLI_VERSION} -gt 1 ]] && ROOT_CA_CSR="$(echo ${ROOT_CA_CSR} | base64)"`
	
	`ROOT_CA_CERT_ARN=`aws acm-pca issue-certificate \ --certificate-authority-arn ${ROOT_CA_ARN} \ --template-arn arn:aws:acm-pca:::template/RootCACertificate/V1 \`
		AWS DevOps, DevOps engineer

Task	Description	Skills required
	``` --signing- algorithm SHA256WIT HRSA \ --validity Value=10,Type=YEARS \ --csr "${ROOT_C A_CSR}" \ --query Certifica teArn --output text` ``` 3. Retrieve the root certifica te. ``` ROOT_CA_CERT=`aws  acm-pca get-certi ficate \     --certificate-arn  ${ROOT_CA_CERT_ARN}  \     --certificate- authority-arn  ${ROOT_CA_ARN} \     --query Certifica te --output text` # store for later use aws acm-pca get-certi ficate \ --certificate-arn ${ROOT_CA_CERT_ARN} \ --certificate- authority-arn ${ROOT_CA_ARN} \ --query Certifica te --output text > ca-cert.pem ``` 4. Import the root CA certifica te to install it on the CA.	

Task	Description	Skills required
	```[[${AWS_CLI_VERSION} -gt 1]] && ROOT_CA_CERT="$(echo ${ROOT_CA_CERT} | base64)" aws acm-pca import-certificate-authority-certificate \ --certificate-authority-arn $ROOT_CA_ARN \ --certificate "${ROOT_CA_CERT}"``` For more details, see Installing the CA certificate in the AWS documentation.	
Request a managed certificate.	To request a private certificate in AWS Certificate Manager to use with your private ALB, use the following command: ```export TLS_CERTIFICATE_ARN=`aws acm request-certificate \ --domain-name "*.${DOMAIN_DOMAIN}" \ --certificate-authority-arn ${ROOT_CA_ARN} \ --query CertificateArn --output text` ```	DevOps engineer, AWS DevOps

Task	Description	Skills required
Use the private CA to issue a client certificate.	• To create a certificate signing request (CSR) for the two services, use the following AWS CLI command: `openssl req -out client_csr1.pem -new -newkey rsa:2048 -nodes -keyout client_private-key1.pem` `openssl req -out client_csr2.pem -new -newkey rsa:2048 -nodes -keyout client_private-key2.pem` This command returns the CSR and the private key for the two services. • To issue a certificate for the services, run the following commands to use the private CA that you created: `SERVICE_ONE_CERT_ARN=`aws acm-pca issue-certificate \` ` --certificate-authority-arn ${ROOT_CA_ARN} \`	DevOps engineer, AWS DevOps

Task	Description	Skills required
	``` --csr fileb://c lient_csr1.pem \ --signing-algorith m "SHA256WITHRSA" \ --validity Value=5,Type="YEARS" --query CertificateArn --output text`  echo "SERVICE_ONE_CERT_ ARN: ${SERVICE _ONE_CERT_ARN}"  aws acm-pca get-certi ficate \     --certificate-auth ority-arn ${ROOT_CA _ARN} \     --certificate-arn  ${SERVICE_ONE_CERT _ARN} \      | jq -r '.Certifi cate' > client_ce rt1.cert  SERVICE_TWO_CERT_ ARN=`aws acm-pca issue- certificate \ --certificate-auth ority-arn ${ROOT_CA _ARN} \ --csr fileb://c lient_csr2.pem \ --signing-algorith m "SHA256WITHRSA" \ --validity Value=5,Type="YEARS" --query CertificateArn --output text` ```	

Task	Description	Skills required

	``
	echo "SERVICE_TWO_CERT_
	ARN: ${SERVICE
	_TWO_CERT_ARN}"
	
	aws acm-pca get-certi
	ficate \
	    --certificate-auth
	ority-arn ${ROOT_CA
	_ARN} \
	    --certificate-arn
	 ${SERVICE_TWO_CERT
	_ARN} \
	     | jq -r '.Certifi
	cate' > client_ce
	rt2.cert
	``
	For more information, see [Issue private end-entity certificates]() in the AWS documentation.	

Provision AWS services

Task	Description	Skills required
Provision AWS services with the CloudFormation template.	To provision the virtual private cloud (VPC), Amazon ECS cluster, Amazon ECS services, Application Load Balancer, and Amazon Elastic Container Registry (Amazon ECR), use the CloudFormation template.	DevOps engineer
Get variables.	Verify that you have an Amazon ECS cluster with two services running. To retrieve	DevOps engineer

Task	Description	Skills required

	the resource details and store them as variables, use the following commands:
	`export LoadBalancerDNS=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`LoadBalancerDNS`].OutputValue')`
	
	`export ECRRepositoryUri=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`ECRRepositoryUri`].OutputValue')`
	
	`export ECRRepositoryServiceOneUri=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`ECRRepositoryServiceOneUri`].OutputValue')`
	
	`export ECRRepositoryServiceTwoUri=$(aws cloudformation`	

Task	Description	Skills required
	``` describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`ECRRepository ServiceTwoUri`].Ou tputValue') export ClusterName= $(aws cloudformation describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`ClusterName`] .OutputValue') export BucketName= $(aws cloudformation describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`BucketName`]. OutputValue') export Service1L istenerArn=$(aws cloudformation describe-stacks -- stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputK ey==`Service1Liste nerArn`].OutputVal ue') ```	

Task	Description	Skills required
	```export Service2ListenerArn=$(aws cloudformation describe-stacks --stack-name ecs-mtls \ --output text \ --query 'Stacks[0].Outputs[?OutputKey==`Service2ListenerArn`].OutputValue')```	

Task	Description	Skills required
Create a CodeBuild project.	To use a CodeBuild project to create the Docker images for your Amazon ECS services, do the following: 1. Sign in to the AWS Management Console, and open the CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/. 2. Create a new project. For **Source**, choose the Git repository that you created. For information about different kinds of Git repository integration, see [Working with connections](#) in the AWS documentation. 3. Confirm that **Privileged** mode is enabled. To build Docker images, this mode is necessary. Otherwise, the image will not build successfully. 4. Use the custom `buildspec.yaml` file shared for each service. 5. Provide values for the project name and descripti on. For more details, see [Create a build project in AWS](#)	AWS DevOps, DevOps engineer

Task	Description	Skills required
	CodeBuild in the AWS documentation.	

Task	Description	Skills required

Build the Docker images.	You can use CodeBuild to perform the image build process. CodeBuild needs permissions to interact with Amazon ECR and to work with Amazon S3.
	
	As part of the process, the Docker image is built and pushed to the Amazon ECR registry. For details about the template and the code, see [Additional information](#).
	
	(Optional) To build locally for test purposes, use the following command:
	
	`# login to ECR`
	`aws ecr get-login -password | docker login --username AWS --password-stdin $ECRRepositoryUri`
	
	`# build image for service one`
	`cd /service1`
	`aws s3 cp s3://$BucketName/serviceone/ service1/ --recursive`
	`docker build -t $ECRRepositoryServiceOneUri .`
	`docker push $ECRRepositoryServiceOneUri`
	
	`# build image for service two`
		DevOps engineer

Task	Description	Skills required
	``` cd ../service2 aws s3 cp s3://$Buc ketName/servicetwo/ service2/ --recursive docker build -t $ECRRepositoryServ iceTwoUri . docker push $ECRRepos itoryServiceTwoUri ```	

Enable mutual TLS

Task	Description	Skills required
Upload the CA certificate to Amazon S3.	To upload the CA certifica te to the Amazon S3 bucket, use the following example command: `aws s3 cp ca-cert.pem s3://$BucketName/a cm-trust-store/`	AWS DevOps, DevOps engineer

Create the trust store.	To create the trust store, use the following example command:
	``
	TrustStoreArn=`aws
	 elbv2 create-trust-
	store --name acm-pca-t
	rust-certs \
	    --ca-certificates-
	bundle-s3-bucket
	 $BucketName \
	    --ca-certificates-
	bundle-s3-key acm-
	trust-store/ca-
	cert.pem --query
	``
		AWS DevOps, DevOps engineer

Task	Description	Skills required
	``` 'TrustStores[].Tru stStoreArn' --output text` ```	
Upload client certificates.	To upload client certificates to Amazon S3 for Docker images, use the following example command: ``` # for service one aws s3 cp client_ce rt1.cert s3://$Buc ketName/serviceone/ aws s3 cp client_pr ivate-key1.pem s3:// $BucketName/serviceone / # for service two aws s3 cp client_ce rt2.cert s3://$Buc ketName/servicetwo/ aws s3 cp client_pr ivate-key2.pem s3:// $BucketName/servicetwo / ```	AWS DevOps, DevOps engineer

Task	Description	Skills required
Modify the listener.	To enable mutual TLS on the ALB, modify the HTTPS listeners by using the following commands: `aws elbv2 modify-li stener \ --listener-arn $Service1ListenerArn \ --certificates CertificateArn=$TL S_CERTIFICATE_ARN_TWO \ --ssl-policy ELBSecurityPolicy- 2016-08 \ --protocol HTTPS \ --port 8080 \ --mutual-authentic ation Mode=veri fy,TrustStoreArn=$ TrustStoreArn,Igno reClientCertificat eExpiry=false`  `aws elbv2 modify-li stener \ --listener-arn $Service2ListenerArn \ --certificates CertificateArn=$TL S_CERTIFICATE_ARN_TWO \ --ssl-policy ELBSecurityPolicy- 2016-08 \ --protocol HTTPS \ --port 8090 \`	AWS DevOps, DevOps engineer

Task	Description	Skills required
	``` --mutual-authentication Mode=verify,TrustStoreArn=$TrustStoreArn,IgnoreClientCertificateExpiry=false ``` For more information, see [Configuring mutual TLS on an Application Load Balancer](#) in the AWS documentation.	

Update the services

Task	Description	Skills required

Update the Amazon ECS task definition.	To update the Amazon ECS task definition, modify the `image` parameter in the new revision.
	
	To get the values for the respective services, update the task definitions with the new Docker images Uri that you built in the previous steps: `echo $ECRRepositoryServiceOneUri` or `echo $ECRRepositoryServiceTwoUri`
	
	``
	    "containerDefinitions": [
	        {
	``
		AWS DevOps, DevOps engineer

Task	Description	Skills required

	``
	              "name":
	 "nginx",
	              "image":
	 "public.ecr.aws/ng
	inx/nginx:latest",
	 # <----- change to new
	 Uri
	              "cpu": 0,
	``
	For more information, see [Updating an Amazon ECS task definition](#) using the console in the AWS documentation.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Update the Amazon ECS service.</td><td>Update the service with the latest task definition. This task definition is the blueprint for the newly built Docker images, and it contains the client certificate that's required for the mutual TLS authentication.

To update the service, use the following procedure:

1. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/v2.
2. On the Clusters page, choose the cluster.
3. On the cluster details page, in the Services section, select the checkbox next to the service, and then choose Update.
4. To have your service start a new deployment, select Force new deployment.
5. For Task definition, choose the task definition family and the latest revision.
6. Choose Update.

Repeat the steps for the other service.</td><td>AWS administrator, AWS DevOps, DevOps engineer</td></tr>
</table>

Access the application

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Copy the application URL.</td><td>Use the Amazon ECS console to view the task. When the task status has been updated to Running, select the task. In the Task section, copy the task ID.</td><td>AWS administrator, AWS DevOps</td></tr>
<tr><td>Test your application.</td><td>To test your application, use ECS Exec to access the tasks.

1. For service one, use the following command:
<pre>container="nginx"

ECS_EXEC_TASK_ARN
="<TASK ARN>"
aws ecs execute-c
ommand --cluster
 $ClusterName \
    --task $ECS_EXEC
_TASK_ARN \
    --container
 $container \
    --interactive \
    --command "/bin/
bash"</pre>2. In the service one task’s container, use the following command to enter the internal load balancer <code>url</code> and the listener port that points to service two. Then specify the path to the client</td><td>AWS administrator, AWS DevOps</td></tr>
</table>

Task	Description	Skills required

	certificate to test the application:
	`curl -kvs https://<internal-alb-url>:8090 --key /usr/local/share/ca-certificates/client.key --cert /usr/local/share/ca-certificates/client.crt`
	3. In the service two task’s container, use the following command to enter the internal load balancer `url` and the listener port that points to service one. Then specify the path to the client certificate to test the application:
	`curl -kvs https://<internal-alb-url>:8080 --key /usr/local/share/ca-certificates/client.key --cert /usr/local/share/ca-certificates/client.crt`
	Note
	The `-k` flag in the curl commands (as part of `-kvs`) disables	

Task	Description	Skills required
	SSL certificate validation. You can remove this flag when using an SSL certificate that matches your domain name, enabling proper certificate validation.	

Related resources

Amazon ECS documentation

- Creating an Amazon ECS task definition using the console
- Creating a container image for use on Amazon ECS
- Amazon ECS clusters
- Amazon ECS for AWS Fargate
- Amazon ECS networking best practices
- Amazon ECS service definition parameters

Other AWS resources

- How do I use AWS private CA to configure mTLS on the Application Load Balancer? (AWS re:Post)

Additional information

Editing the Dockerfile

The following code shows the commands that you edit in the Dockerfile for service 1:

```
FROM public.ecr.aws/nginx/nginx:latest
WORKDIR /usr/share/nginx/html
```

```
RUN echo "Returning response from Service 1: Ok" > /usr/share/nginx/html/index.html
ADD client_cert1.cert client_private-key1.pem /usr/local/share/ca-certificates/
RUN chmod -R 400 /usr/local/share/ca-certificates/
```

The following code shows the commands that you edit in the Dockerfile for service 2:

```
FROM public.ecr.aws/nginx/nginx:latest
WORKDIR /usr/share/nginx/html
RUN echo "Returning response from Service 2: Ok" > /usr/share/nginx/html/index.html
ADD client_cert2.cert client_private-key2.pem /usr/local/share/ca-certificates/
RUN chmod -R 400 /usr/local/share/ca-certificates/
```

If you're building the Docker images with CodeBuild, the `buildspec` file uses the CodeBuild build number to uniquely identify image versions as a tag value. You can change the `buildspec` file to fit your requirements, as shown in the following `buildspec` custom code:

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker login --
username AWS --password-stdin $ECR_REPOSITORY_URI
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
        # change the S3 path depending on the service
      - aws s3 cp s3://$YOUR_S3_BUCKET_NAME/serviceone/ $CodeBuild_SRC_DIR/ --
recursive
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $ECR_REPOSITORY_URI:latest .
      - docker tag $ECR_REPOSITORY_URI:latest $ECR_REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $ECR_REPOSITORY_URI:latest
      - docker push $ECR_REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
```

```
      # for ECS deployment reference
      - printf '[{"name":"%s","imageUri":"%s"}]' $CONTAINER_NAME $ECR_REPOSITORY_URI:
$IMAGE_TAG > imagedefinitions.json

artifacts:
  files:
    - imagedefinitions.json
```

More patterns

- Assess application readiness for migration to the AWS Cloud by using CAST Highlight
- Automate deletion of AWS CloudFormation stacks and associated resources
- Automate dynamic pipeline management for deploying hotfix solutions in Gitflow environments by using AWS Service Catalog and AWS CodePipeline
- Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK
- Build and push Docker images to Amazon ECR using GitHub Actions and Terraform
- Containerize mainframe workloads that have been modernized by Blu Age
- Create a custom log parser for Amazon ECS using a Firelens log router
- Deploy a CI/CD pipeline for Java microservices on Amazon ECS
- Deploy agentic systems on Amazon Bedrock with the CrewAI framework by using Terraform
- Deploy an Amazon EKS cluster from AWS Cloud9 using an EC2 instance profile
- Deploy an environment for containerized Blu Age applications by using Terraform
- Deploy preprocessing logic into an ML model in a single endpoint using an inference pipeline in Amazon SageMaker
- Deploy workloads from Azure DevOps pipelines to private Amazon EKS clusters
- Implement AI-powered Kubernetes diagnostics and troubleshooting with K8sGPT and Amazon Bedrock integration
- Manage blue/green deployments of microservices to multiple accounts and Regions by using AWS code services and AWS KMS multi-Region keys
- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk
- Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS
- Set up a minimum viable data space to share data between organizations
- Modernize ASP.NET Web Forms applications on AWS
- Monitor Amazon ECR repositories for wildcard permissions using AWS CloudFormation and AWS Config
- Monitor application activity by using CloudWatch Logs Insights
- Set up a CI/CD pipeline for hybrid workloads on Amazon ECS Anywhere by using AWS CDK and GitLab

- Set up a Helm v3 chart repository in Amazon S3
- Set up end-to-end encryption for applications on Amazon EKS using cert-manager and Let's Encrypt
- Simplify Amazon EKS multi-tenant application deployment by using Flux
- Streamline machine learning workflows from local development to scalable experiments by using SageMaker AI and Hydra
- Structure a Python project in hexagonal architecture using AWS Lambda
- Test AWS infrastructure by using LocalStack and Terraform Tests
- Train and deploy a custom GPU-supported ML model on Amazon SageMaker
- Coordinate resource dependency and task execution by using the AWS Fargate WaitCondition hook construct
- Use Amazon Bedrock agents to automate creation of access entry controls in Amazon EKS through text-based prompts

Serverless

Topics

- Build a serverless React Native mobile app by using AWS Amplify
- Manage tenants across multiple SaaS products on a single control plane
- Consolidate Amazon S3 presigned URL generation and object downloads by using an endpoint associated with static IP addresses
- Create a cross-account Amazon EventBridge connection in an organization
- Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Firehose with AWS CDK
- Implement path-based API versioning by using custom domains in Amazon API Gateway
- Import the psycopg2 library to AWS Lambda to interact with your PostgreSQL database
- Integrate Amazon API Gateway with Amazon SQS to handle asynchronous REST APIs
- Process events asynchronously with Amazon API Gateway and AWS Lambda
- Process events asynchronously with Amazon API Gateway and Amazon DynamoDB Streams
- Process events asynchronously with Amazon API Gateway, Amazon SQS, and AWS Fargate
- Run AWS Systems Manager Automation tasks synchronously from AWS Step Functions
- Run parallel reads of S3 objects by using Python in an AWS Lambda function
- Send telemetry data from AWS Lambda to OpenSearch for real-time analytics and visualization
- Set up a serverless cell router for a cell-based architecture
- Set up private access to an Amazon S3 bucket through a VPC endpoint
- Troubleshoot states in AWS Step Functions by using Amazon Bedrock
- Chain AWS services together using a serverless approach
- More patterns

Build a serverless React Native mobile app by using AWS Amplify

Deekshitulu Pentakota, Amazon Web Services

Summary

This pattern shows how to create a serverless backend for a React Native mobile app by using AWS Amplify and the following AWS services:

- AWS AppSync
- Amazon Cognito
- Amazon DynamoDB

After you configure and deploy the app’s backend by using Amplify, Amazon Cognito authenticates app users and authorizes them to access the app. AWS AppSync then interacts with the frontend app and with a backend DynamoDB table to create and fetch data.

> **Note**
>
> This pattern uses a simple "ToDoList" app as an example, but you can use a similar procedure to create any React Native mobile app.

Prerequisites and limitations

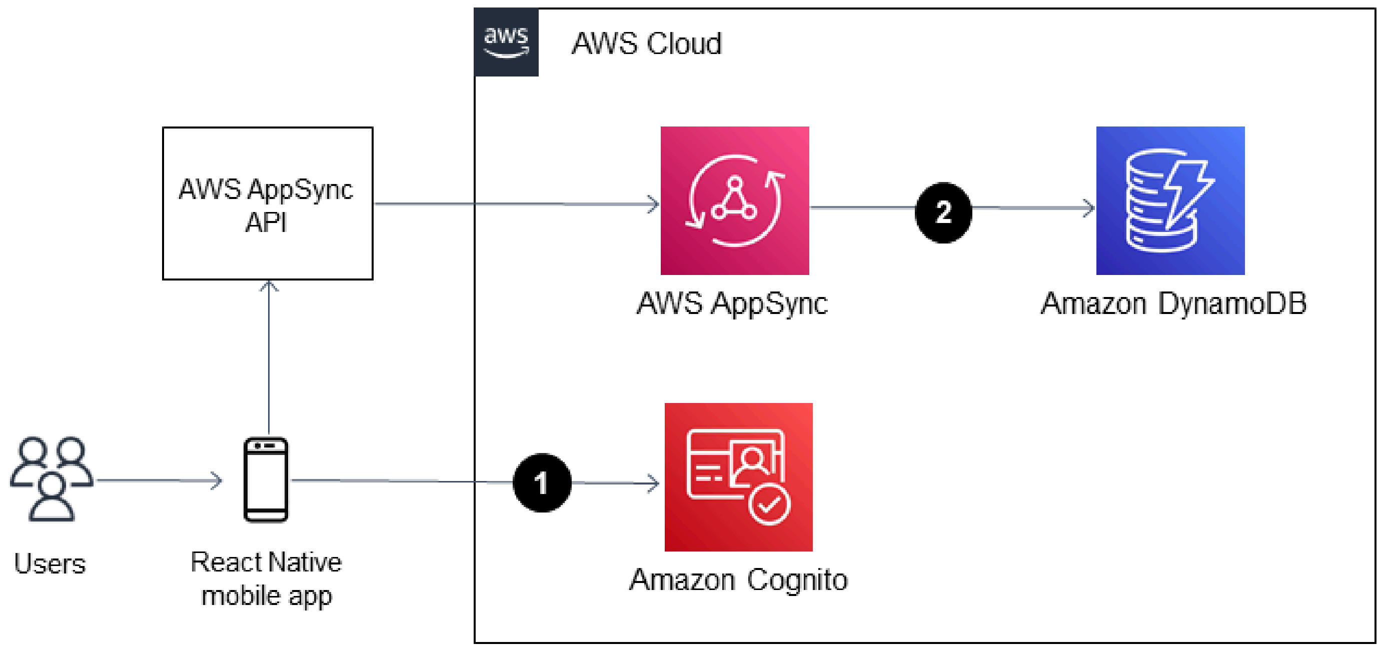
Prerequisites

- An active AWS Account
- Amplify Command Line Interface (Amplify CLI), installed and configured
- XCode (any version)
- Microsoft Visual Studio (any version, any code editor, any text editor)
- Familiarity with Amplify
- Familiarity with Amazon Cognito
- Familiarity with AWS AppSync
- Familiarity with DynamoDB
- Familiarity with Node.js

- Familiarity with npm
- Familiarity with React and React Native
- Familiarity with JavaScript and ECMAScript 6 (ES6)
- Familiarity with GraphQL

Architecture

The following diagram shows an example architecture for running a React Native mobile app's backend in the AWS Cloud:



The diagram shows the following architecture:

1. Amazon Cognito authenticates app users and authorizes them to access the app.
2. To create and fetch data, AWS AppSync uses a GraphQL API to interact with the frontend app and a backend DynamoDB table.

Tools

AWS services

- AWS Amplify is a set of purpose-built tools and features that helps frontend web and mobile developers quickly build full-stack applications on AWS.

- AWS AppSync provides a scalable GraphQL interface that helps application developers combine data from multiple sources, including Amazon DynamoDB, AWS Lambda, and HTTP APIs.
- Amazon Cognito provides authentication, authorization, and user management for web and mobile apps.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.

Code

The code for the sample application that's used in this pattern is available in the GitHub aws-amplify-react-native-ios-todo-app repository. To use the sample files, follow the instructions in the **Epics** section of this pattern.

Epics

Create and run your React Native app

Task	Description	Skills required
Set up a React Native development environment.	For instructions, see Setting up the development environment in the React Native documentation.	App developer
Create and run the ToDoList React Native mobile app in iOS Simulator.	1. Create a new React Native mobile app project directory in in your local environment by running the following command in a new terminal window: `npx react-native init ToDoListAmplify` 2. Navigate to the project's root directory by running the following command:	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>cd ToDoListAmplify</code></pre>
3. Run the app by running the following command:
<pre><code>npx react-native
run-ios</code></pre></td><td></td></tr>
</table>

Initialize a new backend environment for the app

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create the backend services needed to support the app in Amplify.</td><td>1. In your local environment, run the following command from the project’s root directory (ToDoListAmplify):
<pre><code>amplify init</code></pre>
2. A prompt appears that asks you to provide information about the app. Enter the required information based on your use case. Then, press Enter.

For the ToDoList app setup used in this pattern, apply the following example configuration.

Example React Native Amplify app configuration settings</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
	``` ? Name: ToDoListAmplify ? Environment: dev ? Default editor: Visual Studio Code ? App type: javascript ? Javascript framework: react-native ? Source Directory Path: src ? Distribution Directory Path: / ? Build Command: npm run-script build ? Start Command: npm run-script start ? Select the authentication method you want to use: AWS profile ? Please choose the profile you want to use: default ``` For more information, see Create a new Amplify backend in the Amplify Dev Center documentation.	

Task	Description	Skills required
	Note The `amplify init` command provisions the following resources by using [AWS CloudFormation]():	
	• AWS Identity and Access Management (IAM) roles for authenticated and unauthenticated users (**Auth Role** and **Unauth Role**) • An Amazon Simple Storage Service (Amazon S3) bucket for deployment (for this pattern's example app, **Amplify-meta.json**) • A backend environment in [Amplify Hosting]()	

Add Amazon Cognito authentication to your Amplify React Native app

Task	Description	Skills required
Create an Amazon Cognito authentication service.	1. In your local environment, run the following command from the project's root directory (**ToDoListAmplify**): `amplify add auth`	App developer

Task	Description	Skills required

	2. A prompt appears that asks you to provide information about the authentication service's configuration settings. Enter the required information based on your use case. Then, press **Enter**.
	
	For the ToDoList app setup used in this pattern, apply the following example configuration.
	
	Example authentication service configuration settings
	
	`? Do you want to use the default authentication and security configuration? \`
	`Default configuration`
	
	`? How do you want users to be able to sign in? \`
	`Username`
	
	`? Do you want to configure advanced settings? \`
	`No, I am done`
	
	Note
	The `amplify add auth` command	

Task	Description	Skills required
	creates the necessary folders, files, and dependency files in a local folder (**amplify**) within the project's root directory. For the ToDoList app setup used in this pattern, the **aws-exports.js** is created for this purpose.	

Deploy the Amazon Cognito service to the AWS Cloud.	1. From the project's root directory, run the following Amplify CLI command:
	`amplify push`
	2. A prompt to confirm deployment appears. Enter **Yes**. Then, press **Enter**.
	
	Note
	To see the deployed services in your project, go to the Amplify console by running the following command:
	
	`amplify console`
		App developer

Task	Description	Skills required

Install the required Amplify libraries for React Native and the CocoaPods dependencies for iOS.	1. Install the required Amplify open-source client libraries by running the following command from the project's root directory:
	`npm install aws-amplify aws-amplify-react-native \ amazon-cognito-identity-js @react-native-community/netinfo \ @react-native-async-storage/async-storage`
	2. Install the required CocoaPods dependencies for iOS by running the following command:
	`npx pod-install`
		App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Import and configure the Amplify service.</td><td>In the app’s entry point file (for example, App.js), import and load the Amplify service’s configuration file by entering the following lines of code:
<pre>import Amplify from
 'aws-amplify'
import config from './
src/aws-exports'
Amplify.configur
e(config)</pre>
Note
If you receive an error after importing the Amplify service in the app’s entry point file, stop the app. Then, open XCode and select the ToDoListAmplify.xcworkspace from the project’s iOS folder and run the app.</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
Update your app's entry point file to use the withAuthe nticator Higher-order component (HOC).	**Note** The `withAuthenticator` HOC provides sign-in, sign-up, and forgot password workflows in your app by using only a few lines of code. For more information, see [Option 1: Use pre-build UI components](#) in the Amplify Dev Center. Also, [Higher-order components](#) in the React documentation. 1. In the app's entry point file (for example, **App.js**), import the `withAuthenticator` HOC by entering the following lines of code: `import { withAuthenticator } from 'aws-amplify-react-native'` 2. Export the withAuthe nticator HOC by entering the following code:	App developer

Task	Description	Skills required

	`export default withAuthenticator(App)`
	
	withAuthenticator HOC code example
	
	`import Amplify from 'aws-amplify'`
	`import config from './src/aws-exports'`
	`Amplify.configure(config)`
	`import { withAuthenticator } from 'aws-amplify-react-native';`
	
	`const App = () => {`
	`  return null;`
	`};`
	
	`export default withAuthenticator(App);`
	
	Note
	In iOS Simulator, the app shows the login screen that's provided by the Amazon Cognito service.	

Task	Description	Skills required
Test the authentication service setup.	In iOS Simulator, do the following: 1. Create a new account in the app by using a real email address. A verificat ion code is then sent to the registered email. 2. Verify the account set up by using the code that you receive in the verification email. 3. Enter the username and password that you created. Then, choose **Sign In**. A welcome screen appears. **Note** You can also open the [Amazon Cognito console] and check if a new user has been created in the **Identity Pool** or not.	App developer

Connect an AWS AppSync API and DynamoDB database to the app

Task	Description	Skills required
Create an AWS AppSync API and DynamoDB database.	1. Add an AWS AppSync API to your app and automatic ally provision a DynamoDB	App developer

Task	Description	Skills required

	database by running the following Amplify CLI command from the project's root directory:
	`amplify add api`
	2. A prompt appears that asks you to provide information about the API and database configuration settings. Enter the required information based on your use case. Then, press **Enter**. The Amplify CLI opens the GraphQL schema file in your text editor.
	
	For the ToDoList app setup used in this pattern, apply the following example configuration.
	
	Example API and database configuration settings
	`? Please select from one of the below mentioned services: \`
	`GraphQL`
	
	`? Provide API name: todolistamplify`
	
	`? Choose the default authorization type for the API \`
	`Amazon Cognito User Pool`	

Task	Description	Skills required

	<pre>Do you want to use the
	 default authentication
	 and security configura
	tion
	
	? Default configuration
	 How do you want users
	 to be able to sign in?
	 \
	Username
	
	Do you want to configure
	 advanced settings? \
	No, I am done.
	
	? Do you want to
	 configure advanced
	 settings for the
	 GraphQL API \
	No, I am done.
	
	? Do you have an
	 annotated GraphQL
	 schema? \
	No
	
	? Choose a schema
	 template: \
	Single object with
	 fields (e.g., "Todo"
	 with ID, name,
	 description)
	
	? Do you want to edit
	 the schema now? \
	Yes</pre> **Example GraphQL schema** <pre> type Todo @model {</pre>	

Task	Description	Skills required

	``
	    id: ID!
	    name: String!
	    description: String
	}
	``	

Task	Description	Skills required
Deploy the AWS AppSync API.	1. In the project's root directory, run the following Amplify CLI command: `amplify push` 2. A prompt appears that asks you to provide more information about the API and database configuration settings. Enter the required information based on your use case. Then, press **Enter**. Your app can now interact with the AWS AppSync API. For the ToDoList app setup used in this pattern, apply the following example configura tion. **Example AWS AppSync API configuration settings** **Note** The following configuration creates the GraphQL API in AWS AppSync and a **Todo** table in Dynamo DB.	App developer

Task	Description	Skills required

	<pre>? Are you sure you want to continue? Yes
	? Do you want to generate code for your newly created GraphQL API Yes
	? Choose the code generation language target javascript
	? Enter the file name pattern of graphql queries, mutations and subscriptions src/graphql/**/*.js
	? Do you want to generate/update all possible GraphQL operations - \
	queries, mutations and subscriptions Yes
	? Enter maximum statement depth \
	[increase from default if your schema is deeply nested] 2</pre>	

Task	Description	Skills required
Connect the app's frontend to the AWS AppSync API.	To use the example ToDoList app provided in this pattern, copy the code from the **App.js** file in the aws-amplify-react-native-ios-todo-app GitHub repository. Then, integrate the example code into your local environment. The example code provided in the repository’s **App.js** file does the following: • Shows the form for creating a **ToDo Item** with **Title** and **Description** fields • Displays the list of to-do items (**Title** and **Descripti on**) • Posts and fetches data by using `aws-amplify` methods	App developer

Related resources

- AWS Amplify
- Amazon Cognito
- AWS AppSync
- Amazon DynamoDB
- React (React documentation)

Manage tenants across multiple SaaS products on a single control plane

Ramanna Avancha, Kishan Kavala, Anusha Mandava, and Jenifer Pascal, Amazon Web Services

Summary

This pattern shows how to manage tenant lifecycles across multiple software as a service (SaaS) products on a single control plane in the AWS Cloud. The reference architecture provided can help organizations reduce the implementation of redundant, shared features across their individual SaaS products and provide governance efficiencies at scale.

Large enterprises can have multiple SaaS products across various business units. These products often need to be provisioned for use by external tenants at different subscription levels. Without a common tenant solution, IT administrators must spend time managing undifferentiated features across multiple SaaS APIs, instead of focusing on core product feature development.

The common tenant solution provided in this pattern can help centralize the management of many of an organization's shared SaaS product features, including the following:

- Security
- Tenant provisioning
- Tenant data storage
- Tenant communications
- Product management
- Metrics logging and monitoring

Prerequisites and limitations

Prerequisites

- An active AWS account
- Knowledge of Amazon Cognito or a third-party identity provider (IdP)
- Knowledge of Amazon API Gateway
- Knowledge of AWS Lambda
- Knowledge of Amazon DynamoDB

- Knowledge of AWS Identity and Access Management (IAM)
- Knowledge of AWS Step Functions
- Knowledge of AWS CloudTrail and Amazon CloudWatch
- Knowledge of Python libraries and code
- Knowledge of SaaS APIs, including the different types of users (organizations, tenants, administrators, and application users), subscription models, and tenant isolation models
- Knowledge of your organization's multi-product SaaS requirements and multi-tenant subscriptions

Limitations

- Integrations between the common tenant solution and individual SaaS products aren't covered in this pattern.
- This pattern deploys the Amazon Cognito service in a single AWS Region only.

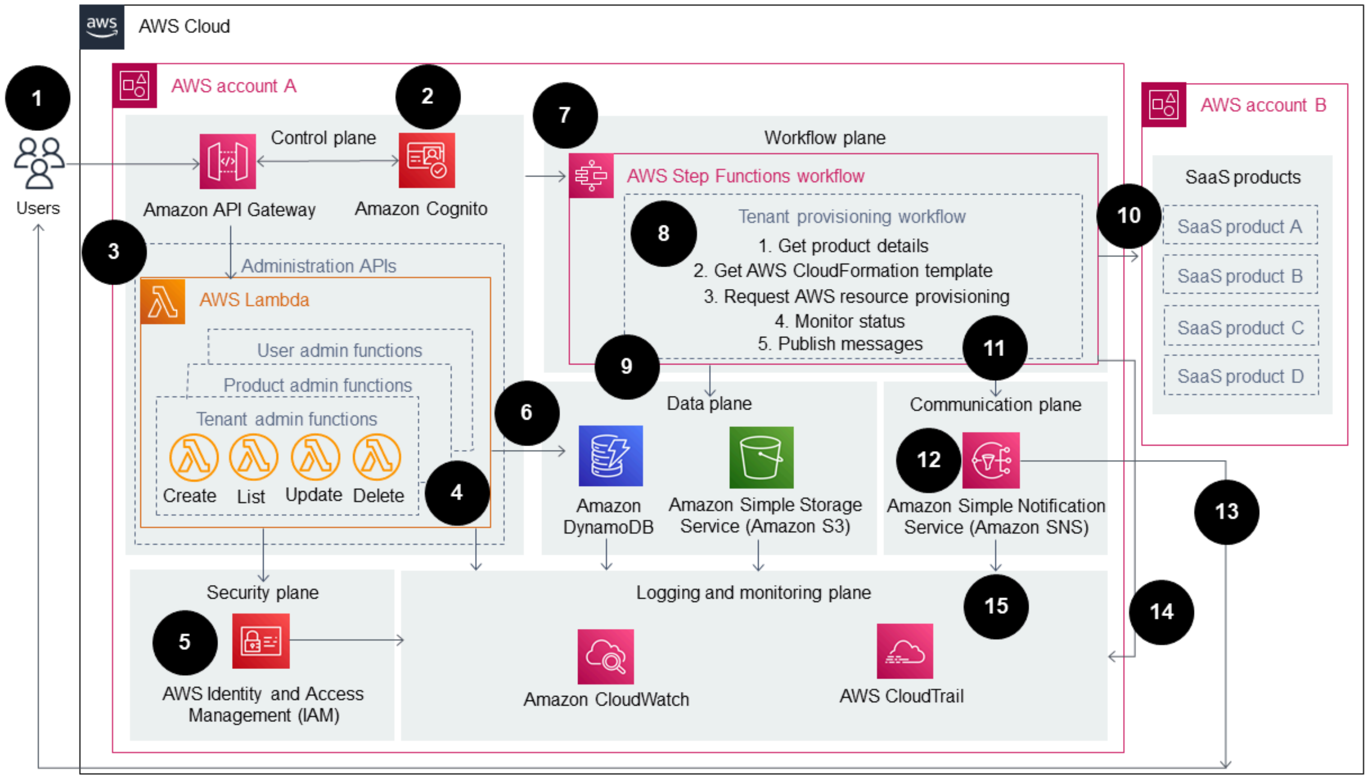
Architecture

Target technology stack

- Amazon API Gateway
- Amazon Cognito
- AWS CloudTrail
- Amazon CloudWatch
- Amazon DynamoDB
- IAM
- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon Simple Notification Service (Amazon SNS)
- AWS Step functions

Target architecture

The following diagram shows an example workflow for managing tenant lifecycles across multiple SaaS products on a single control plane in the AWS Cloud.



The diagram shows the following workflow:

1. An AWS user initiates tenant provisioning, product provisioning, or administration-related actions by making a call to an API Gateway endpoint.
2. The user is authenticated by an access token that's retrieved from an Amazon Cognito user pool, or another IdP.
3. Individual provisioning or administration tasks are run by Lambda functions that are integrated with API Gateway API endpoints.
4. Administration APIs for the common tenant solution (for tenants, products, and users) gather all of the required input parameters, headers, and tokens. Then, the administration APIs invoke the associated Lambda functions.
5. IAM permissions for both the administration APIs and the Lambda functions are validated by the IAM service.
6. Lambda functions store and retrieve data from the catalogs (for tenants, products, and users) in DynamoDB and Amazon S3.
7. After permissions are validated, an AWS Step Functions workflow is invoked to perform a specific task. The example in the diagram shows a tenant provisioning workflow.

8. Individual AWS Step Functions workflow tasks are run in a predetermined workflow (state machine).
9. Any essential data that's needed to run the Lambda function associated with each workflow task is retrieved from either DynamoDB or Amazon S3. Other AWS resources might need to be provisioned by using an AWS CloudFormation template.
10. If needed, the workflow sends a request to provision additional AWS resources for a specific SaaS product to that product's AWS account.
11. When the request succeeds or fails, the workflow publishes the status update as a message to an Amazon SNS topic.
12. Amazon SNS is subscribed to the Step Functions workflow's Amazon SNS topic.
13. Amazon SNS then sends the workflow status update back to the AWS user.
14. Logs of each AWS service's actions, including an audit trail of API calls, are sent to CloudWatch. Specific rules and alarms can be configured in CloudWatch for each use case.
15. Logs are archived in Amazon S3 buckets for auditing purposes.

Automation and scale

This pattern uses a CloudFormation template to help automate the deployment of the common tenant solution. The template can also help you quickly scale the associated resources up or down.

For more information, see Working with AWS CloudFormation templates in the *AWS CloudFormation User Guide*.

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- Amazon Cognito provides authentication, authorization, and user management for web and mobile apps.
- AWS CloudTrail helps you audit the governance, compliance, and operational risk of your AWS account.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

Best practices

The solution in this pattern uses a single control plane to manage the onboarding of multiple tenants and to provision access to multiple SaaS products. The control plane helps administrative users manage four other, feature-specific planes:

- Security plane
- Workflow plane
- Communication plane
- Logging and monitoring plane

Epics

Configure the security plane

Task	Description	Skills required
Establish the requirements for your multi-tenant SaaS platform.	Establish detailed requireme nts for the following:	Cloud architect, AWS systems administrator

Task	Description	Skills required
	• Tenants • Users • Roles • SaaS products • Subscriptions • Profile exchanges	
Set up the Amazon Cognito service.	Follow the instructions in Getting started with Amazon Cognito in the *Amazon Cognito Developer Guide*.	Cloud architect
Configure the required IAM policies.	Create the required IAM policies for your use case. Then, map the policies to IAM roles in Amazon Cognito. For more information, see Managing access using policies and Role-based access control in the *Amazon Cognito Developer Guide*.	Cloud administrator, Cloud architect, AWS IAM security

Task	Description	Skills required
Configure the required API permissions.	Set up API Gateway access permissions by using IAM roles and policies, and Lambda authorizers. For instructions, see the following sections of the *Amazon API Gateway Developer Guide*: • Control access to an API with IAM permissions • Use API Gateway Lambda authorizers	Cloud administrator, Cloud architect

Configure the data plane

Task	Description	Skills required
Create the required data catalogs.	1. Create DynamoDB tables to store data for the user catalogs. Make sure that you include user attribute s and roles. Also, make sure that you perform data modeling on the catalog tables to maintain the required and optional attributes for each user and role. 2. Create DynamoDB tables to store data for the product catalogs. Make sure that you model the	DBA

Task	Description	Skills required
	specific use cases for your SaaS products. 3. Create DynamoDB tables to store data for the tenant catalogs. Make sure that you set up subscript ion models for tenants, products and licensing for multi-SaaS subscriptions, and tags. For more information, see Setting up DynamoDB in the *Amazon DynamoDB Developer Guide*.	

Configure the control plane

Task	Description	Skills required
Create Lambda functions and API Gateway APIs to run required control plane tasks.	Create separate Lambda functions and API Gateway APIs to add, delete, and manage the following: • Users • Tenants • Products For more information, see Using AWS Lambda with Amazon API Gateway in the *AWS Lambda Developer Guide*.	App developer

Configure the workflow plane

Task	Description	Skills required
Identify the tasks that AWS Step Functions workflows must run.	Identify and document the detailed AWS Step Functions workflow requirements for the following: • Users • Tenants • Products **Important** Make sure that key stakeholders approve the requirements.	App owner
Create the required AWS Step Functions workflows.	1. Create the required workflows for users, tenants, and products in AWS Step Functions. For more information, see the *AWS Step Functions Developer Guide*. 2. Identify the retry and error handling mechanisms. For more information, see Handling errors, retries, and adding alerting to Step Function State Machines on the AWS Blog. 3. Implement the workflow steps by using Lambda	App developer, Build lead

Task	Description	Skills required
	functions. For instructions, see [Creating a Step Functions state machine that uses Lambda](#) in the *AWS Step Functions Developer Guide*. 4. Integrate any external services with AWS Step Functions as needed. 5. Maintain each workflow's status in a DynamoDB table and communicate each workflow's status by using Amazon SNS.	

Configure the communication plane

Task	Description	Skills required
Create Amazon SNS topics.	Create Amazon SNS topics to receive notifications about the following: • Workflow statuses • Errors • Retries For more information, see [Creating an SNS topic](#) in the *Amazon SNS Developer Guide*.	App owner, Cloud architect
Subscribe endpoints to each Amazon SNS topic.	To receive messages published to an Amazon SNS	App developer, Cloud architect

Task	Description	Skills required
	topic, you must subscribe an endpoint to each topic. For more information, see [Subscribing to an Amazon SNS topic] in the *Amazon SNS Developer Guide*.	

Configure the logging and monitoring plane

Task	Description	Skills required
Activate logging for each component of the common tenant solution.	Activate logging at the component level for each resource in the common tenant solution that you created. For instructions, see the following: • How do I turn on CloudWatch Logs for troubleshooting my API Gateway REST API or WebSocket API? (AWS Knowledge Center) • Logging using CloudWatch Logs (*AWS Step Functions Developer Guide*) • AWS Lambda function logging in Python (*AWS Lambda Developer Guide*)	App developer, AWS systems administrator, Cloud administrator

Task	Description	Skills required
	• Logging and monitoring in Amazon Cognito (*Amazon Cognito Developer Guide*) • Monitoring with Amazon CloudWatch (*Amazon DynamoDB Developer Guide*) **Note** You can consolidate logs for each resource into a centralized logging account by using IAM policies. For more information, see Centralized logging and multiple-account security guardrails.	

Provision and deploy the common tenant solution

Task	Description	Skills required
Create CloudFormation templates.	Automate the deployment and maintenance of the full common tenant solution and all its components by using CloudFormation templates. For more information, see the *AWS CloudFormation User Guide*.	App developer, DevOps engineer, CloudFormation developer

Related resources

- Control access to a REST API using Amazon Cognito user pools as authorizer (*Amazon API Gateway Developer Guide*)
- Use API Gateway Lambda authorizers (*Amazon API Gateway Developer Guide*)
- Amazon Cognito user pools (*Amazon Cognito Developer Guide*)
- Cross-account cross-Region CloudWatch console (*Amazon CloudWatch User Guide*)

Consolidate Amazon S3 presigned URL generation and object downloads by using an endpoint associated with static IP addresses

Song Jin, Eunhye Jo, and Jun Soung Lee, Amazon Web Services

Summary

This pattern simplifies access to Amazon Simple Storage Service (Amazon S3) by creating secure, custom presigned URLs for object downloads. The solution provides a single endpoint with a unique domain and static IP addresses. It's tailored for customers who require consolidation of both API and Amazon S3 endpoints under a unified domain with static IP addresses. The use case involves users following an IP and domain allowlist firewall policy, limiting API access to specific domains and IP addresses.

The architecture employs key AWS services, including AWS Global Accelerator, Amazon API Gateway, AWS Lambda, Application Load Balancer, AWS PrivateLink, and Amazon S3. This design centralizes the API for generating presigned URLs and the Amazon S3 endpoint under a single domain, linked to an accelerator with two static IP addresses. Consequently, users can effortlessly request presigned URLs and download Amazon S3 objects through a unified domain endpoint with static IP addresses.

This architecture is especially beneficial for customers with strict policies or compliance requirements, such as those in the public, medical, and finance sectors.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A public hosted zone for your custom domain name
- A domain imported in AWS Certificate Manager (ACM) in the AWS Region of your choice

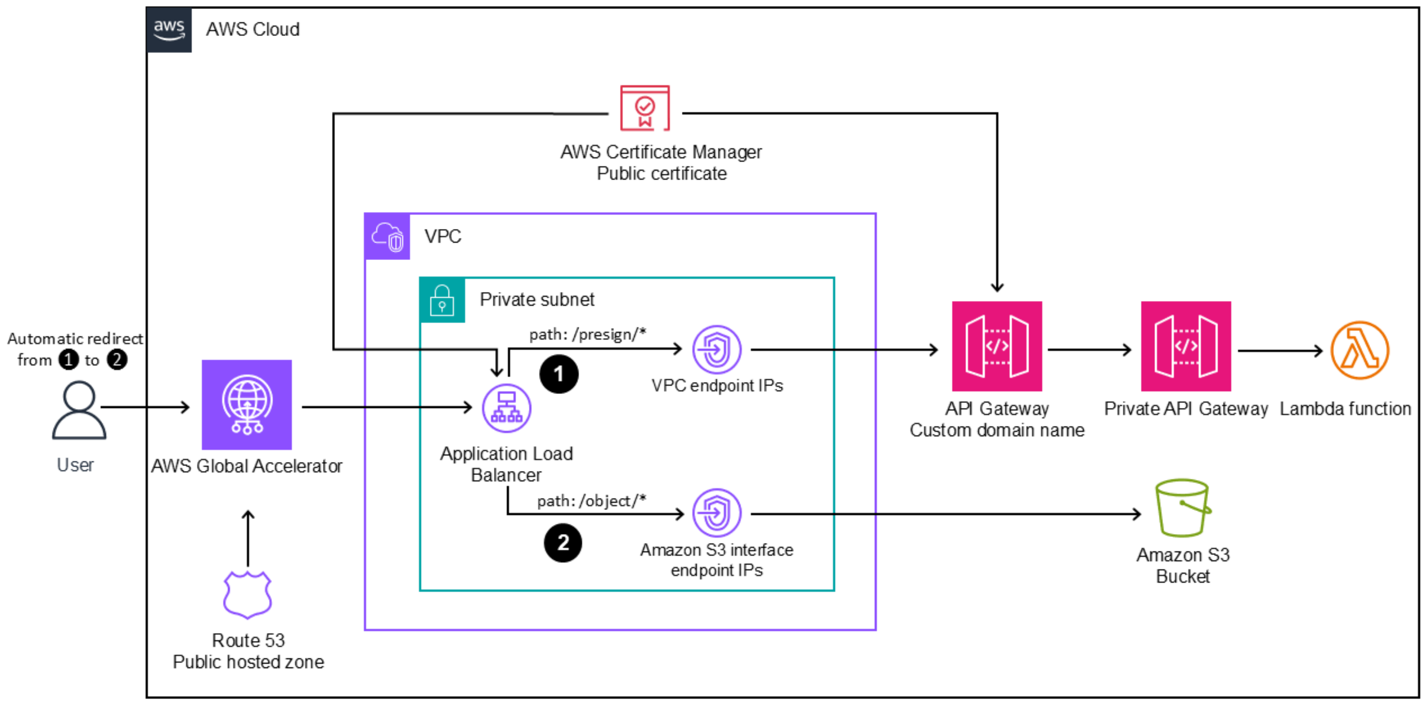
Limitations

- The Amazon S3 bucket name must match the domain name of the endpoint. This requirement is to ensure that the Amazon S3 endpoint can be served through the single API endpoint.
- The custom domain name used in API Gateway should align with the domain name of the single API endpoint.

- Some AWS services aren't available in all AWS Regions. For Region availability, see [AWS Services by Region](). For specific endpoints, see [Service endpoints and quotas](), and choose the link for the service.

Architecture

The following diagram shows the target architecture and workflow for this pattern.



The diagram illustrates the following concept and workflow:

1. A user initiates a request to generate a presigned URL by using the custom endpoint served through AWS Global Accelerator, using the custom domain name and associated IP addresses.
2. A Lambda function generates the presigned URL, pointing to the custom endpoint. It responds with a 301 redirect that contains the generated presigned URL. Through the redirected presigned URL, the user downloads the object automatically by using the custom endpoint served through Global Accelerator.

The components of the overall architecture for presigned URL generation and object download workflow are as follows:

- Provisioning of static IP addresses by Global Accelerator.
- Registration of the accelerator's alias as an A record into the Amazon Route 53 public hosted zone with the custom domain name.
- Creation of an Amazon S3 bucket with a bucket name that matches the registered custom domain name.
- Creation of VPC endpoints for API Gateway and the Amazon S3 service.
- Configuration of an internal-facing Application Load Balancer to connect to Global Accelerator.
- Assignment of a custom domain name for API Gateway with an ACM certificate attached.
- Deployment of a private API Gateway integrated with a Lambda function.
- The Lambda function is equipped with an AWS Identity and Access Management (IAM) role attached (with GetObject permissions).

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- Application Load Balancers distribute incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, in multiple Availability Zones.
- AWS Certificate Manager (ACM) helps you create, store, and renew public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS Global Accelerator is a global service that supports endpoints in multiple AWS Regions. You can create accelerators that direct traffic to optimal endpoints over the AWS global network. This improves the availability and performance of your internet applications that are used by a global audience.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

- AWS PrivateLink helps you create unidirectional, private connections from your virtual private clouds (VPCs) to services outside of the VPC.
- Amazon Route 53 is a highly available and scalable DNS web service.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

You can deploy this pattern by using either the AWS CDK or Terraform based on your preference. The Epics section contains instructions for both deployment methods. The code for this pattern is available in the following GitHub repositories:

- **AWS CDK** – s3-presignedurl-staticips-endpoint-with-cdk
- **Terraform** – s3-presignedurl-staticips-endpoint-with-terraform

Best practices

- To enhance security in the production environment, it's crucial to implement authorization mechanisms, such as Amazon Cognito, to restrict access to the `PresignedUrl` generation API.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Prepare the environment

Task	Description	Skills required
Decide on a domain name.	Decide on a public domain name for the unified Amazon	AWS administrator, Network administrator

Task	Description	Skills required
	S3 endpoint. The domain name is also used as the Amazon S3 bucket name.	
Create a public hosted zone.	Create a public hosted zone in Amazon Route 53. Its domain name must match the domain name that's used in API Gateway.	AWS administrator, Network administrator
Prepare an SSL certificate.	Use AWS Certificate Manager (ACM) to request or import an SSL certificate for your web application domain.	AWS administrator, Network administrator

Deploy the pattern with Terraform

Task	Description	Skills required
Set up the Terraform development environment.	To set up the developme nt environment, do the following: 1. Install Terraform version 1.0 or later. 2. Clone the GitHub s3-presig nedurl-staticips-endpoint-with-terraform repository by running the following command in a terminal window: `git clone https://g ithub.com/aws-samp les/s3-presignedur`	AWS administrator, Cloud administrator

Task	Description	Skills required
	`l-staticips-endpoint-with-terraform.git`	

Task	Description	Skills required

Modify the `.tfvars` and `provider.tf` files.	1. **Create the required `.tfvars` files –** Create `apg.tfvars` files in both the `1.vpc_alb_ga` and `2.apigw_s3_lambda` directories. These files will contain your environment-specific variable values.
	• For `1.vpc_alb_ga/apg.tfvars`, create a file with the following template:
	`hcl`
	` environment_name = "test"`
	` domain = "your-domain.com"`
	` s3_bucket_prefix = "your-bucket-prefix"`
	` aws_region = "ap-northeast-2"`
	` # Change to your preferred region`
	• For `2.apigw_s3_lambda/apg.tfvars`, create a file with the following template (you'll need to update the `vpc_id` after completing step 1):
	`hcl`
	` vpc_id = "vpc-xxxxxxxxx"`
		AWS administrator, Cloud administrator

Task	Description	Skills required

	``
	# Update this with the VPC ID output from step 1
	environment_name = "test"
	domain = "your-domain.com"
	s3_bucket_prefix = "your-bucket-prefix"
	aws_region = "ap-northeast-2" # Change to your preferred region
	``
	2. **Modify the** `provider.tf` **files** – In both the `1.vpc_alb_ga` and `2.apigw_s3_lambda` folders, edit the `provider.tf` files to match your local AWS configuration:
	``
	hcl
	 provider "aws" {
	  region = "ap-northeast-2" # Change to your preferred region
	  profile = "apg" # Change to your AWS profile name
	}
	``
	Note the following:	

Task	Description	Skills required
	• Replace the placeholder values in the `.tfvars` files with your actual values. • After completing the first Terraform deployment (`1.vpc_alb_ga`), update the `vpc_id` value in `2.apigw_s3_lambda/apg.tfvars` with the VPC ID from the output. • Make sure that your AWS profile name in the `provider.tf` file matches an existing profile in your `~/.aws/credentials` file. • The domain value should match your Route 53 hosted zone domain. • The `s3_bucket_prefix` combined with domain will form your Amazon S3 bucket name (for example, `your-bucket-prefix.your-domain.com`). • If you change `aws_region`, you might also need to update the hardcoded availability zones in the `main.tf` files to match your chosen AWS Region.	

Task	Description	Skills required

Provision network resources.	To provision network resources, run the following commands:
	`cd ./2.vpc_alb_ga`
	`terraform init`
	`terraform plan --var-file=apg.tfvars`
	`terraform apply --var-file=apg.tfvars`
	During the `apply` command's execution, type **yes** when prompted.
		AWS administrator, Cloud administrator

Provision API Gateway, Amazon S3, and Lambda.	To provision network resources, use the following commands:
	`cd ./2.apigw_s3_lambda`
	`terraform init`
	`terraform plan --var-file=apg.tfvars`
	`terraform apply --var-file=apg.tfvars`
		AWS administrator, Cloud administrator

Deploy the pattern with AWS CDK

Task	Description	Skills required
Set up the AWS CDK development environment.	To set up the development environment, do the following: 1. Install the AWS CDK.	AWS administrator, Cloud administrator

Task	Description	Skills required
	2. Clone the GitHub [s3-presignedurl-staticips-endpoint-with-cdk](https://github.com/aws-samples/s3-presignedurl-staticips-endpoint-with-cdk) repository by running the following command in a terminal window: `git clone https://github.com/aws-samples/s3-presignedurl-staticips-endpoint-with-cdk.git`	

Task	Description	Skills required

Configure domain settings in the `config/index.ts` file.	To edit the options of the constant variable, use the following commands:
	
	`export const options = {`
	`    certificateArn: '{arn of the acm which created before}',`
	`    dnsAttr: {`
	`        zoneName: '{public hosted zone name}',`
	`        hostedZoneId: 'hosted zone Id',`
	`    },`
	`    domainNamePrefix: '{Prefix for the domain}',`
	`    presignPath: 'presign',`
	`    objectsPath: 'objects',`
	`};`
	
	In the commands, replace each placeholder with your own information:
	
	• Replace `'{arn of the acm which created before}'` with your ACM certificate Amazon Resource Name (ARN).
	• Replace `'{public hosted zone name}'` with your domain name.
		AWS administrator, Cloud administrator

Task	Description	Skills required
	• Replace `'hosted zone Id'` with your Route 53 hosted zone ID. • Replace `'{Prefix for the domain}'` with the domain prefix that you want to use. • Optionally, modify `'presign'` and `'objects'` paths if needed for your specific use case.	

Deploy the stacks.	To deploy two stacks, one for the virtual private cloud (VPC) and another for the application, use the following command:
	`$ npm install`
	`$ cdk synth`
	`$ cdk deploy --all`
		AWS administrator, Cloud administrator

Test the pattern

Task	Description	Skills required
Verify the IP addresses of the endpoint.	To verify that the domain for this pattern has static IP addresses, use the following command: `nslookup ${s3-bucket-prefix}.${domain}`	Network administrator

Task	Description	Skills required
Upload a test file that you can later download.	Upload the test file to the `'/objects'` folder in the Amazon S3 bucket.	AWS administrator, Cloud administrator
Invoke the API to generate a presigned URL.	To generate a presigned URL, call the URL from a browser or API client (for example, Postman) using the following format: `https://${s3-bucket-prefix}.${domain}/presign/objects/${uploaded-filename}` Replace the placeholder values in `${s3-bucket-prefix}` and `${domain}` with the values that you set in previous steps.	App owner
Check the result.	The expected result is that you should receive a 301 (Moved Permanently) redirect status code. This response will contain the presigned URL, which should automatically initiate the download of your test file.	Test engineer

Clean up with Terraform

Task	Description	Skills required
Destroy API Gateway, Amazon S3, and Lambda resources.	To delete resources, use the following commands: ```cd ./2.apigw_s3_lambda terraform init terraform plan --destroy --var-file=apg.tfvars terraform destroy --var-file=apg.tfvars```	AWS administrator, Cloud administrator
Destroy network resources.	To delete network resources, use the following commands: ```cd ./1.vpc_alb_ga terraform init terraform plan --destroy --var-file=apg.tfvars terraform destroy --var-file=apg.tfvars```	AWS administrator, Cloud administrator

Clean up with AWS CDK

Task	Description	Skills required
Destroy the stacks.	To destroy both the VPC and application stacks, use the following command: ```$ cdk destroy --all```	AWS administrator, Cloud administrator
Empty and delete the Amazon S3 buckets.	Empty and delete the object Amazon S3 bucket and the	AWS administrator, Cloud administrator

Task	Description	Skills required

	logs Amazon S3 bucket that are not deleted by default.
	
	The Amazon S3 bucket names are `${s3-bucket-prefix}.${domain}` and `${s3-bucket-prefix}.${domain}-logs` .
	
	If you prefer to use the [AWS Command Line Interface (AWS CLI)]() for deleting the buckets, use the following commands:
	
	`$ aws s3 rm s3://${s3-bucket-prefix}.${domain} --recursive`
	`$ aws s3 rb s3://${s3-bucket-prefix}.${domain} --force`
	`$ aws s3 rm s3://${s3-bucket-prefix}.${domain}-logs --recursive`
	`$ aws s3 rb s3://${s3-bucket-prefix}.${domain}-logs --force`
	
	Replace `${s3-bucket-prefix}` and `${domain}` with the values you set in previous steps.,/p>	

Related resources

AWS Blogs

- Accessing an Amazon API Gateway via static IP addresses provided by AWS Global Accelerator
- Generate a presigned URL in modular AWS CDK for JavaScript
- Hosting Internal HTTPS Static Websites with ALB, S3, and PrivateLink

Create a cross-account Amazon EventBridge connection in an organization

Sam Wilson and Robert Stone, Amazon Web Services

Summary

Large distributed systems use Amazon EventBridge to communicate changes in state between various Amazon Web Services (AWS) accounts in an AWS Organizations organization. However, EventBridge is generally able to target only endpoints or consumers in the same AWS account. The exception is an event bus in a different account. That event bus is a valid target. To consume events from an event bus in another account, the events must be pushed from the source account's event bus to the destination account’s event bus. To avoid challenges when managing critical events across applications within different AWS accounts, use the recommended approach presented in this pattern.

This pattern illustrates how to implement an event-driven architecture with EventBridge that involves multiple AWS accounts in an AWS Organizations organization. The pattern uses AWS Cloud Development Kit (AWS CDK) Toolkit and AWS CloudFormation.

EventBridge offers a serverless event bus that helps you receive, filter, transform, route, and deliver events. A critical component of event-driven architectures, EventBridge supports separation between producers of messages and consumers of those messages. In a single account, this is straight forward. A multi-account structure requires additional considerations for events on the event bus in one account to be consumed in other accounts within the same organization.

For information about account-specific considerations for producers and consumers, see the Additional information section.

Prerequisites and limitations

Prerequisites

- An AWS Organizations organization with at least two associated AWS accounts
- An AWS Identity and Access Management (IAM) role in both AWS accounts that allows you to provision infrastructure in both AWS accounts by using AWS CloudFormation
- Git installed locally
- AWS Command Line Interface (AWS CLI) installed locally
- AWS CDK installed locally and bootstrapped in both AWS accounts

Product versions

This pattern has been built and tested by using the following tools and versions:

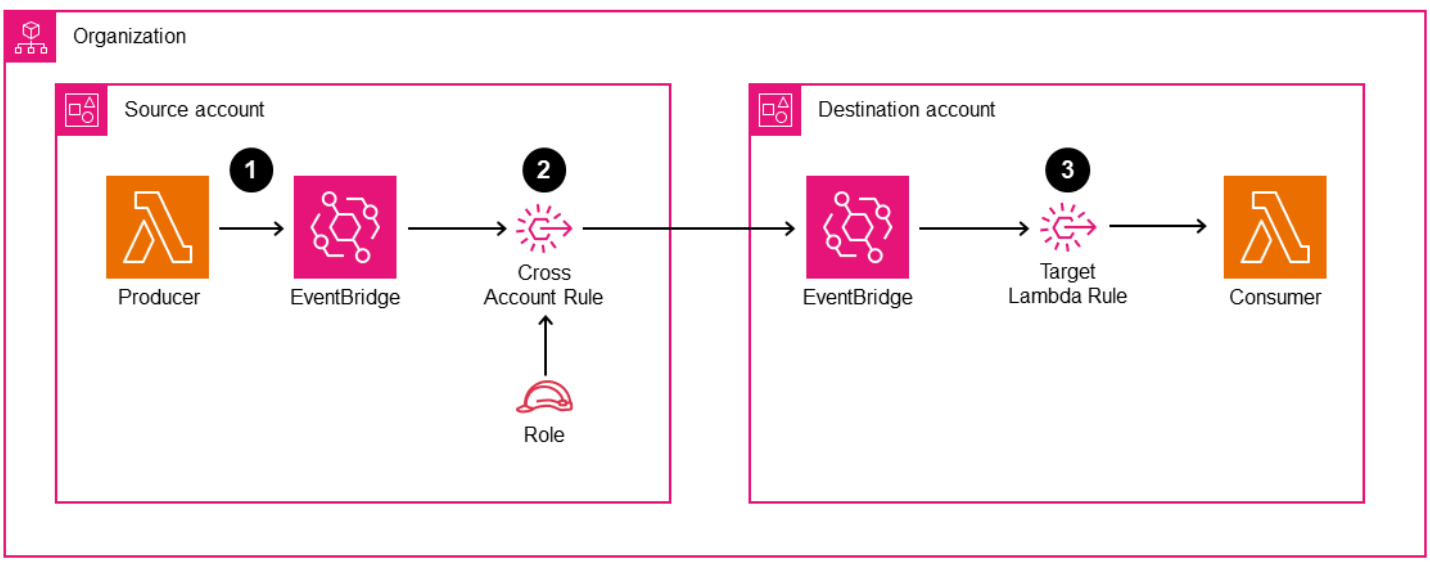
- AWS CDK Toolkit 2.126.0
- Node.js 18.19.0
- npm 10.2.3
- Python 3.12

This pattern should work with any version of AWS CDK v2 or npm. Node.js versions 13.0.0 through 13.6.0 are not compatible with AWS CDK.

Architecture

Target architecture

The following diagram shows the architecture workflow for pushing an event from one account and consuming it in another account.



The workflow contains the following steps:

1. The Producer AWS Lambda function in the Source account puts an event on the account's EventBridge event bus.
2. The cross-account EventBridge rule routes the event to an EventBridge event bus in the Destination account.

3. The EventBridge event bus in the Destination account has a target Lambda rule that invokes the Consumer Lambda function.

A best practice is to use a Dead Letter Queue (DLQ) for handling failed invocations of the Consumer Lambda function. However, the DLQ was omitted from this solution for clarity. To learn more about how to implement a DLQ in your workflows and improve your workflows' ability to recover from failures, see the Implementing AWS Lambda error handling patterns blog post.

Automation and scale

AWS CDK automatically provisions the required architecture. EventBridge can scale to thousands of records per second depending on the AWS Region. For more information, see the Amazon EventBridge quotas documentation.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code. This pattern uses the AWS CDK Toolkit, a command line cloud development kit that helps you interact with your AWS CDK app.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.

Other tools

- Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub cross-account-eventbridge-in-organization repository.

Best practices

For best practices when working with EventBridge, see the following resources:

- Best practices for Amazon EventBridge event patterns
- Best practices when defining rules in Amazon EventBridge

Epics

Prepare your local AWS CDK deployment environment

Task	Description	Skills required
Configure local credentials for the Source account and Destination account.	Review Setting up new configuration and credentials, and use the authentication and credential method that makes the most sense to your environment. **Important** Be sure to configure the AWS CLI for both Source account and Destination account authentication. These instructions assume that you have configured two AWS profiles locally:	App developer

Task	Description	Skills required
	`sourceAccount` and `destinationAccount` .	

Bootstrap both AWS accounts.	To bootstrap the accounts, run the following commands:
	`cdk bootstrap --profile sourceAccount`
	`cdk bootstrap --profile destinationAccount`
		App developer

Clone the pattern code.	To clone the repository, run the following command:
	`git clone git@github.com:aws-samples/aws-cdk-examples.git`
	Then, change the directory to the newly cloned project folder:
	`cd aws-cdk-examples/python/cross-account-eventbridge-in-organization`
		App developer

Deploy ProducerStack to the Source account

Task	Description	Skills required
Modify `cdk.json` with your AWS Organizations and account details.	In the root folder of the project, make the following changes to `cdk.json`:	App developer

Task	Description	Skills required
	• `organization_id` – The Organizations ID of the accounts involved in your deployment • `event_bus_name` – `CrossAccount`, or you can rename it • `rules[].targets[].arn` – The AWS account ID of the consuming account (Destination account)	
Deploy the ProducerStack resources.	Run the following command from the project's root directory: `cdk deploy ProducerStack --profile sourceAccount` When prompted, accept the new IAM roles and other security-related permissions created through AWS CloudFormation.	App developer

Task	Description	Skills required
Verify that ProducerStack resources are deployed.	To verify the resources, do the following: 1. On the AWS Management Console for the Source account, choose **CloudFormation**. 2. From the list of stacks, choose **ProducerStack**. 3. On the **Stack Info** tab, verify that the stack status is `CREATE_COMPLETE` . Optionally, on the **Resources** tab, review the configured resources.	App developer

Deploy ConsumerStack to the Destination account

Task	Description	Skills required
Deploy the ConsumerStack resources.	Run the following command from the project's root directory: `cdk deploy ConsumerStack --profile destinationAccount` When prompted, accept the new IAM roles and other security-related permissions created through AWS CloudFormation.	App developer

Task	Description	Skills required
Verify that ConsumerStack resources are deployed	1. On the console for the Destination account, choose **CloudFormation**. 2. From the list of stacks, choose **ConsumerStack**. 3. On the **Stack Info** tab, verify that the stack status is `CREATE_COMPLETE` . Optionally, on the **Resources** tab, review the configured resources.	App developer

Produce and consume events

Task	Description	Skills required
Invoke the Producer Lambda function.	1. On the console for the Source account, choose **Lambda**. 2. From the list of functions , choose **ProducerStack-ProducerLambdaXXXX** (XXXX represents a sequence of characters automatically generated by AWS CDK). 3. Choose the **Test** tab. 4. In the **Test event** section, choose **Test**. The Event JSON text area content can be any valid JSON that is provided to	App developer

Task	Description	Skills required
	the Lambda function as the payload. In this case, the default provided JSON is sufficient. 5. Verify that the **Executing function: succeeded** message appears in a green banner above the **Test event** section.	

Task	Description	Skills required
Verify that the event was received.	1. On the console for the Destination account, choose **Lambda**. 2. From the list of functions, choose **ConsumerStack-ConsumerLambdaXXXX** (XXXX represents a sequence of characters automatically generated by AWS CDK). 3. Choose the **Monitor** tab. 4. In the **Monitor** section, choose **View CloudWatch logs**. 5. In the newly opened tab, choose the log stream name of the most recent log stream. 6. Verify that a log statement such as the following appears: `[DEBUG] 2024-04-08T19:08:10.091Z 9c16844a-f9de-444d-b621-86afe64d4cc8 Event: {'version':'0', 'id':'0b9faa96-973f-8be2-ecf8-75e4f328b980', 'detail-type':'TestType', 'source':'Producer', 'account'`	App developer

Task	Description	Skills required
	`:'XXXXXXXXXXXX', 'time':'2024-04-08T19:08:09Z', 'region':'us-east-1', 'resources': [], 'detail': {'key1':'value1', 'key2':'value2', 'key3':'value3'}}`	

Cleanup

Task	Description	Skills required
Destroy the ConsumerStack resources.	If you are using this pattern as a test, clean up the deployed resources to avoid incurring additional costs. Run the following command from the project's root directory: `cdk destroy ConsumerStack --profile destinationAccount` You will be prompted to confirm deletion of the stack.	App developer
Destroy the ProducerStack resources.	Run the following command from the project's root directory:	App developer

Task	Description	Skills required
	`cdk destroy ProducerStack --profile sourceAccount` You will be prompted to confirm deletion of the stack.	

Troubleshooting

Issue	Solution
No event was received in the Destination account.	1. Verify that the provided Organizations ID is correct. 2. Verify that the Source account is part of the provided organization. 3. Verify that the event bus rules in the Source account map to the correct information in the Destination account.
Invoking a Lambda function from the console returns the following error:: `User: arn:aws:iam::123456789012:user/XXXXX is not authorized to perform: lambda:Invoke`	Contact your AWS account administrator to receive the appropriate `lambda:Invoke` action permissions on the `ProducerStack-ProducerLambdaXXXX` Lambda function.

Related resources

References

- AWS Organizations User Guide
- Amazon EventBridge event patterns
- Rules in Amazon EventBridge

Tutorials and videos

- Tutorial: Creating and configuration an organization
- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge (COM301-R)

Additional information

Producer rule

In the Source account, an EventBridge event bus is created to accept messages from producers (as shown in the *Architecture* section). A rule with accompanying IAM permissions is created on this event bus. The rules target the EventBridge event bus in the Destination account based on the following `cdk.json` structure:

```
"rules": [
  {
    "id": "CrossAccount",
    "sources": ["Producer"],
    "detail_types": ["TestType"],
    "targets": [
      {
        "id": "ConsumerEventBus",
        "arn": "arn:aws:events:us-east-2:012345678901:event-bus/CrossAccount"
      }
    ]
  }
]
```

For each consuming event bus, the event pattern and the target event bus must be included.

Event Pattern

Event patterns filter which events this rule will apply to. For purposes of this example, the event sources and the record `detail_types` identify which events to transmit from the Source account's event bus to the Destination account's event bus.

Target event bus

This rule targets an event bus that exists in another account. The full `arn` (Amazon Resource Name) is needed to uniquely identify the target event bus, and the `id` is the logical ID used by AWS CloudFormation. The target event bus need not actually exist at the time of target rule creation.

Destination account-specific considerations

In the Destination account, an EventBridge event bus is created to receive messages from the Source account's event bus. To allow events to be published from the Source account, you must create a resource-based policy:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowOrgToPutEvents",
    "Effect": "Allow",
    "Principal": "*",
    "Action": "events:PutEvents",
    "Resource": "arn:aws:events:us-east-2:012345678901:event-bus/CrossAccount",
    "Condition": {
      "StringEquals": {
        "aws:PrincipalOrgID": "o-XXXXXXXXX"
      }
    }
  }]
}
```

It's especially important to grant the `events:PutEvents` permission, which allows any other account in the same organization to publish events to this event bus. Setting `aws:PrincipalOrgId` as the organization ID grants the needed permissions.

Event pattern

You can modify the included event pattern to meet your use case:

```
rule = events.Rule(
    self,
    self.id + 'Rule' + rule_definition['id'],
    event_bus=event_bus,
    event_pattern=events.EventPattern(
        source=rule_definition['sources'],
        detail_type=rule_definition['detail_types'],
    )
)
```

To reduce unnecessary processing, the event pattern should specify that only events to be processed by the Destination account are transmitted to the Destination account's event bus.

Resource-based policy

This example uses the organization ID to control which accounts are allowed to put events on the Destination account's event bus. Consider using a more restrictive policy, such as specifying the Source account.

EventBridge quotas

Keep in mind the following quotas:

- 300 rules per event bus is the default quota. This can be expanded if necessary, but it should fit most use cases.
- Five targets per rule is the maximum allowed. We recommend that application architects should use a distinct rule for each Destination account to support fine-grained control over the event pattern.

Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Firehose with AWS CDK

Shashank Shrivastava and Daniel Matuki da Cunha, Amazon Web Services

Summary

This pattern provides sample code and an application for delivering records from Amazon DynamoDB to Amazon Simple Storage Service (Amazon S3) by using Amazon Kinesis Data Streams and Amazon Data Firehose. The pattern's approach uses AWS Cloud Development Kit (AWS CDK) L3 constructs and includes an example of how to perform data transformation with AWS Lambda before data is delivered to the target S3 bucket on the Amazon Web Services (AWS) Cloud.

Kinesis Data Streams records item-level modifications in DynamoDB tables and replicates them to the required Kinesis data stream. Your applications can access the Kinesis data stream and view the item-level changes in near-real time. Kinesis Data Streams also provides access to other Amazon Kinesis services, such as Firehose and Amazon Managed Service for Apache Flink. This means that you can build applications that provide real-time dashboards, generate alerts, implement dynamic pricing and advertising, and perform sophisticated data analysis.

You can use this pattern for your data integration use cases. For example, transportation vehicles or industrial equipment can send high volumes of data to a DynamoDB table. This data can then be transformed and stored in a data lake hosted in Amazon S3. You can then query and process the data and predict any potential defects by using serverless services such as Amazon Athena, Amazon Redshift Spectrum, Amazon Rekognition, and AWS Glue.

Prerequisites and limitations

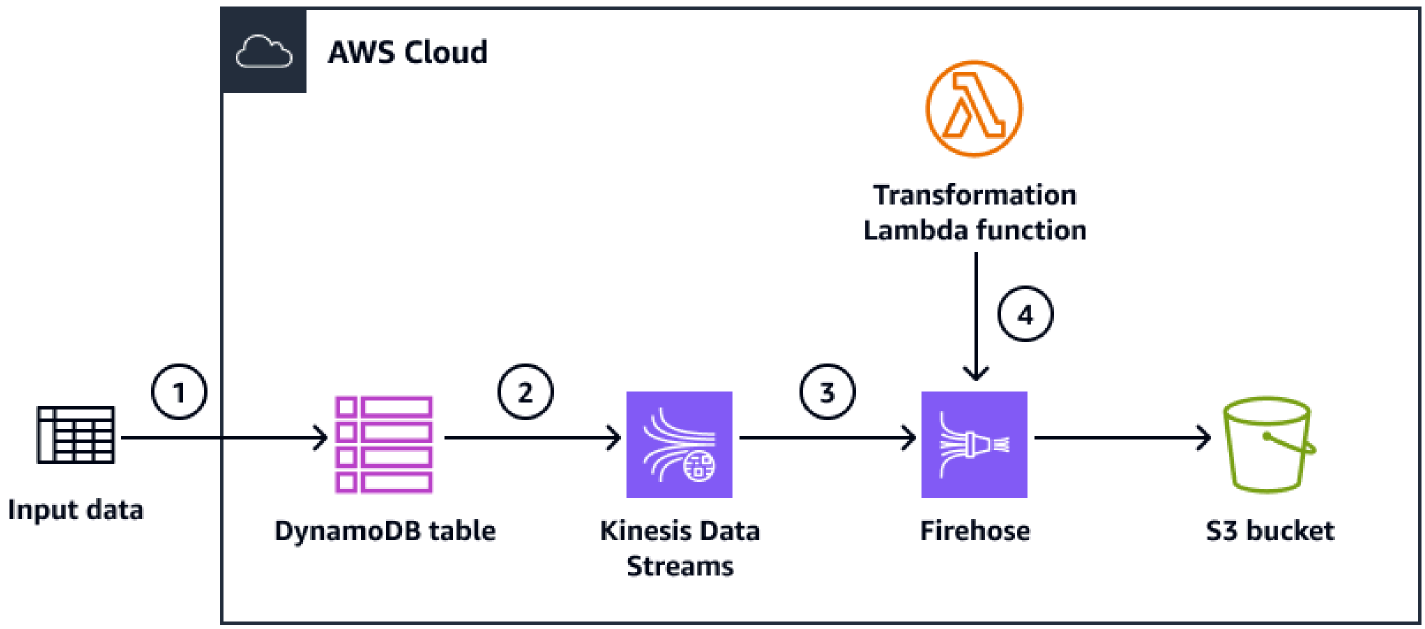
Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information, see Getting started with the AWS CLI in the AWS CLI documentation.
- Node.js (18.x+) and npm, installed and configured. For more information, see Downloading and installing Node.js and npm in the npm documentation.
- aws-cdk (2.x+), installed and configured. For more information, see Getting started with the AWS CDK in the AWS CDK documentation.

- The GitHub aws-dynamodb-kinesisfirehose-s3-ingestion repository, cloned and configured on your local machine.
- Existing sample data for the DynamoDB table. The data must use the following format: `{"SourceDataId": {"S": "123"},"MessageData":{"S": "Hello World"}}`

Architecture

The following diagram shows an example workflow for delivering records from DynamoDB to Amazon S3 by using Kinesis Data Streams and Firehose.



The diagram shows the following workflow:

1. Data is ingested using Amazon API Gateway as a proxy for DynamoDB. You can also use any other source to ingest data into DynamoDB.
2. Item-level changes are generated in near-real time in Kinesis Data Streams for delivery to Amazon S3.
3. Kinesis Data Streams sends the records to Firehose for transformation and delivery.
4. A Lambda function converts the records from a DynamoDB record format to JSON format, which contains only the record item attribute names and values.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CDK Toolkit is a command line cloud development kit that helps you interact with your AWS CDK app.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.

Code repository

The code for this pattern is available in the GitHub aws-dynamodb-kinesisfirehose-s3-ingestion repository.

Epics

Set up and configure the sample code

Task	Description	Skills required
Install the dependencies.	On your local machine, install the dependencies from the `package.json` files in the `pattern/aws-dynamodb-kinesisstreams-s3` and `sample-application` directories by running the following commands: `cd <project_root>/pattern/aws-dynamodb-kinesisstreams-s3`	App developer, General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

```
npm install && npm run
 build
```

```
cd <project_root>/sam
ple-application/
```

```
npm install && npm run
 build
```

</td><td></td></tr>
<tr><td>Generate the CloudFormation template.</td><td>1. Run the <code>cd <project_root>/sample-application/</code> command.
2. Run the <code>cdk synth</code> command to generate the CloudFormation template.
3. The <code>AwsDynamodbKinesisfirehoseS3IngestionStack.template.json</code> output is stored in the <code>cdk.out</code> directory.
4. Use AWS CDK or the AWS Management Console to process the template in CloudFormation.</td><td>App developer, General AWS, AWS DevOps</td></tr>
</table>

Deploy the resources

Task	Description	Skills required
Check and deploy the resources.	1. Run the `cdk diff` command to identify the resource types that are created by the AWS CDK construct. 2. Run the `cdk deploy` command to deploy the resources.	App developer, General AWS, AWS DevOps

Ingest data into the DynamoDB table to test the solution

Task	Description	Skills required

Ingest your sample data into the DynamoDB table.	Send a request to your DynamoDB table by running the following command in AWS CLI:
	`aws dynamodb put-item --table-name <your_table_name> --item '{"<table_partition_key>":{"S": "<partition_key_ID>"},"MessageData":{"S": "<data>"}}'`
	example:
	`aws dynamodb put-item --table-name SourceData_table`
		App developer

Task	Description	Skills required
	`--item '{"SourceDataId": {"S": "123"},"MessageData":{"S": "Hello World"}}'` By default, the `put-item` doesn't return any value as output if the operation succeeds. If the operation fails, it returns an error. The data is stored in DynamoDB and then sent to Kinesis Data Streams and Firehose. **Note** You use different approaches to add data into a DynamoDB table. For more information, see Load data into tables in the DynamoDB documentation.	

Task	Description	Skills required
Verify that a new object is created in the S3 bucket.	Sign in to the AWS Management Console and monitor the S3 bucket to verify that a new object was created with the data that you sent. For more information, see GetObject in the Amazon S3 documentation.	App developer, General AWS

Clean up resources

Task	Description	Skills required
Clean up resources.	Run the `cdk destroy` command to delete all the resources used by this pattern.	App developer, General AWS

Related resources

- s3-static-site-stack.ts (GitHub repository)
- aws-apigateway-dynamodb module (GitHub repository)
- aws-kinesisstreams-kinesisfirehose-s3 module (GitHub repository)
- Change data capture for DynamoDB Streams (DynamoDB documentation)
- Using Kinesis Data Streams to capture changes to DynamoDB (DynamoDB documentation)

Implement path-based API versioning by using custom domains in Amazon API Gateway

Corey Schnedl, Marcelo Barbosa, Mario Lopez Martinez, Anbazhagan Ponnuswamy, Gaurav Samudra, and Abhilash Vinod, Amazon Web Services

Summary

This pattern demonstrates how you can use the API mappings feature of custom domains to implement a path-based API versioning solution for Amazon API Gateway.

Amazon API Gateway is a fully managed service that you can use to create, publish, maintain, monitor, and secure APIs at any scale. By using the service's custom domain feature, you can create custom domain names that are simpler with more intuitive URLs that you can provide to your API users. You can use API mappings to connect API stages to a custom domain name. After you create a domain name and configure DNS records, you use API mappings to send traffic to your APIs through your custom domain name.

After an API becomes publicly available, consumers use it. As a public API evolves, its service contract also evolves to reflect new features and capabilities. However, it's unwise to change or remove existing features. Any breaking changes might impact the consumer's applications and break them at runtime. API versioning is important to avoid breaking backward compatibility and breaking a contract.

You need a clear strategy for API versioning to help consumers adopt them. Versioning APIs by using path-based URLs is the most straightforward and commonly used approach. In this type of versioning, versions are explicitly defined as part of API URIs. The following example URLs show how a consumer can use the URI to specify an API version for their request:

`https://api.example.com/api/v1/orders`

`https://api.example.com/api/v2/orders`

`https://api.example.com/api/vX/orders`

This pattern uses the AWS Cloud Development Kit (AWS CDK) to build, deploy, and test a sample implementation of a scalable path-based versioning solution for your API. AWS CDK is an open source software development framework to model and provision your cloud application resources using familiar programming languages.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Ownership of a domain is required to use this pattern's sample repository and to use Amazon API Gateway custom domain functionality. You can use Amazon Route 53 to create and manage your domains for your organization. For information about how to register or transfer a domain with Route 53, see Registering new domains in the Route 53 documentation.
- Before setting up a custom domain name for an API, you must have an SSL/TLS certificate ready in AWS Certificate Manager.
- You must create or update your DNS provider's resource record to map to your API endpoint. Without such a mapping, API requests bound for the custom domain name can't reach API Gateway.

Limitations

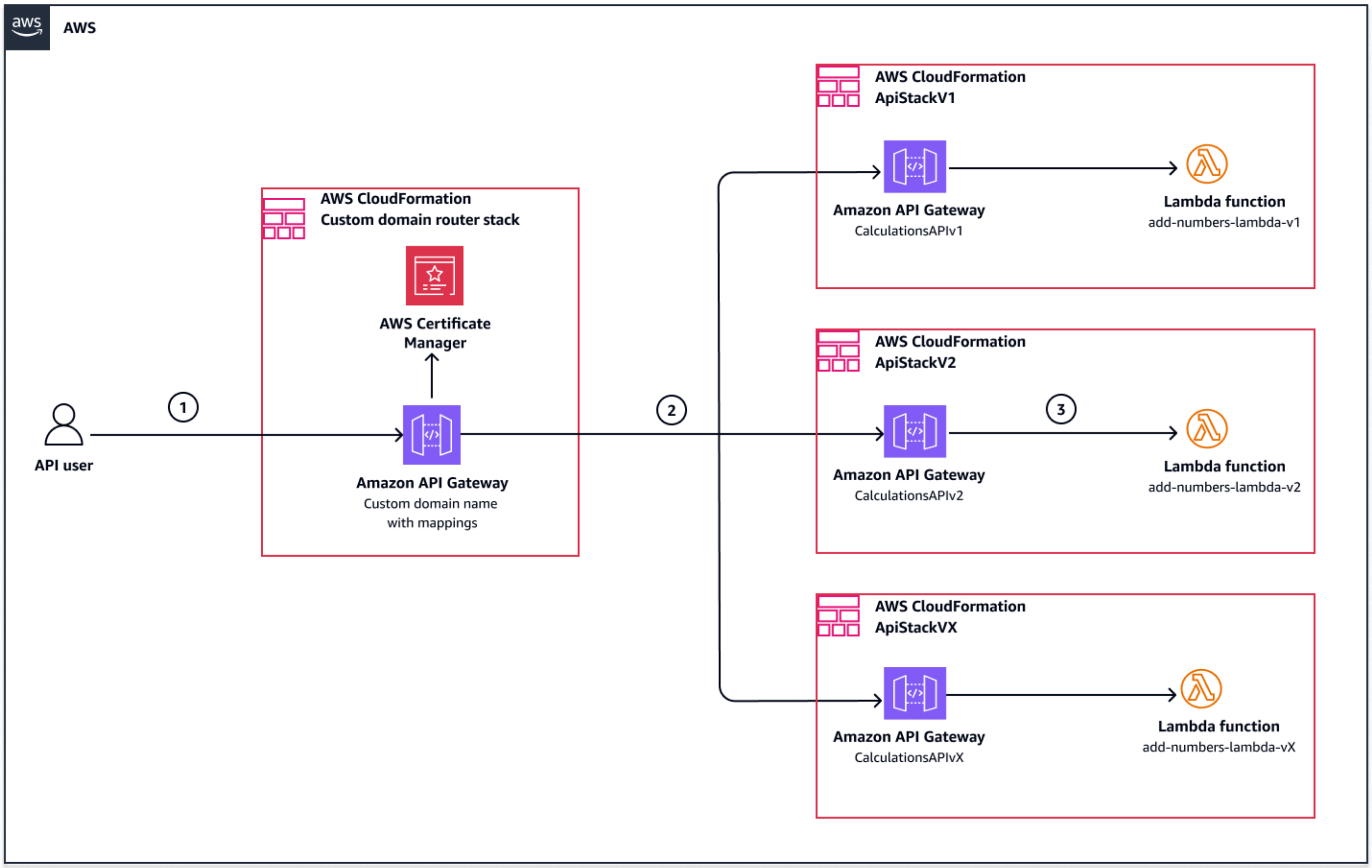
- Custom domain names are not supported for private APIs.
- A custom domain name must be unique within an AWS Region across all AWS accounts.
- To configure API mappings with multiple levels, you must use a Regional custom domain name and use the TLS 1.2 security policy.
- In an API mapping, the custom domain name and mapped APIs must be in the same AWS account.
- API mappings must contain only letters, numbers, and the following characters: `$-_.+!*'()/`
- The maximum length for the path in an API mapping is 300 characters.
- You can have 200 API mappings with multiple levels for each domain name.
- You can only map HTTP APIs to a Regional custom domain name with the TLS 1.2 security policy.
- You can't map WebSocket APIs to the same custom domain name as an HTTP API or REST API.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- This sample implementation uses AWS CDK in TypeScript version 2.149.0.

Architecture

The following diagram shows the architecture workflow.



The diagram illustrates the following:

1. The API user sends a request to Amazon API Gateway with a custom domain name.
2. API Gateway dynamically routes the user's request to an appropriate instance and stage of API Gateway, based on the path indicated in the URL of the request. The following table shows an example of how the different URL-based paths can be routed to specific stages for different instances of API Gateway.

API	Stage	Path	Default endpoint
CalculationAPIv1	api	apiv1	Enabled
CalculationAPIv2	api	apiv2	Enabled

CalculationAPIvX	api	apivX	Enabled

3. The destination API Gateway instance processes the request and returns the result to the user.

Automation and scale

We recommend that you use separate AWS CloudFormation stacks for each version of your API. With this approach, you can have complete isolation between the backend APIs that can be routed to by the custom domain API mapping feature. An advantage of this approach is that different versions of your API can be deployed or removed independently without introducing the risk of modifying another API. This approach increases resilience through isolation of CloudFormation stacks. Also, it provides you with different back-end options for your API such as AWS Lambda, AWS Fargate, HTTP endpoints, and actions of AWS services.

You can use Git branching strategies, such as Gitflow, in combination with isolated CloudFormation stacks to manage the source code that's deployed to different versions of the API. By using this option, you can maintain different versions of your API without the need to duplicate the source code for new versions. With Gitflow, you can add tags to commits within your git repository as releases are performed. As a result, you have a complete snapshot of the source code related to a specific release. As updates need to be performed, you can check out the code from a specific release, make updates, and then deploy the updated source code to the CloudFormation stack that aligns with the corresponding major version. This approach reduces the risk of breaking another API version because each version of the API has isolated source code and is deployed to separate CloudFormation stacks.

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- AWS Certificate Manager (ACM) helps you create, store, and renew public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications.
- AWS Cloud Development Kit (AWS CDK) is an open-source software development framework for defining your cloud infrastructure in code and provisioning it through AWS CloudFormation. This pattern's sample implementation uses the AWS CDK in TypeScript. Working with the AWS CDK in TypeScript uses familiar tools, including the Microsoft TypeScript compiler (`tsc`), Node.js, and the node package manager (`npm`). If you prefer, you can use Yarn although the examples in this

pattern use `npm`. The modules that comprise the AWS Construct Library are distributed through the `npm` repository, npmjs.org.

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Route 53 is a highly available and scalable DNS web service.
- AWS WAF is a web application firewall that helps you monitor HTTP and HTTPS requests that are forwarded to your protected web application resources.

Other tools

- Bruno is an open source, git-friendly API testing client.
- cdk-nag is an open source utility that checks AWS CDK applications for best practices by using rule packs.

Code repository

The code for this pattern is available in the GitHub path-based-versioning-with-api-gateway repository.

Best practices

- Use a robust continuous integration and continuous delivery (CI/CD) pipeline to automate the testing and deployment of your CloudFormation stacks that are built with the AWS CDK. For more information related to this recommendation, see the AWS Well-Architected DevOps Guidance.
- AWS WAF is a managed firewall that easily integrates with services like Amazon API Gateway. Although AWS WAF isn't a necessary component for this versioning pattern to work, we recommend as a security best practice to include AWS WAF with API Gateway.
- Encourage API consumers to upgrade regularly to the latest version of your API so that older versions of your API can be deprecated and removed efficiently.
- Before using this approach in a production setting, implement a firewall and authorization strategy for your API.

- Implement access to the management of AWS resources of your AWS account by using the [least-privilege access model].
- To enforce best practices and security recommendations for applications built with the AWS CDK, we recommend that you use the [cdk-nag utility].

Epics

Prepare your local environment

Task	Description	Skills required
Clone the repository.	To clone the sample applicati on repository, run the following command: `git clone https://github.com/aws-samples/path-based-versioning-with-api-gateway`	App developer
Navigate to the cloned repository.	To navigate to the cloned repository folder location, run the following command: `cd api-gateway-custom-domain-versioning`	App developer
Install the required dependencies.	To install the required dependencies, run the following command: `npm install`	App developer

Deploy the CloudFormation routing stack

Task	Description	Skills required
Initiate deployment of the routing stack.	To initiate the deployment of the CloudFormation routing stack `CustomDomainRouterStack`, run the following command, replacing `example.com` with the name of the domain that you own: `npx cdk deploy CustomDomainRouterStack --parameters PrerequisiteDomainName=example.com` **Note** The stack deployment will not succeed until the following domain DNS validation task is performed successfully.	App developer

Verify domain ownership

Task	Description	Skills required
Verify ownership of your domain.	The certificate will remain in a **Pending validation** status until you prove ownership of the associated domain.	App developer, AWS systems administrator, Network administrator

Task	Description	Skills required
	To prove ownership, add CNAME records to the hosted zone that is associated with the domain. For more information, see [DNS validation](#) in the AWS Certificate Manager documentation. Adding the appropriate records enables the `CustomDomainRouterStack` deployment to succeed.	

Task	Description	Skills required
Create an alias record to point to your API Gateway custom domain.	After the certificate is issued and validated successfully, [create a DNS record](#) that points to your Amazon API Gateway custom domain URL. The custom domain URL is uniquely generated by the provisioning of the custom domain and is specified as a CloudFormation output parameter. Following is an [example of the record](#): **Routing policy**: Simple routing **Record name**: `demo.api-gateway-custom-domain-versioning.example.com` **Alias**: Yes **Record type**: A DNS record of type "A" that points to an AWS resource **Value**: `d-xxxxxxxxxx.execute-api.xx-xxxx-x.amazonaws.com` **TTL (seconds)**: 300	App developer, AWS systems administrator, Network administrator

Deploy CloudFormation stacks and invoke the API

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy the <code>ApiStackV1</code> stack.</td><td>To deploy the <code>ApiStackV1</code> stack, use the following command:
<pre><code>npm run deploy-v1</code></pre>
The following CDK code adds API mapping:
<pre><code>var apiMapping = new
 CfnApiMapping(this,
 "ApiMapping", {
      apiId: this.lamb
daRestApi.restApiId,
      domainName:
 props.customDomain
Name.domainName,
      stage: "api",
      apiMappingKey:
 "api/v1",
    });</code></pre></td><td>App developer</td></tr>
<tr><td>Deploy the <code>ApiStackV2</code> stack.</td><td>To deploy the <code>ApiStackV2</code> stack, use the following command:
<pre><code>npm run deploy-v2</code></pre></td><td>App developer</td></tr>
<tr><td>Invoke the API.</td><td>To invoke the API and test the API endpoints by using Bruno, see the instructions in Additional information.</td><td>App developer</td></tr>
</table>

Clean up resources

Task	Description	Skills required
Clean up resources.	To destroy the resources associated with this sample application, use the following command: `npx cdk destroy --all` **Note** Make sure that you clean up any Route 53 DNS records that were added manually for the domain ownership verification process.	App developer

Troubleshooting

Issue	Solution
The deployment of `CustomDomainRouterStack` times out because the certificate can't be validated.	Make sure that you added the proper DNS validation CNAME records as described in the earlier task. Your new certificate might continue to display a status of **Pending validation** for up to 30 minutes after adding the DNS validation records. For more information, see DNS validation in the AWS Certificate Manager documentation.

Related resources

- Implementing header-based API Gateway versioning with Amazon CloudFront – This AWS Compute Blog post offers a header-based versioning strategy as an alternative to the path-based versioning strategy outlined in this pattern.
- AWS CDK Workshop – This introductory workshop focuses on building and deploying applications on AWS by using the AWS Cloud Development Kit (AWS CDK). This workshop supports Go, Python, and TypeScript.

Additional information

Testing your API with Bruno

We recommend that you use Bruno, an open source API testing tool, to verify that the path-based routing is working properly for the sample application. This pattern provides a sample collection to facilitate testing your sample API.

To invoke and test your API, use the following steps:

1. Install Bruno.
2. Open Bruno.
3. In this pattern's code repository, select **Bruno/Sample-API-Gateway-Custom-Domain-Versioning** and open the collection.
4. To see the **Environments** dropdown in the top right of the user interface (UI), select any request in the collection.
5. In the **Environments** dropdown, select **Configure**.
6. Replace the `REPLACE_ME_WITH_YOUR_DOMAIN` value with your custom domain.
7. Choose **Save**, and then close the **Configuration** section.
8. For **Sandbox Environment**, verify that the **Active** option is selected.
9. Invoke your API by using the **->** button for the selected request.
10. Take note on how validation (passing in non-number values) is handled in V1 compared to V2.

To see screenshots of example API invocation and comparison of V1 and V2 validation, see **Testing your sample API** in the `README.md` file in this pattern's code repository.

Import the psycopg2 library to AWS Lambda to interact with your PostgreSQL database

Louis Hourcade, Amazon Web Services

Summary

[Psycopg](https://) is a PostgresSQL database adapter for Python. Developers use the `psycopg2` library to write Python applications that interact with PostgreSQL databases.

On Amazon Web Services (AWS), developers also use AWS Lambda to run code for applications or backend services. Lambda is a serverless, event-driven compute service runs code without the need to provision or manage servers.

By default, when you create a new function that uses a Python runtime (version 3.9, 3.8, or 3.7), the Lambda runtime environment is created from a base image for Lambda provided by AWS. Libraries, such as `pandas` or `psycopg2`, aren't included in the base image. To use a library, you need to bundle it in a custom package and attach it to Lambda.

There are multiple ways to bundle and attach a library, including the following:

- Deploy your Lambda function from a .zip file archive.
- Deploy your Lambda function from a custom container image.
- Create a Lambda layer, and attach it to your Lambda function.

This pattern demonstrates the first two options.

With a .zip deployment package, adding the `pandas` library to your Lambda function is relatively straightforward. Create a folder on your Linux machine, add the Lambda script together with the `pandas` library and the library's dependencies to the folder, zip the folder, and provide it as a source for your Lambda function.

Although using a .zip deployment package is a common practice, that approach doesn't work for the `psycopg2` library. This pattern first shows the error that you get if you use a .zip deployment package to add the `psycopg2` library to your Lambda function. The pattern then shows how to deploy Lambda from a Dockerfile and edit the Lambda image to make the `psycopg2` library work.

For information about the three resources that the pattern deploys, see the Additional information section.

Prerequisites and limitations

Prerequisites

- An active AWS account with sufficient permissions to deploy the AWS resources used by this pattern
- AWS Cloud Development Kit (AWS CDK) installed globally by running `npm install -g aws-cdk`
- A Git client
- Python
- Docker

Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Product versions

- AWS Lambda runtime version: Python 3.8 (The pattern can be adapted for other Python versions.)
- Psycopg2 version 2.9.3
- Pandas version 1.5.2

Architecture

Solution overview

To illustrate the challenges that you might face when using the `psycopg2` library in Lambda, the pattern deploys two Lambda functions:

- One Lambda function with the Python 3.8 runtime created from a .zip file. The `psycopg2` and `pandas` libraries are installed in this .zip deployment package by using pip.
- One Lambda function with the Python 3.8 runtime created from a Dockerfile. The Dockerfile installs the `psycopg2` and `pandas` libraries into the Lambda container image.

The first Lambda function installs the `pandas` library and its dependencies in a .zip file, and Lambda can use that library.

The second Lambda function demonstrates that by building a container image for your Lambda function, you can run the `pandas` and `psycopg2` libraries in Lambda.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- pandas is a Python-based open source tool for data analysis and manipulation.
- Psycopg is a PostgreSQL database adapter for the Python language that is designed for multithreaded applications. This pattern uses Psycopg 2.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the import-psycopg2-in-lambda-to-interact-with-postgres-database repository on GitHub.

Best practices

This pattern provides you with a working example of using AWS CDK to create a Lambda function from a Dockerfile. If you reuse this code in your application, make sure that the deployed resources meet all security requirements. Use tools such as Checkov, which scans cloud infrastructure configurations to find misconfiguration before the infrastructure is deployed.

Epics

Clone the repository and configure the deployment

Task	Description	Skills required

Clone the repository.	To clone the GitHub repository on your local machine, run the following commands:
	```git clone https://github.com/aws-samples/import-psycopg2-in-lambda-to-interact-with-postgres-database.git```
	```cd AWS-lambda-psycopg2```
		General AWS

Configure your deployment.	Edit the `app.py` file with information about your AWS account:
	```aws_acccount = "AWS_ACCOUNT_ID"```
	```region = "AWS_REGION"```
	```# Select the CPU architecture you are using to build the image (ARM or X86)```
	```architecture = "ARM"```
		General AWS

Bootstrap your AWS account and deploy the application

Task	Description	Skills required
Bootstrap your AWS account.	If you haven't already [bootstrapped your AWS environment](), run the	General AWS

Task	Description	Skills required
	following commands with the AWS credentials of your AWS account: `cdk bootstrap aws://<tooling-account-id>/<aws-region>`	
Deploy the code.	To deploy the AWS CDK application, run the following command: `cdk deploy AWSLambdaPyscopg2`	General AWS

Test the Lambda functions from the AWS Management Console

Task	Description	Skills required
Test the Lambda function created from the .zip file.	To test the Lambda function that was created from the .zip file, do the following: 1. Sign in to the console, and open the Lambda console at https://console.aws.amazon.com/lambda/. 2. Select the `lambda-from-zip` Lambda function. 3. Create a test event to invoke the function. 4. When invoked, the function should raise an	General AWS

Task	Description	Skills required
	error that includes the following message: ```"errorMessage": Unable to import module 'lambda_code': libpq.so.5: cannot open shared object, "stackTrace": [] "errorType": Runtime.ImportModuleError",``` 5. Open the Amazon CloudWatch console at https://console.aws.amazon.com/cloudwatch/. The CloudWatch logs show that the `pandas` library has been imported successfully but that the `psycopg2` library import has failed. Because Lambda doesn't find the required PostgreSQL libraries in the default image, it can't use the `psycopg2` library.	

Task	Description	Skills required

Test the Lambda function created from the Dockerfile.	To use the `psycopg2` library within your Lambda function, you must edit the Lambda Amazon Machine Image (AMI).
	
	To test the Lambda function that was created from the Dockerfile, do the following:
	
	1. Sign in to the console, and open the Lambda console.
	2. Select the `lambda-from-docker` Lambda function.
	3. Create a test event to invoke the function.
	4. When invoked, the function should run successfully.
	
	The following code shows the Dockerfile that the AWS CDK template creates:
	
	`# Start from lambda Python3.8 image`
	`FROM public.ecr.aws/lambda/python:3.8`
	
	`# Copy the lambda code, together with its requirements`
	`COPY lambda/requirements.txt ${LAMBDA_TASK_ROOT}`
		General AWS

Task	Description	Skills required

	``
	COPY lambda/lambda_code
	.py ${LAMBDA_TASK_ROOT
	}
	
	# Install postgresql-
	devel in your image
	RUN yum install -y gcc
	 postgresql-devel
	
	# install the requireme
	nts for the Lambda code
	RUN pip3 install -r
	 requirements.txt
	 --target "${LAMBDA
	_TASK_ROOT}"
	
	# Command can be
	 overwritten by
	 providing a different
	 command in the template
	 directly.
	CMD ["lambda_code.hand
	ler"]
	``
	
	The Dockerfile takes the AWS provided Lambda image for the Python 3.8 runtime and installs postgresql-devel, which contains the libraries needed to compile applicati ons that directly interact with the PostgreSQL managemen t server. The Dockerfile also installs the `pandas` and `psycopg2` libraries, which are indicated in the `requireme nts.txt` file.	

Related resources

- [AWS CDK documentation]
- [AWS Lambda documentation]

Additional information

In this pattern, the AWS CDK template provides an AWS stack with three resources:

- An [AWS Identity and Access Management (IAM) role] for the Lambda functions.
- A Lambda function with a Python 3.8 runtime. The function is deployed from the `Constructs/lambda/lambda_deploy.zip` deployment package.
- A Lambda function with a Python 3.8 runtime. The function is deployed from the Dockerfile under the `Constructs` folder

The script for both Lambda functions checks whether the `pandas` and `psycopg2` libraries are successfully imported:

```
import pandas
print("pandas successfully imported")

import psycopg2
print("psycopg2 successfully imported")

def handler(event, context):
    """Function that checks whether psycopg2  and pandas are successfully imported or
 not"""
    return {"Status": "psycopg2 and pandas successfully imported"}
```

The `lambda_deploy.zip` deployment package is built with the `Constructs/lambda/build.sh` bash script. This script creates a folder, copies the Lambda script, installs the `pandas` and `psycopg2` libraries, and generates the .zip file. To generate the .zip file yourself, run this bash script and redeploy the AWS CDK stack.

The Dockerfile starts with the AWS provided base image for Lambda with a Python 3.8 runtime. The Dockerfile installs the `pandas` and `psycopg2` libraries on top of the default image.

Integrate Amazon API Gateway with Amazon SQS to handle asynchronous REST APIs

Natalia Colantonio Favero and Gustavo Martim, Amazon Web Services

Summary

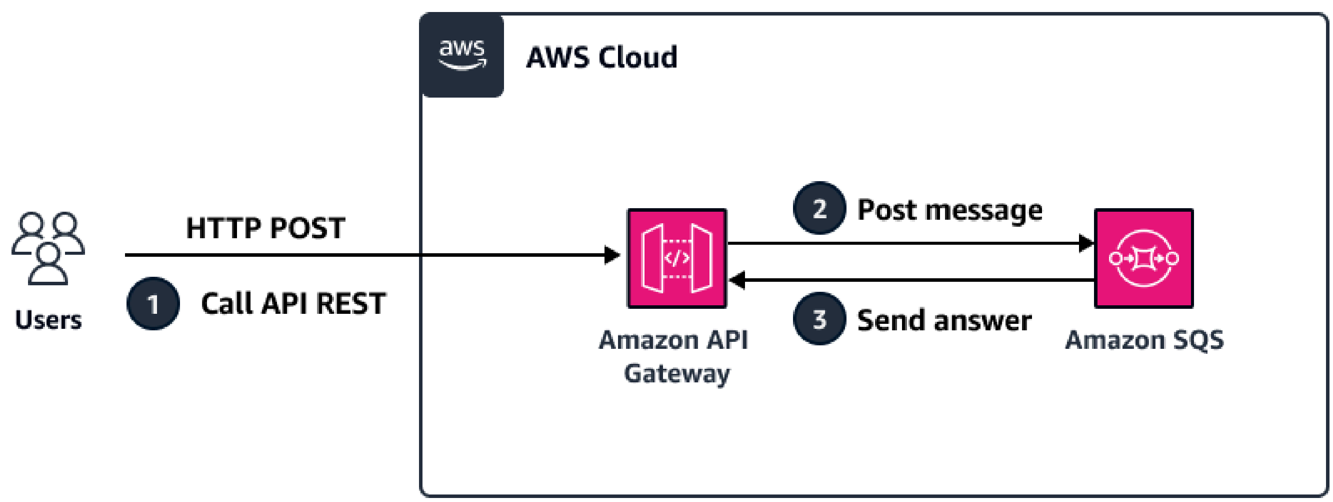
When you deploy REST APIs, sometimes you need to expose a message queue that client applications can publish. For example, you might have problems with the latency of third-party APIs and delays in responses, or you might want to avoid the response time of database queries or avoid scaling the server when there are a large number of concurrent APIs. In these scenarios, the client applications that publish to the queue only need to know that the API received the data—not what happens after the data was received.

This pattern creates a REST API endpoint by using [Amazon API Gateway](https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html) to send a message to [Amazon Simple Queue Service (Amazon SQS)](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html). It creates an easy-to-implement integration between the two services that avoids direct access to the SQS queue.

Prerequisites and limitations

- An active AWS account

Architecture



The diagram illustrates these steps:

1. Request a POST REST API endpoint by using a tool such as Postman, another API, or other technologies.
2. API Gateway posts a message, which is received on the request's body, on the queue.
3. Amazon SQS receives the message and sends an answer to API Gateway with a success or failure code.

Tools

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.

Epics

Create an SQS queue

Task	Description	Skills required
Create a queue.	To create an SQS queue that receives the messages from the REST API: 1. Sign in to your AWS account. 2. Open the Amazon SQS console at https://console.aws.amazon.com/sqs/. 3. Choose **Create queue**. 4. On the **Create queue** page, choose the correct AWS	App developer

Task	Description	Skills required
	Region from the **Region** dropdown list. 5. For **Type**, keep the default setting (**Standard**). 6. Enter a **Name** for your queue. 7. Keep the default values for all other settings. 8. Choose **Create queue**.	

Provide access to Amazon SQS

Task	Description	Skills required
Create an IAM role.	This IAM role gives API Gateway resources full access to Amazon SQS. 1. Open the IAM console at https://console.aws.amazon.com/iam/. 2. In the navigation pane, choose **Roles**, **Create role**. 3. For **Trusted entity type**, choose **AWS service**. 4. For **Use case**, choose **API Gateway** from the dropdown list, and then choose **Next**, **Next**. 5. For **Role name**, enter **AWSGatewayRoleForSQS** and an optional descripti	App developer, AWS administrator

Task	Description	Skills required
	on, and then choose **Create role**. 6. In the **Roles** pane, search for **AWSGatewayRoleForSQS**, and select its checkbox. 7. In the **Permissions policies** section, choose **Add permissions**, **Attach policies**. 8. Search for **AmazonSQSFullAccess** and select it. 9. Choose **Add permissions**. 10. In the **Summary** section of **AWSGatewayRoleForSQS**, copy the Amazon Resource Number (ARN). You will use this ID in a later step.	

Create a REST API

Task	Description	Skills required
Create a REST API.	This is the REST API that HTTP requests are sent to. 1. Open the API Gateway console at https://console.aws.amazon.com/apigateway/. 2. In the **REST API** section, choose **Build.**	App developer

Task	Description	Skills required
	3. For **API name**, enter a name and an optional description for your API, keep all other default settings, and then choose **Create API**.	

Task	Description	Skills required
Connect API Gateway to Amazon SQS.	This step enables the message to flow from inside the HTTP request’s body to Amazon SQS. 1. On the [API Gateway console](), choose the API that you created. 2. On the **Resources** page, in the **Methods** section, choose **Create method**. 3. For **Method type**, choose **POST**. 4. For **Integration type**, choose **AWS service**. 5. For **AWS Region**, choose the Region where you created your SQS queue. 6. For **AWS service**, choose **Simple Queue Service (SQS)**. 7. For HTTP method, choose **POST**. 8. For **Action type**, choose **Use path override**. 9. For **Path override**, enter **<AWS account ID>/<name of SQS queue>**. 10. For **Execution role**, paste the ARN of the role that you created earlier. 11. Choose **Create method**.	App developer

Test the REST API

Task	Description	Skills required

Test the REST API.	Run a test to check for missing configuration:
	
	1. On the [API Gateway console](https://console.aws.amazon.com/apigateway/), choose the REST API that you created.
	2. In the **Resources** pane, choose the **POST** method.
	3. Choose the **Test** tab. (Use the right arrow if the tab isn't displayed.)
	4. For **Request body**, paste the following JSON code:
	`{ "message": "lorem ipsum" }`
	5. Choose **Test**.
	
	You will receive an error that's similar to the following:
	`<UnknownOperationException/>`
		App developer

Change the API integration to forward the request properly to Amazon SQS.	Complete the configuration to fix the integration error: 1. On the [API Gateway console](https://console.aws.amazon.com/apigateway/), choose the API	App developer

Task	Description	Skills required
	you created, and then choose **POST**. 2. The **Method Execution** section shows the visual mapping between API Gateway and Amazon SQS. From this section, choose **Integration request**, and then choose **Edit**. 3. Expand the **HTTP headers** section, and then choose the **Add request header** parameter. • For **Name**, specify **Content-Type**. • For **Mapped from**, enter **'application/x-www-form-urlencoded'**. Make sure to include the single quotation marks. • Select the **Caching** checkbox. 4. Expand the **Mapping templates** section. • Choose **Add mapping template**. • For **Content type**, enter **application/json**. • For **Template body**, paste this code:	

Task	Description	Skills required
	`Action=SendMessage&MessageBody=$input.body` • Choose **Save**.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Test and validate the message in Amazon SQS.</td><td>Run a test to confirm that the test completed successfully:

1. On the API Gateway console, choose the REST API you created.
2. In the Resources pane, choose the POST method.
3. Choose the Test tab. (Use the right arrow if the tab isn't displayed.)
4. For Request body, paste the following JSON code:
<pre>{
     "message":
 "lorem ipsum"
}</pre>5. Choose Test.
6. Open the Amazon SQS console.
7. In the navigation pane, choose Queues, and then choose your queue.
8. Choose Send and receive messages.
9. Choose Poll for messages.
10. Choose Message. It should display the following:
<pre>Body { "message":
 "lorem ipsum" }</pre></td><td>App developer</td></tr>
</table>

Task	Description	Skills required

Test API Gateway with a special character.	Run a test that includes special characters (such as &) that aren't acceptable in a message:
	
	1. On the [API Gateway console](#), choose your API.
	2. Repeat the test from the earlier step by using the following JSON code:
	`{ "message": "lorem ipsum &" }`
	3. Choose **Test**.
	
	You will receive an error such as the following:
	`{ "Error": { "Code": "AccessDenied", "Message": "Access to the resource https://sqs.us-east-2.amazonaws.com/976166761794/Apg2 is denied.", "Type": "Sender" }, "RequestId": "e83c9c67-bcf6-5e9a-91e9-c737094b17ab"`
		App developer

Task	Description	Skills required
	```}``` This is because special characters aren't supported by default in the message body. In the next step, you'll configure API Gateway to support special characters. For more information about content type conversions, see the [API Gateway documentation]().	

Task	Description	Skills required

Change the API configuration to support special characters.	Adjust the configuration to accept special characters in the message:
	
	1. On the [API Gateway console](https://console.aws.amazon.com/apigateway/), choose the API you created, and then choose **POST**.
	2. Choose **Integration request**, and then choose **Edit**.
	3. Change **Content Handling** to **Convert to text.**
	4. In the **Mapping templates** section:
	• For **Content type**, enter **application/json**.
	• For **Template body**, specify:
	`Action=SendMessage&MessageBody=$util.urlEncode($input.body)`
	• Choose **Save**.
	5. Choose the **Test** tab.
	6. For **Request body**, enter the JSON code from earlier:
	`{ " message": "lorem ipsum &" }`
		App developer

Task	Description	Skills required
	7. Choose **Test**. 8. Open the [Amazon SQS console](https://console.aws.amazon.com/sqs/). 9. Select your queue, and then choose **Send and receive messages**, **Poll for messages**, **Message** as earlier. The new message should include the special character.	

Deploy the REST API

Task	Description	Skills required
Deploy the API.	To deploy the REST API: 1. Open the [API Gateway console](https://console.aws.amazon.com/apigateway/). 2. Choose your API. 3. Choose **Deploy API**. For more information about this step, see the [API Gateway documentation](https://docs.aws.amazon.com/apigateway/).	App developer
Test with an external tool.	Run a test with an external tool to confirm that the message is received successfu lly:	App developer

Task	Description	Skills required
	1. Open a tool such as Postman, Insomnia, or cURL. 2. Run your API. 3. Open the Amazon SQS console. 4. Select your queue. 5. Load messages to see the new message.	

Clean Up

Task	Description	Skills required
Delete the API.	On the API Gateway console, choose the API you created, and then choose **Delete**.	App developer
Delete the IAM role.	On the IAM console, in the **Roles** pane, select **AWSGatewayRoleForSQS**, and then choose **Delete**.	App developer
Delete the SQS queue.	On the Amazon SQS console, in the **Queues** pane, choose the SQS queue you created, and then choose **Delete**.	App developer

Related resources

- SQS-SendMessage (API Gateway documentation)
- Content type conversions in API Gateway (API Gateway documentation)
- $util variables (API Gateway documentation)

- How do I integrate an API Gateway REST API with Amazon SQS and resolve common errors? (AWS re:Post article)

Process events asynchronously with Amazon API Gateway and AWS Lambda

Andrea Meroni, Mariem Kthiri, Nadim Majed, and Michael Wallner, Amazon Web Services

Summary

Amazon API Gateway is a fully managed service that developers can use to create, publish, maintain, monitor, and secure APIs at any scale. It handles the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.

An important service quota of API Gateway is the integration timeout. The timeout is the maximum time in which a backend service must return a response before the REST API returns an error. The hard limit of 29 seconds is generally acceptable for synchronous workloads. However, that limit represents a challenge for those developers who want to use API Gateway with asynchronous workloads.

This pattern shows an example architecture to process events asynchronously using API Gateway and AWS Lambda. The architecture supports running processing jobs of duration up to 15 minutes, and it uses a basic REST API as the interface.

Projen is used to set up the local development environment and to deploy the example architecture to a target AWS account, in combination with the AWS Cloud Development Kit (AWS CDK) Toolkit, Docker, and Node.js. Projen automatically sets up a Python virtual environment with pre-commit and the tools that are used for code quality assurance, security scanning, and unit testing. For more information, see the Tools section.

Prerequisites and limitations

Prerequisites

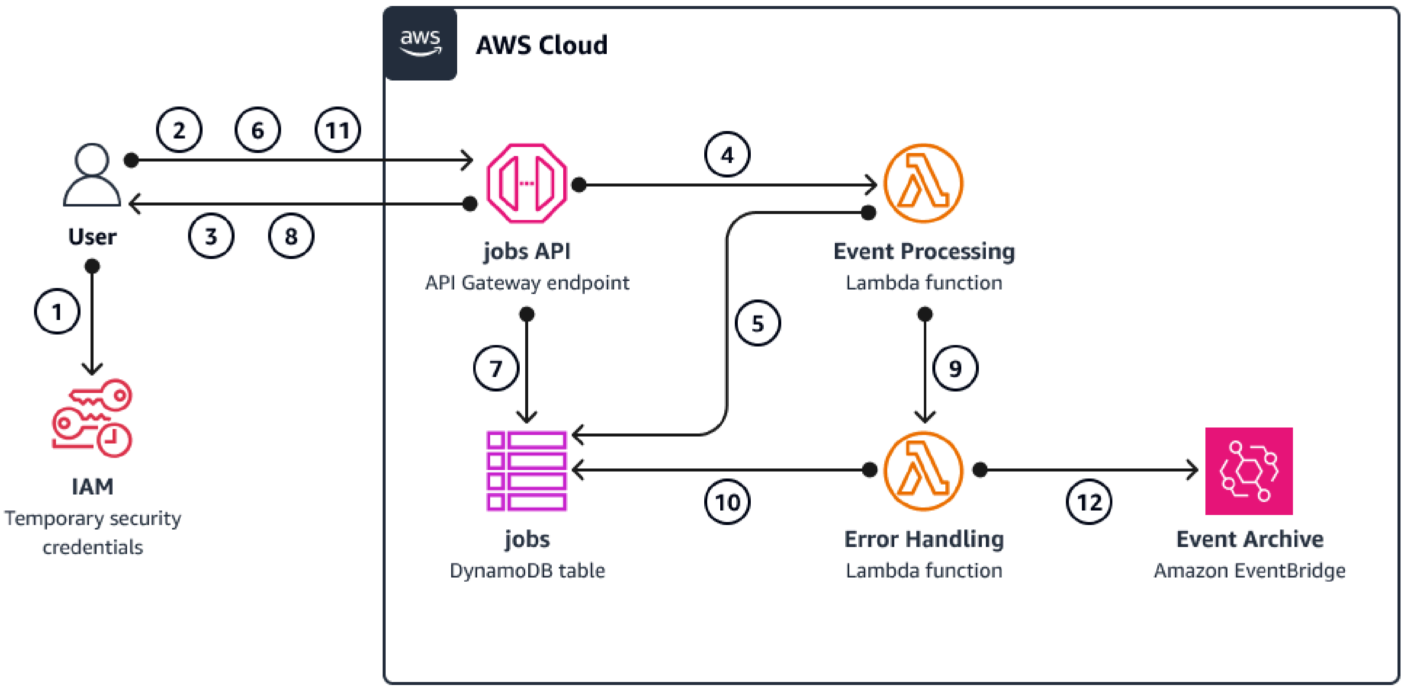
- An active AWS account
- The following tools installed on your workstation:
  - AWS Cloud Development Kit (AWS CDK) Toolkit version 2.85.0
  - Docker version 20.10.21
  - Node.js version 18.13.0
  - Projen version 0.71.111
  - Python version 3.9.16

Limitations

- The maximum runtime of a job is limited by the maximum runtime for Lambda functions (15 minutes).
- The maximum number of concurrent job requests is limited by the reserved concurrency of the Lambda function.

Architecture

The following diagram shows the interaction of the jobs API with the event-processing and error-handling Lambda functions, with events stored in an Amazon EventBridge event archive.



A typical workflow includes the following steps:

1. You authenticate against AWS Identity and Access Management (IAM) and obtain security credentials.
2. You send an HTTP `POST` request to the `/jobs` jobs API endpoint, specifying the job parameters in the request body.
3. The jobs API, which is an API Gateway REST API, returns to you an HTTP response that contains the job identifier.
4. The jobs API invokes asynchronously the event-processing Lambda function.

5. The event-processing function processes the event, and then it puts the job results in the jobs Amazon DynamoDB table
6. You send an HTTP GET request to the `/jobs/{jobId}` jobs API endpoint, with the job identifier from step 3 as `{jobId}`.
7. The jobs API queries the `jobs` DynamoDB table to retrieve the job results.
8. The jobs API returns an HTTP response that contains the job results.
9. If the event processing fails, the event-processing function sends the event to the error-handling function.
10. The error-handling function puts the job parameters in the `jobs` DynamoDB table.
11. You can retrieve the job parameters by sending an HTTP GET request to the `/jobs/{jobId}` jobs API endpoint.
12. If the error handling fails, the error-handling function sends the event to an EventBridge event archive.

    You can replay the archived events by using EventBridge.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Other tools

- autopep8 automatically formats Python code based on the Python Enhancement Proposal (PEP) 8 style guide.
- Bandit scans Python code to find common security issues.
- Commitizen is a Git commit checker and `CHANGELOG` generator.
- cfn-lint is an AWS CloudFormation linter
- Checkov is a static code-analysis tool that checks infrastructure as code (IaC) for security and compliance misconfigurations.
- jq is a command-line tool for parsing JSON.
- Postman is an API platform.
- pre-commit is a Git hooks manager.
- Projen is a project generator.
- pytest is a Python framework for writing small, readable tests.

Code repository

This example architecture code can be found in the GitHub Asynchronous Event Processing with API Gateway and Lambda repository.

Best practices

- This example architecture doesn't include monitoring of the deployed infrastructure. If your use case requires monitoring, evaluate adding CDK Monitoring Constructs or another monitoring solution.
- This example architecture uses IAM permissions to control the access to the jobs API. Anyone authorized to assume the `JobsAPIInvokeRole` will be able to invoke the jobs API. As such, the access control mechanism is binary. If your use case requires a more complex authorization model, evaluate using a different access control mechanism.
- When a user sends an HTTP `POST` request to the `/jobs` jobs API endpoint, the input data is validated at two different levels:
  - Amazon API Gateway is in charge of the first request validation.
  - The event processing function performs the second request.

  No validation is performed when the user does an HTTP `GET` request to the `/jobs/{jobId}` jobs API endpoint. If your use case requires additional input validation and an increased level of security, evaluate using AWS WAF to protect your API.

Epics

Set up the environment

Task	Description	Skills required
Clone the repository.	To clone the repository locally, run the following command: `git clone https://github.com/aws-samples/asynchronous-event-processing-api-gateway-lambda-cdk.git`	DevOps engineer

Set up the project.	Change the directory to the repository root and set up the Python virtual environment and all the tools by using [Projen](https://github.com/projen/projen):
	`cd asynchronous-event-processing-api-gateway-api-gateway-lambda-cdk`
	`npx projen`
		DevOps engineer

Install pre-commit hooks.	To install pre-commit hooks, do the following: 1. Activate the [Python virtual environment](https://docs.python.org/3/library/venv.html): `source .env/bin/activate` 2. Install the [pre-commit](https://pre-commit.com/) hooks:	DevOps engineer

Task	Description	Skills required
	`pre-commit install` `pre-commit install --hook-type commit-msg`	

Deploy the example architecture

Task	Description	Skills required
Bootstrap AWS CDK.	To bootstrap AWS CDK in your AWS account, run the following command: `AWS_PROFILE=$YOUR_AWS_PROFILE npx projen bootstrap`	AWS DevOps
Deploy the example architecture.	To deploy the example architecture in your AWS account, run the following command: `AWS_PROFILE=$YOUR_AWS_PROFILE npx projen deploy`	AWS DevOps

Test the architecture

Task	Description	Skills required
Install test prerequisites.	Install on your workstation the [AWS Command Line Interface (AWS CLI)](), [Postman](), and [jq]().	DevOps engineer

Task	Description	Skills required
	Using Postman to test this example architecture is suggested but not mandatory . If you choose an alternati ve API testing tool, make sure that it supports AWS Signature Version 4 authentic ation, and refer to the exposed API endpoints that can be inspected by exporting the REST API.	

Task	Description	Skills required

Assume the `JobsAPIInvokeRole` .	[Assume](https://docs.aws.amazon.com/cli/latest/reference/sts/assume-role.html) the `JobsAPIInvokeRole` that was printed as output from the deploy command:
	`CREDENTIALS=$(AWS_PROFILE=$<YOUR_AWS_PROFILE> aws sts assume-role \`
	`--no-cli-pager \`
	`--role-arn $<JOBS_API_INVOKE_ROLE_ARN> \`
	`--role-session-name JobsAPIInvoke)`
	`export AWS_ACCESS_KEY_ID=$(cat $CREDENTIALS | jq '.Credentials''.AccessKeyId')`
	`export AWS_SECRET_ACCESS_KEY=$(cat $CREDENTIALS | jq '.Credentials''.SecretAccessKey')`
	`export AWS_SESSION_TOKEN==$(cat $CREDENTIALS | jq '.Credentials''.SessionToken')`
		AWS DevOps

Task	Description	Skills required
Configure Postman.	1. To import the Postman collection that's included in the repository, follow the instructions in the Postman documentation. 2. Set the `JobsAPI` variables with the following values: • `accessKey` – The value of the `Credentials.AccessKeyId` attribute from the `assume-role` command • `baseUrl` – The value of the `JobsApiJobsAPIEndpoint` output from the deploy command, without the trailing slash • `region` – The value of the AWS Region where you deployed the example architecture • `seconds` – The value of the input parameter for the example job. It must be a positive integer • `secretKey` – The value of the `Credentials.SecretAccessKey` attribute from the `assume-role` command	AWS DevOps

Task	Description	Skills required
	• `sessionToken` – The value of the `Credentials.SessionToken` attribute from the `assume-role` command	
Test the example architecture.	To test the example architecture, send requests to the jobs API. For more information, see the Postman documentation.	DevOps engineer

Troubleshooting

Issue	Solution
Destruction and subsequent redeployment of the example architecture fails because the Amazon CloudWatch Logs log group `/aws/apigateway/JobsAPIAccessLogs` already exists.	1. If necessary, export your log data to Amazon S3. 2. Delete the CloudWatch Logs log group `/aws/apigateway/JobsAPIAccessLogs` . 3. Redeploy the example architecture.

Related resources

- API Gateway mapping template and access logging variable reference
- Set up asynchronous invocation of the backend Lambda function

Process events asynchronously with Amazon API Gateway and Amazon DynamoDB Streams

Andrea Meroni, Mariem Kthiri, Nadim Majed, Alessandro Trisolini, and Michael Wallner, Amazon Web Services

Summary

Amazon API Gateway is a fully managed service that developers can use to create, publish, maintain, monitor, and secure APIs at any scale. It handles the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.

An important service quota of API Gateway is the integration timeout. The timeout is the maximum time in which a backend service must return a response before the REST API returns an error. The hard limit of 29 seconds is generally acceptable for synchronous workloads. However, that limit represents a challenge for those developers who want to use API Gateway with asynchronous workloads.

This pattern shows an example architecture for processing events asynchronously using API Gateway, Amazon DynamoDB Streams, and AWS Lambda. The architecture supports running parallel processing jobs with the same input parameters, and it uses a basic REST API as the interface. In this example, using Lambda as the backend limits the duration of jobs to 15 minutes. You can avoid this limit by using an alternative service to process incoming events (for example, AWS Fargate).

Projen is used to set up the local development environment and to deploy the example architecture to a target AWS account, in combination with the AWS Cloud Development Kit (AWS CDK) Toolkit, Docker and Node.js. Projen automatically sets up a Python virtual environment with pre-commit and the tools that are used for code quality assurance, security scanning, and unit testing. For more information, see the Tools section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The following tools installed on your workstation:
  - AWS Cloud Development Kit (AWS CDK) Toolkit version 2.85.0 or later

- [Docker](https://www.docker.com/) version 20.10.21 or later
- [Node.js](https://nodejs.org/en/download) version 18 or later
- [Projen](https://github.com/projen/projen) version 0.71.111 or later
- [Python](https://www.python.org/downloads/) version 3.9.16 or later

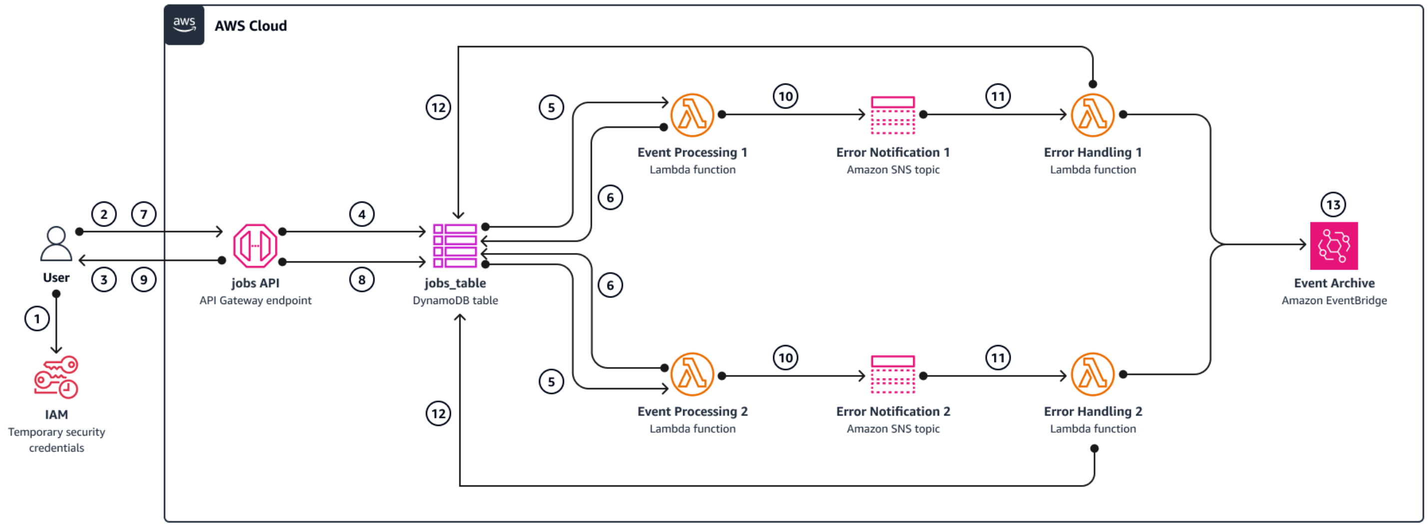
Limitations

- The advised maximum number of readers for DynamoDB Streams is two to avoid throttling.
- The maximum runtime of a job is limited by the maximum runtime for Lambda functions (15 minutes).
- The maximum number of concurrent job requests is limited by the reserved concurrency of the Lambda functions.

Architecture

Architecture

The following diagram shows the interaction of the jobs API with DynamoDB Streams and the event-processing and error-handling Lambda functions, with events stored in an Amazon EventBridge event archive.



A typical workflow includes the following steps:

1. You authenticate against AWS Identity and Access Management (IAM) and obtain security credentials.

2. You send an HTTP POST request to the /jobs jobs API endpoint, specifying the job parameters in the request body.
3. The jobs API returns to you an HTTP response that contains the job identifier.
4. The jobs API puts the job parameters in the jobs_table Amazon DynamoDB table.
5. The jobs_table DynamoDB table DynamoDB stream invokes the event-processing Lambda functions.
6. The event-processing Lambda functions process the event and then put the job results in the jobs_table DynamoDB table. To help ensure consistent results, the event-processing functions implement an optimistic locking mechanism.
7. You send an HTTP GET request to the /jobs/{jobId} jobs API endpoint, with the job identifier from step 3 as {jobId}.
8. The jobs API queries the jobs_table DynamoDB table to retrieve the job results.
9. The jobs API returns an HTTP response that contains the job results.
10. If the event processing fails, the event-processing function's source mapping sends the event to the error-handling Amazon Simple Notification Service (Amazon SNS) topic.
11. The error-handling SNS topic asynchronously pushes the event to the error-handling function.
12. The error-handling function puts the job parameters in the jobs_table DynamoDB table.

    You can retrieve the job parameters by sending an HTTP GET request to the /jobs/{jobId} jobs API endpoint.
13. If the error handling fails, the error-handling function sends the event to an Amazon EventBridge archive.

    You can replay the archived events by using EventBridge.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.

- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

Other tools

- autopep8 automatically formats Python code based on the Python Enhancement Proposal (PEP) 8 style guide.
- Bandit scans Python code to find common security issues.
- Commitizen is a Git commit checker and `CHANGELOG` generator.
- cfn-lint is an AWS CloudFormation linter
- Checkov is a static code-analysis tool that checks infrastructure as code (IaC) for security and compliance misconfigurations.
- jq is a command-line tool for parsing JSON.
- Postman is an API platform.
- pre-commit is a Git hooks manager.
- Projen is a project generator.
- pytest is a Python framework for writing small, readable tests.

Code repository

This example architecture code can be found in the GitHub Asynchronous Processing with API Gateway and DynamoDB Streams repository.

Best practices

- This example architecture doesn't include monitoring of the deployed infrastructure. If your use case requires monitoring, evaluate adding CDK Monitoring Constructs or another monitoring solution.

- This example architecture uses IAM permissions to control the access to the jobs API. Anyone authorized to assume the `JobsAPIInvokeRole` will be able to invoke the jobs API. As such, the access control mechanism is binary. If your use case requires a more complex authorization model, evaluate using a different access control mechanism.
- When a user sends an HTTP `POST` request to the `/jobs` jobs API endpoint, the input data is validated at two different levels:
  - API Gateway is in charge of the first request validation.
  - The event processing function performs the second request.

  No validation is performed when the user does an HTTP `GET` request to the `/jobs/{jobId}` jobs API endpoint. If your use case requires additional input validation and an increased level of security, evaluate using AWS WAF to protect your API.
- To avoid throttling, the DynamoDB Streams documentation discourages users from reading with more than two consumers from the same stream's shard. To scale out the number of consumers, we recommend using Amazon Kinesis Data Streams.
- Optimistic locking has been used in this example to ensure consistent updates of items in the `jobs_table` DynamoDB table. Depending on the use-case requirement, you might need to implement more reliable locking mechanisms, such as pessimistic locking.

Epics

Set up the environment

Task	Description	Skills required
Clone the repository.	To clone the repository locally, run the following command: `git clone https://github.com/aws-samples/asynchronous-event-processing-api-gateway-dynamodb-streams-cdk.git`	DevOps engineer

Task	Description	Skills required

Set up the project.	Change the directory to the repository root, and set up the Python virtual environment and all the tools by using [Projen]():
	`cd asynchronous-event-processing-api-gateway-api-gateway-dynamodb-streams-cdk`
	`npx projen`
		DevOps engineer

Install pre-commit hooks.	To install pre-commit hooks, do the following:
	1. Activate the [Python virtual environment]():
	`source .env/bin/activate`
	2. Install the [pre-commit]() hooks:
	`pre-commit install`
	`pre-commit install --hook-type commit-msg`
		DevOps engineer

Deploy the example architecture

Task	Description	Skills required
Bootstrap AWS CDK.	To bootstrap [AWS CDK]() in your AWS account, run the following command:	AWS DevOps

Task	Description	Skills required
	```AWS_PROFILE=$YOUR_AWS_PROFILE npx projen bootstrap```	
Deploy the example architecture.	To deploy the example architecture in your AWS account, run the following command: ```AWS_PROFILE=$YOUR_AWS_PROFILE npx projen deploy```	AWS DevOps

Test the architecture

Task	Description	Skills required
Install test prerequisites.	Install on your workstation the AWS Command Line Interface (AWS CLI), Postman, and jq. Using Postman to test this example architecture is suggested but not mandatory. If you choose an alternative API testing tool, make sure that it supports AWS Signature Version 4 authentication, and refer to the exposed API endpoints that can be inspected by exporting the REST API.	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Assume the `JobsAPIInvokeRole` .</td><td>

Assume the `JobsAPIInvokeRole` that was printed as output from the `deploy` command:

```
CREDENTIALS=$(AWS_PROFILE=$<YOUR_AWS_PROFILE> aws sts assume-role \
--no-cli-pager \
--role-arn $<JOBS_API_INVOKE_ROLE_ARN> \
--role-session-name JobsAPIInvoke)
export AWS_ACCESS_KEY_ID=$(cat $CREDENTIALS | jq ‘.Credentials’’.AccessKeyId’)
export AWS_SECRET_ACCESS_KEY=$(cat $CREDENTIALS | jq ‘.Credentials’’.SecretAccessKey’)
export AWS_SESSION_TOKEN==$(cat $CREDENTIALS | jq ‘.Credentials’’.SessionToken’)
```

</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required

Configure Postman.	• To import the Postman collection that's included in the repository, follow the instructions in the [Postman documentation]().
	• Set the `JobsAPI` [variables]() with the following values:
	  • `accessKey` – The value of the `Credentials.AccessKeyId` attribute from the `assume-role` command.
	  • `baseUrl` – The value of the `JobsApiJobsAPIEndpoint` output from the `deploy` command, without the trailing slash.
	  • `region` – The value of the AWS Region where you deployed the example architecture.
	  • `seconds` – The value of the input parameter for the example job. It must be a positive integer.
	  • `secretKey` – The value of the `Credentials.SecretAccessKey` attribute from the `assume-role` command.
		AWS DevOps

Task	Description	Skills required
	• `sessionToken` – The value of the `Credentials.SessionToken` attribute from the `assume-role` command.	
Test the example architecture.	To test the example architecture, send requests to the jobs API. For more information, see the Postman documentation.	DevOps engineer

Troubleshooting

Issue	Solution
Destruction and subsequent redeployment of the example architecture fails because the Amazon CloudWatch Logs log group `/aws/apigateway/JobsAPIAccessLogs` already exists.	1. If necessary, export your log data to Amazon Simple Storage Service (Amazon S3). 2. Delete the CloudWatch Logs log group `/aws/apigateway/JobsAPIAccessLogs` . 3. Redeploy the example architecture.

Related resources

- API Gateway mapping template and access logging variable reference
- Change data capture for DynamoDB Streams
- Optimistic locking with version number
- Using Kinesis Data Streams to capture changes to DynamoDB

Process events asynchronously with Amazon API Gateway, Amazon SQS, and AWS Fargate

Andrea Meroni, Mariem Kthiri, Nadim Majed, Alessandro Trisolini, and Michael Wallner, Amazon Web Services

Summary

Amazon API Gateway is a fully managed service that developers can use to create, publish, maintain, monitor, and secure APIs at any scale. It handles the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.

An important service quota of API Gateway is the integration timeout. The timeout is the maximum time in which a backend service must return a response before the REST API returns an error. The hard limit of 29 seconds is generally acceptable for synchronous workloads. However, that limit represents a challenge for those developers who want to use API Gateway with asynchronous workloads.

This pattern shows an example architecture to process events asynchronously using API Gateway, Amazon Simple Queue Service (Amazon SQS) and AWS Fargate. The architecture supports running processing jobs without duration restrictions, and it uses a basic REST API as the interface.

Projen is used to set up the local development environment and to deploy the example architecture to a target AWS account, in combination with the AWS Cloud Development Kit (AWS CDK), Docker, and Node.js. Projen automatically sets up a Python virtual environment with pre-commit and the tools that are used for code quality assurance, security scanning, and unit testing. For more information, see the Tools section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The following tools installed on your workstation:
  - AWS Cloud Development Kit (AWS CDK) Toolkit version 2.85.0 or later
  - Docker version 20.10.21 or later
  - Node.js version 18 or later

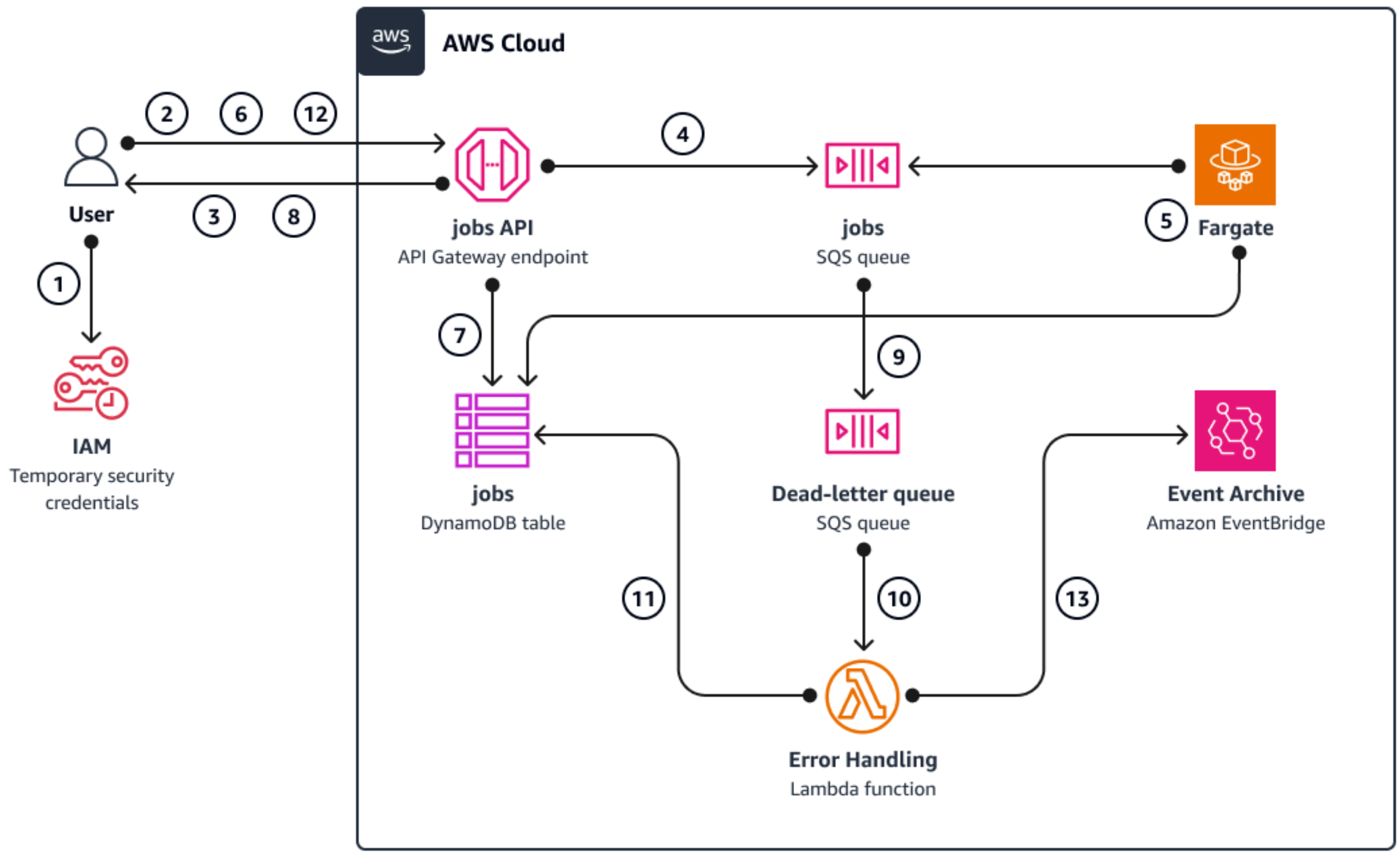
- [Projen](#) version 0.71.111 or later
- [Python](#) version 3.9.16 or later

Limitations

- Concurrent jobs are limited to 500 tasks per minute, which is the maximum number of tasks that Fargate can provision.

Architecture

The following diagram shows the interaction of the jobs API with the `jobs` Amazon DynamoDB table, the event-processing Fargate service, and the error-handling AWS Lambda function. Events are stored in an Amazon EventBridge event archive.



A typical workflow includes the following steps:

1. You authenticate against AWS Identity and Access Management (IAM) and obtain security credentials.

2. You send an HTTP `POST` request to the `/jobs` jobs API endpoint, specifying the job parameters in the request body.
3. The jobs API, which is an API Gateway REST API, returns to you an HTTP response that contains the job identifier.
4. The jobs API sends a message to the SQS queue.
5. Fargate pulls the message from the SQS queue, processes the event, and then puts the job results in the `jobs` DynamoDB table.
6. You send an HTTP `GET` request to the `/jobs/{jobId}` jobs API endpoint, with the job identifier from step 3 as `{jobId}`.
7. The jobs API queries the `jobs` DynamoDB table to retrieve the job results.
8. The jobs API returns an HTTP response that contains the job results.
9. If the event processing fails, the SQS queue sends the event to the dead-letter queue (DLQ).
10. An EventBridge event initiates the error-handling function.
11. The error-handling function puts the job parameters in the `jobs` DynamoDB table.
12. You can retrieve the job parameters by sending an HTTP `GET` request to the `/jobs/{jobId}` jobs API endpoint.
13. If the error handling fails, the error-handling function sends the event to an EventBridge archive.

    You can replay the archived events by using EventBridge.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Fargate helps you run containers without needing to manage servers or Amazon Elastic Compute Cloud (Amazon EC2) instances. It's used in conjunction with Amazon Elastic Container Service (Amazon ECS).
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.

- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.

Other tools

- autopep8 automatically formats Python code based on the Python Enhancement Proposal (PEP) 8 style guide.
- Bandit scans Python code to find common security issues.
- Commitizen is a Git commit checker and `CHANGELOG` generator.
- cfn-lint is an AWS CloudFormation linter
- Checkov is a static code-analysis tool that checks infrastructure as code (IaC) for security and compliance misconfigurations.
- jq is a command-line tool for parsing JSON.
- Postman is an API platform.
- pre-commit is a Git hooks manager.
- Projen is a project generator.
- pytest is a Python framework for writing small, readable tests.

Code repository

This example architecture code can be found in the GitHub Asynchronous Processing with API Gateway and SQS repository.

Best practices

- This example architecture doesn't include monitoring of the deployed infrastructure. If your use case requires monitoring, evaluate adding CDK Monitoring Constructs or another monitoring solution.
- This example architecture uses IAM permissions to control the access to the jobs API. Anyone authorized to assume the `JobsAPIInvokeRole` will be able to invoke the jobs API. As such, the access control mechanism is binary. If your use case requires a more complex authorization model, evaluate using a different access control mechanism.

- When a user sends an HTTP POST request to the /jobs jobs API endpoint, the input data is validated at two different levels:
  - API Gateway is in charge of the first request validation.
  - The event processing function performs the second request.

  No validation is performed when the user does an HTTP GET request to the /jobs/{jobId} jobs API endpoint. If your use case requires additional input validation and an increased level of security, evaluate using AWS WAF to protect your API.

Epics

Set up the environment

Task	Description	Skills required
Clone the repository.	To clone the repository locally, run the following command: `git clone https://github.com/aws-samples/asynchronous-event-processing-api-gateway-sqs-cdk.git`	DevOps engineer

Set up the project.	Change the directory to the repository root, and set up the Python virtual environment and all the tools by using Projen:
	`cd asynchronous-event-processing-api-gateway-api-gateway-sqs-cdk`
	`npx projen`
		DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install pre-commit hooks.</td><td>To install pre-commit hooks, do the following:
1. Activate the Python virtual environment:
<pre>source .env/bin/
activate</pre>2. Install the pre-commit hooks:
<pre>pre-commit install
pre-commit install --
hook-type commit-msg</pre></td><td>DevOps engineer</td></tr>
</table>

Deploy the example architecture

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Bootstrap AWS CDK.</td><td>To bootstrap AWS CDK in your AWS account, run the following command:
<pre>AWS_PROFILE=$YOUR_
AWS_PROFILE npx projen
 bootstrap</pre></td><td>AWS DevOps</td></tr>
<tr><td>Deploy the example architecture.</td><td>To deploy the example architecture in your AWS account, run the following command:</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	`AWS_PROFILE=$YOUR_AWS_PROFILE npx projen deploy`	

Test the architecture

Task	Description	Skills required
Install test prerequisites.	Install on your workstation the AWS Command Line Interface (AWS CLI), Postman, and jq. Using Postman to test this example architecture is suggested but not mandatory. If you choose an alternative API testing tool, make sure that it supports AWS Signature Version 4 authentication, and refer to the exposed API endpoints that can be inspected by exporting the REST API.	DevOps engineer

Assume the `JobsAPIInvokeRole` .	Assume the `JobsAPIInvokeRole` that was printed as output from the `deploy` command:
	
	`CREDENTIALS=$(AWS_PROFILE=$<YOUR_AWS_PROFILE> aws sts assume-role \`
	`--no-cli-pager \`
		AWS DevOps

Task	Description	Skills required

	`--role-arn $<JOBS_API_INVOKE_ROLE_ARN> \`
	`--role-session-name JobsAPIInvoke)`
	`export AWS_ACCESS_KEY_ID=$(cat $CREDENTIALS | jq '.Credentials''.AccessKeyId')`
	`export AWS_SECRET_ACCESS_KEY=$(cat $CREDENTIALS | jq '.Credentials''.SecretAccessKey')`
	`export AWS_SESSION_TOKEN==$(cat $CREDENTIALS | jq '.Credentials''.SessionToken')`	

Task	Description	Skills required

Configure Postman.	• To import the Postman collection that's included in the repository, follow the instructions in the [Postman documentation]().
	• Set the `JobsAPI` [variables]() with the following values:
	  ◦ `accessKey` – The value of the `Credentials.AccessKeyId` attribute from the `assume-role` command.
	  ◦ `baseUrl` – The value of the `JobsApiJobsAPIEndpoint` output from the `deploy` command, without the trailing slash.
	  ◦ `region` – The value of the AWS Region where you deployed the example architecture.
	  ◦ `seconds` – The value of the input parameter for the example job. It must be a positive integer.
	  ◦ `secretKey` – The value of the `Credentials.SecretAccessKey` attribute from the `assume-role` command.
		AWS DevOps

Task	Description	Skills required
	• `sessionToken` – The value of the `Credentials.SessionToken` attribute from the `assume-role` command.	
Test the example architecture.	To test the example architecture, send requests to the jobs API. For more information, see the [Postman documentation]().	DevOps engineer

Troubleshooting

Issue	Solution
Destruction and subsequent redeployment of the example architecture fails because the [Amazon CloudWatch Logs log group]() `/aws/apigateway/JobsAPIAccessLogs` already exists.	1. If necessary, [export your log data to Amazon Simple Storage Service (Amazon S3)](). 2. Delete the CloudWatch Logs log group `/aws/apigateway/JobsAPIAccessLogs` . 3. Redeploy the example architecture.
Destruction and subsequent redeployment of the example architecture fails because the [CloudWatch Logs log group]() `/aws/ecs/EventProcessingServiceLogs` already exists.	1. If necessary, [export your log data to Amazon S3](). 2. Delete the CloudWatch Logs log group `/aws/ecs/EventProcessingServiceLogs`. 3. Redeploy the example architecture.

Related resources

- [API Gateway mapping template and access logging variable reference]
- [How do I integrate an API Gateway REST API with Amazon SQS and resolve common errors?]

Run AWS Systems Manager Automation tasks synchronously from AWS Step Functions

Elie El khoury, Amazon Web Services

Summary

This pattern explains how to integrate AWS Step Functions with AWS Systems Manager. It uses AWS SDK service integrations to call the Systems Manager **startAutomationExecution** API with a task token from a state machine workflow, and pauses until the token returns with a success or failure call. To demonstrate the integration, this pattern implements an Automation document (runbook) wrapper around the `AWS-RunShellScript` or `AWS-RunPowerShellScript` document, and uses `.waitForTaskToken` to synchronously call `AWS-RunShellScript` or `AWS-RunPowerShellScript`. For more information about AWS SDK service integrations in Step Functions, see the AWS Step Functions Developer Guide.

Step Functions is a low-code, visual workflow service that you can use to build distributed applications, automate IT and business processes, and build data and machine learning pipelines by using AWS services. Workflows manage failures, retries, parallelization, service integrations, and observability so you can focus on higher-value business logic.

Automation, a capability of AWS Systems Manager, simplifies common maintenance, deployment, and remediation tasks for AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Relational Database Service (Amazon RDS), Amazon Redshift, and Amazon Simple Storage Service (Amazon S3). Automation gives you granular control over the concurrency of your automations. For example, you can specify how many resources to target concurrently, and how many errors can occur before an automation is stopped.

For implementation details, including runbook steps, parameters, and examples, see the Additional information section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Identity and Access Management (IAM) permissions to access Step Functions and Systems Manager

- An EC2 instance with Systems Manager Agent (SSM Agent) installed on the instance
- An IAM instance profile for Systems Manager attached to the instance where you plan to run the runbook
- A Step Functions role that has the following IAM permissions (which follow the principle of least privilege):

```
{
             "Effect": "Allow",
             "Action": "ssm:StartAutomationExecution",
             "Resource": "*"
 }
```

Product versions

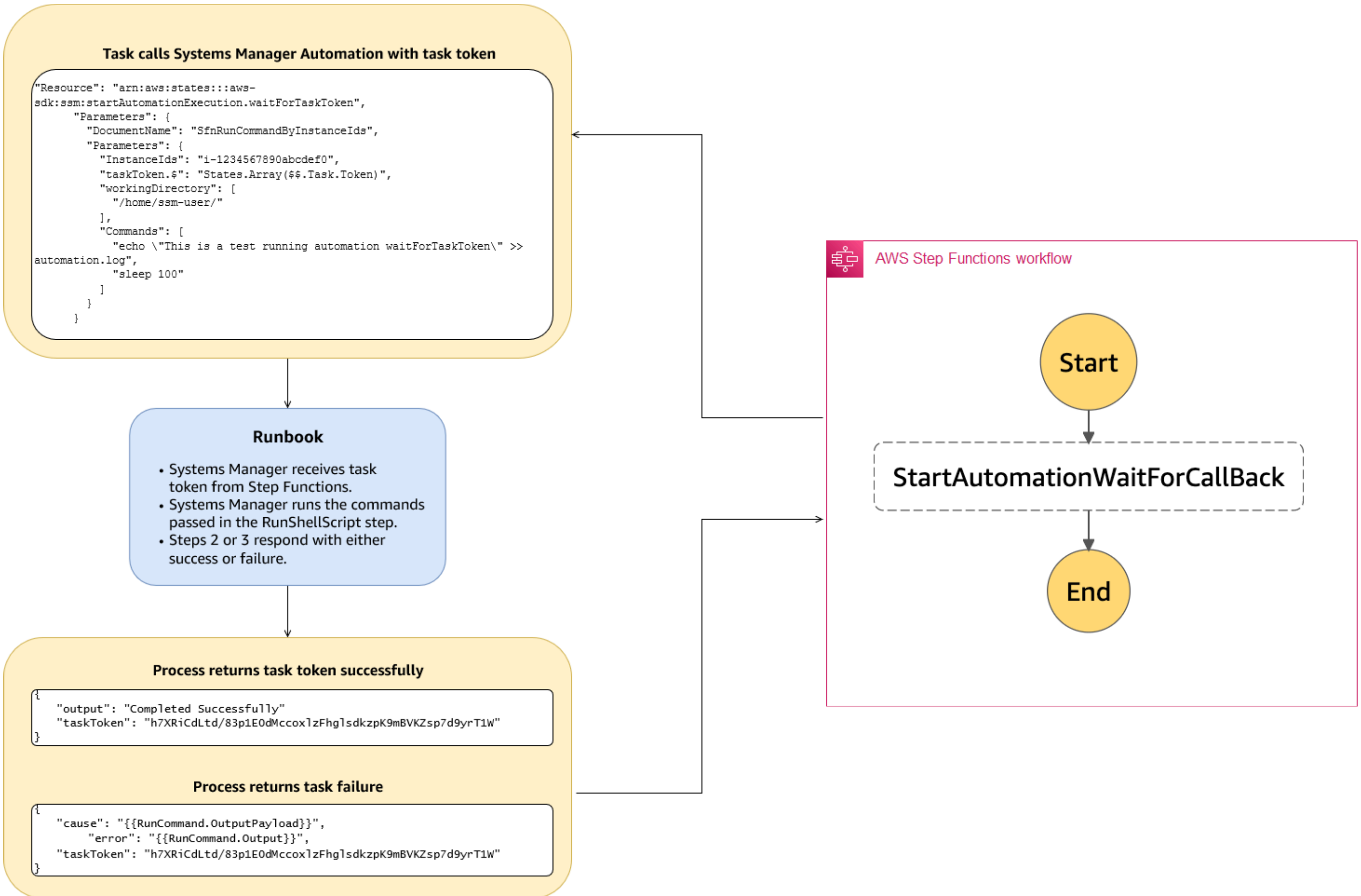
- SSM document schema version 0.3 or later
- SSM Agent version 2.3.672.0 or later

Architecture

Target technology stack

- AWS Step Functions
- AWS Systems Manager Automation

Target architecture



Automation and scale

- This pattern provides an AWS CloudFormation template that you can use to deploy the runbooks on multiple instances. (See the GitHub Step Functions and Systems Manager implementation repository.)

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.

Code

The code for this pattern is available in the GitHub Step Functions and Systems Manager implementation repository.

Epics

Create runbooks

Task	Description	Skills required
Download the CloudFormation template.	Download the `ssm-automation-documents.cfn.json` template from the `cloudformation` folder of the GitHub repository.	AWS DevOps
Create runbooks.	Sign in to the AWS Management Console, open the AWS CloudFormation console, and deploy the template. For more information about deploying CloudFormation templates, see Creating a stack on the AWS CloudFormation console in the CloudFormation documentation. The CloudFormation template deploys three resources: • `SfnRunCommandByInstanceIds` – Runbook that	AWS DevOps

Task	Description	Skills required
	lets you run `AWS-RunShellScript` or `AWS-RunPowerShellScript` by using instance IDs. • `SfnRunCommandByTargets` – Runbook that lets you run `AWS-RunShellScript` or `AWS-RunPowerShellScript` by using targets. • `SSMSyncRole` – The IAM role assumed by the runbooks.	

Create a sample state machine

Task	Description	Skills required

Create a test state machine.	Follow the instructions in the [AWS Step Functions Developer Guide](https://docs.aws.amazon.com/step-functions/latest/dg/) to create and run a state machine. For the definition, use the following code. Make sure to update the `InstanceIds` value with the ID of a valid Systems Manager-enabled instance in your account.
	`{`
	`  "Comment": "A description of my state machine",`
		AWS DevOps

Task	Description	Skills required

	`"StartAt": "StartAut omationWaitForCall Back",`
	`"States": {`
	`"StartAutomationWa itForCallBack": {`
	`"Type": "Task",`
	`"Resource": "arn:aws:states::: aws-sdk:ssm:startA utomationExecution .waitForTaskToken",`
	`"Parameters": {`
	`"DocumentName": "SfnRunCommandByIn stanceIds",`
	`"Parameters": {`
	`"Instance Ids": [`
	`"i-123456 7890abcdef0"`
	`],`
	`"taskToken. $": "States.Array($$.T ask.Token)",`
	`"workingD irectory": [`
	`"/home/ssm- user/"`
	`],`
	`"Commands": [`
	`"echo \"This is a test running automation waitForTa skToken\" >> automatio n.log",`
	`"sleep 100"`
	`],`
	`"executio nTimeout": [`
	`"10800"`
	`],`	

Task	Description	Skills required

	``
	           "delivery
	Timeout": [
	               "30"
	           ],
	           "shell": [
	               "Shell"
	           ]
	             }
	       },
	       "End": true
	     }
	   }
	}
	``
	
	This code calls the runbook to run two commands that demonstrate the `waitForTaskToken` call to Systems Manager Automation.
	
	The `shell` parameter value (`Shell` or `PowerShell`) determines whether the Automation document runs `AWS-RunShellScript` or `AWS-RunPowerShellScript`.
	
	The task writes "This is a test running automation waitForTaskToken" into the `/home/ssm-user/automation.log` file, and then sleeps for 100 seconds before it responds with the task token and releases the next task in the workflow.	

Task	Description	Skills required

	If you want to call the `SfnRunCommandByTargets` runbook instead, replace the `Parameters` section of the previous code with the following:
	``
	"Parameters": {
	          "Targets": [
	            {
	              "Key": "InstanceIds",
	              "Values": [
	                "i-02573cafcfEXAMPLE",
	                "i-0471e04240EXAMPLE"
	              ]
	            }
	          ],
	``	

Task	Description	Skills required
Update the IAM role for the state machine.	The previous step automatically creates a dedicated IAM role for the state machine. However, it doesn't grant permissions to call the runbook. Update the role by adding the following permissions: `{ "Effect": "Allow", "Action": "ssm:StartAutomationExecution", "Resource": "*" }`	AWS DevOps
Validate the synchronous calls.	Run the state machine to validate the synchronous call between Step Functions and Systems Manager Automation. For sample output, see the Additional information section.	AWS DevOps

Related resources

- Getting started with AWS Step Functions (*AWS Step Functions Developer Guide*)
- Wait for a callback with the task token (*AWS Step Functions Developer Guide*, service integration patterns)
- send_task_success and send_task_failure API calls (Boto3 documentation)
- AWS Systems Manager Automation (*AWS Systems Manager User Guide*)

Additional information

Implementation details

This pattern provides a CloudFormation template that deploys two Systems Manager runbooks:

- `SfnRunCommandByInstanceIds` runs the `AWS-RunShellScript` or `AWS-RunPowerShellScript` command by using instance IDs.
- `SfnRunCommandByTargets` runs the `AWS-RunShellScript` or `AWS-RunPowerShellScript` command by using targets.

Each runbook implements four steps to achieve a synchronous call when using the `.waitForTaskToken` option in Step Functions.

Step	Action	Description
1	`Branch`	Checks the `shell` parameter value (`Shell` or `PowerShell`) to decide whether to run `AWS-RunShellScript` for Linux or `AWS-RunPowerShellScript` for Windows.
2	`RunCommand_Shell` or `RunCommand_PowerShell`	Takes several inputs and runs the `RunShellScript` or `RunPowerShellScript` command. For more information, check the **Details** tab for the `RunCommand_Shell` or `RunCommand_PowerShell` Automation document on the Systems Manager console.
3	`SendTaskFailure`	Runs when step 2 is aborted or canceled. It calls the Step Functions [send_task_failure](#)

		API, which accepts three parameters as input: the token passed by the state machine, the failure error, and a description of the cause of the failure.
4	`SendTaskSuccess`	Runs when step 2 is successful. It calls the Step Functions [send_task_success](#) API, which accepts the token passed by the state machine as input.

Runbook parameters

`SfnRunCommandByInstanceIds` runbook:

Parameter name	Type	Optional or required	Description
`shell`	String	Required	The instances shell to decide whether to run `AWS-RunShellScript` for Linux or `AWS-RunPowerShellScript` for Windows.
`deliveryTimeout`	Integer	Optional	The time, in seconds, to wait for a command to deliver to the SSM Agent on an instance. This parameter has a minimum value of 30 (0.5 minute) and a maximum value of 2592000 (720 hours).

`executionTimeout`	String	Optional	The time, in seconds, for a command to complete before it is considered to have failed. The default value is 3600 (1 hour). The maximum value is 172800 (48 hours).
`workingDirectory`	String	Optional	The path to the working directory on your instance.
`Commands`	StringList	Required	The shell script or command to run.
`InstanceIds`	StringList	Required	The IDs of the instances where you want to run the command.
`taskToken`	String	Required	The task token to use for callback responses.

`SfnRunCommandByTargets` runbook:

Name	Type	Optional or required	Description
`shell`	String	Required	The instances shell to decide whether to run `AWS-RunShellScript` for Linux or `AWS-RunPowerShellScript` for Windows.

`deliveryTimeout`	Integer	Optional	The time, in seconds, to wait for a command to deliver to the SSM Agent on an instance. This parameter has a minimum value of 30 (0.5 minute) and a maximum value of 2592000 (720 hours).
`executionTimeout`	Integer	Optional	The time, in seconds, for a command to complete before it is considered to have failed. The default value is 3600 (1 hour). The maximum value is 172800 (48 hours).
`workingDirectory`	String	Optional	The path to the working directory on your instance.
`Commands`	StringList	Required	The shell script or command to run.

`Targets`	MapList	Required	An array of search criteria that identifies instances by using key-value pairs that you specify. For example: `[{"Key":"InstanceIds","Values":["i-02573cafcfEXAMPLE", "i-0471e04240EXAMPLE"]}]`
`taskToken`	String	Required	The task token to use for callback responses.

Sample output

The following table provides sample output from the step function. It shows that the total run time is over 100 seconds between step 5 (`TaskSubmitted`) and step 6 (`TaskSucceeded`). This demonstrates that the step function waited for the `sleep 100` command to finish before moving to the next task in the workflow.

ID	Type	Step	Resource	Elapsed Time (ms)	Timestamp
1	`ExecutionStarted`		-	0	Mar 11, 2022 02:50:34.303 PM
2	`TaskStateEntered`	`StartAutomationWaitForCallBack`	-	40	Mar 11, 2022 02:50:34.343 PM

3	TaskScheduled	StartAutomationWaitForCallBack	-	40	Mar 11, 2022 02:50:34.343 PM
4	TaskStarted	StartAutomationWaitForCallBack	-	154	Mar 11, 2022 02:50:34.457 PM
5	TaskSubmitted	StartAutomationWaitForCallBack	-	657	Mar 11, 2022 02:50:34.960 PM
6	TaskSucceeded	StartAutomationWaitForCallBack	-	103835	Mar 11, 2022 02:52:18.138 PM
7	TaskStateExited	StartAutomationWaitForCallBack	-	103860	Mar 11, 2022 02:52:18.163 PM
8	ExecutionSucceeded		-	103897	Mar 11, 2022 02:52:18.200 PM

Run parallel reads of S3 objects by using Python in an AWS Lambda function

Eduardo Bortoluzzi, Amazon Web Services

Summary

You can use this pattern to retrieve and summarize a list of documents from Amazon Simple Storage Service (Amazon S3) buckets in real time. The pattern provides example code to parallel read objects from S3 buckets on Amazon Web Services (AWS). The pattern showcases how to efficiently run I/O bound tasks with AWS Lambda functions using Python.

A financial company used this pattern in an interactive solution to manually approve or reject correlated financial transactions in real time. The financial transaction documents were stored in an S3 bucket related to the market. An operator selected a list of documents from the S3 bucket, analyzed the total value of the transactions that the solution calculated, and decided to approve or reject the selected batch.

I/O bound tasks support multiple threads. In this example code, the concurrent.futures.ThreadPoolExecutor is used with a maximum of 30 simultaneous threads, even though Lambda functions support up to 1,024 threads (with one of those threads being your main process). This limit is because too many threads create latency issues due to context switching and utilization of computing resources. You also need to increase the maximum pool connections in `botocore` so that all threads can perform the S3 object download simultaneously.

The example code uses one 8.3 KB object, with JSON data, in an S3 bucket. The object is read multiple times. After the Lambda function reads the object, the JSON data is decoded to a Python object. In December 2024, the result after running this example was 1,000 reads processed in 2.3 seconds and 10,000 reads processed in 27 seconds using a Lambda function configured with 2,304 MB of memory. AWS Lambda supports memory configurations from 128 MB to 10,240 MB (10 GB), though increasing the Lambdamemory beyond 2,304 MB didn't help to decrease the time to run this particular I/O-bound task.

The AWS Lambda Power Tuning tool was used to test different Lambda memory configurations and verify the best performance-to-cost ratio for the task. For test results, see the Additional information section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Proficiency with Python development

Limitations

- A Lambda function can have at most 1,024 execution processes or threads.
- New AWS accounts have a Lambda memory limit of 3,008 MB. Adjust the AWS Lambda Power Tuning tool accordingly. For more information, see the Troubleshooting section.
- Amazon S3 has a limit of 5,500 GET/HEAD requests per second per partitioned prefix.

Product versions

- Python 3.9 or later
- AWS Cloud Development Kit (AWS CDK) v2
- AWS Command Line Interface (AWS CLI) version 2
- AWS Lambda Power Tuning 4.3.6 (optional)

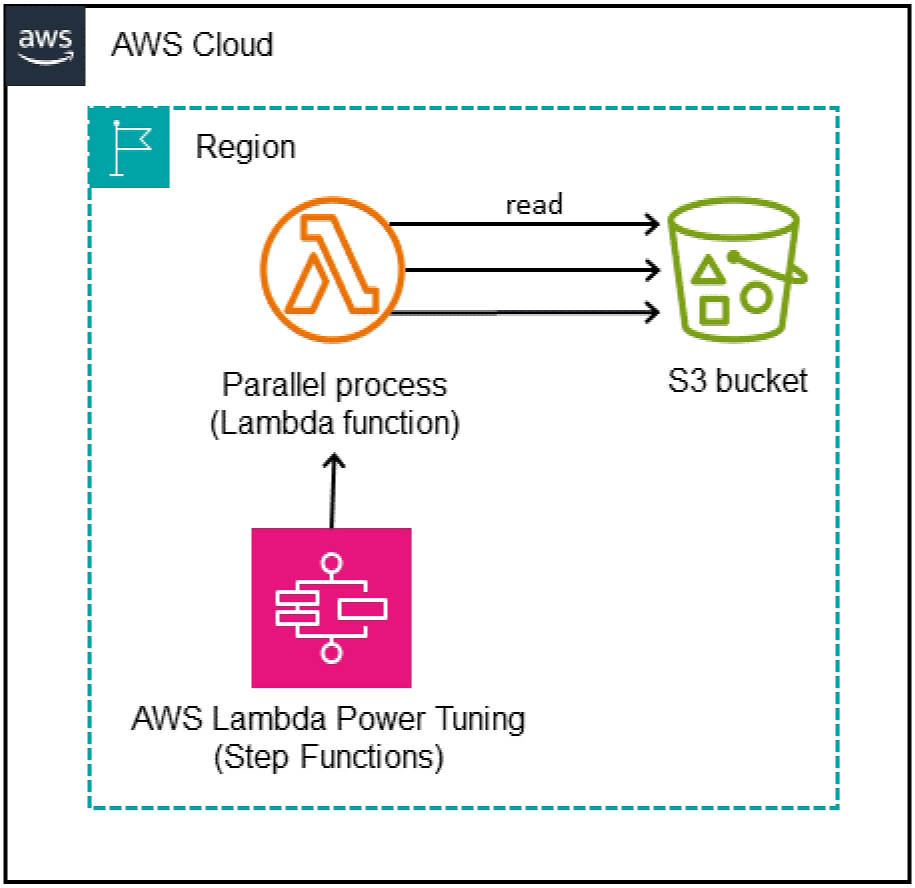
Architecture

Target technology stack

- AWS Lambda
- Amazon S3
- AWS Step Functions (if AWS Lambda Power Tuning is deployed)

Target architecture

The following diagram shows a Lambda function that reads objects from an S3 bucket in parallel. The diagram also has a Step Functions workflow for the AWS Lambda Power Tuning tool to fine-tune the Lambda function memory. This fine-tuning helps to achieve a good balance between cost and performance.



Automation and scale

The Lambda functions scale fast when required. To avoid 503 Slow Down errors from Amazon S3 during high demand, we recommend putting some limits on the scaling.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) v2 is a software development framework that helps you define and provision AWS Cloud infrastructure in code. The example infrastructure was created to be deployed with AWS CDK.
- AWS Command Line InterfaceAWS CLI is an open source tool that helps you interact with AWS services through commands in your command-line shell. In this pattern, AWS CLI version 2 is used to upload an example JSON file.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

- Amazon Simple Storage Service Amazon S3 is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

Other tools

- Python is a general-purpose computer programming language. The reuse of idle worker threads was introduced in Python version 3.8, and the Lambda function code in this pattern was created for Python version 3.9 and later.

Code repository

The code for this pattern is available in the aws-lambda-parallel-download GitHub repository.

Best practices

- This AWS CDK construct relies on your AWS account's user permissions to deploy the infrastructure. If you plan to use AWS CDK Pipelines or cross-account deployments, see Stack synthesizers.
- This example application doesn't have the access logs enabled at the S3 bucket. It's a best practice to enable access logs in production code.

Epics

Prepare the development environment

Task	Description	Skills required
Check the Python installed version.	This code has been tested specifically on Python 3.9 and Python 3.13, and it should work on all versions between these releases. To check your Python version, run `python3 -V` in your terminal, and	Cloud architect

Task	Description	Skills required
	install a newer version if needed. To verify that the required modules are installed, run `python3 -c "import pip, venv"`. No error message means the modules are properly installed and you're ready to run this example.	
Install AWS CDK.	To install the AWS CDK if it isn't already installed, follow the instructions at [Getting started with the AWS CDK](). To confirm that the installed AWS CDK version is 2.0 or later, run `cdk –version`.	Cloud architect
Bootstrap your environment.	To bootstrap your environme nt, if it hasn’t already been done, follow the instructions at [Bootstrap your environme nt for use with the AWS CDK]().	Cloud architect

Clone the example repository

Task	Description	Skills required
Clone the repository.	To clone the latest version of the repository, run the following command:	Cloud architect

Task	Description	Skills required
	`git clone --depth 1 --branch v1.2.0 \ git@github.com:aws-samples/aws-lambda-parallel-download.git`	
Change the working directory to the cloned repository.	Run the following command: `cd aws-lambda-parallel-download`	Cloud architect
Create the Python virtual environment.	To create a Python virtual environment, run the following command: `python3 -m venv .venv`	Cloud architect
Activate the virtual environment.	To activate the virtual environment, run the following command: `source .venv/bin/activate`	Cloud architect
Install the dependencies.	To install the Python dependencies, run the `pip` command: `pip install -r requirements.txt`	Cloud architect

Task	Description	Skills required
Browse the code.	(Optional) The example code that downloads an object from the S3 bucket is at `resources/parallel.py` . The infrastructure code is in the `parallel_download` folder.	Cloud architect

Deploy and test the app

Task	Description	Skills required
Deploy the app.	Run `cdk deploy`. Write down the AWS CDK outputs: • `ParallelDownloadStack.LambdaFunctionARN` • `ParallelDownloadStack.SampleS3BucketName` • `ParallelDownloadStack.StateMachineARN`	Cloud architect
Upload an example JSON file.	The repository contains an example JSON file of about 9 KB. To upload the file to the S3 bucket of the created stack, run the following command:	Cloud architect

Task	Description	Skills required
	`aws s3 cp sample.json s3://<ParallelDownloadStack.SampleS3BucketName>` Replace `<ParallelDownloadStack.SampleS3BucketName>` with the corresponding value from the AWS CDK output.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Run the app.</td><td>To run the app, do the following:
Sign in to the AWS Management Console, navigate to the Lambda console, and locate the Lambda function that has the ARN from the AWS CDK output <code>ParallelDownloadStack.LambdaFunctionARN</code> .
On the Test tab, change the Event JSON to the following:<pre>{"objectKey":
 "sample.json"}</pre>
Choose Test.
To see the result, choose details. The details will show the statistics of the parallel download, the information of the run, and the logs.</td><td>Cloud architect</td></tr>
<tr><td>Add the number of downloads.</td><td>(Optional) To run 1,500 get object calls, use the following JSON in Event JSON of the <code>Test</code> parameter:<pre>{"repeat": 1500,
 "objectKey": "sample.json"}</pre></td><td>Cloud architect</td></tr>
</table>

Optional: Run AWS Lambda Power Tuning

Task	Description	Skills required
Run the AWS Lambda Power Tuning tool.	1. Sign in to the console, and navigate to [Step Functions](https://console.aws.amazon.com/states/home). 2. Locate the state machine with the ARN from the AWS CDK output `ParallelDownloadStack.StateMachineARN`. 3. Choose **Start execution**, and paste the following JSON: `{ "lambdaARN": "<ParallelDownloadStack.LambdaFunctionARN>", "num": 10, "strategy": "balanced", "payload": {"repeat": 2000, "objectKey": "sample.json"} }` Remember to replace `<ParallelDownloadStack.LambdaFunctionARN>` with the value from the AWS CDK output. At the end of the run, the result will be on the	Cloud architect

Task	Description	Skills required
	Execution input and output tab.	
View the AWS Lambda Power Tuning results in a graph.	On the **Execution input and output** tab, copy the `visualization` property link, and paste it in a new browser tab.	Cloud architect

Clean up

Task	Description	Skills required
Remove the objects from the S3 bucket.	Before you destroy the deployed resources, you remove all the objects from the S3 bucket: `aws s3 rm s3://<ParallelDownloadStack.SampleS3BucketName> \ --recursive` Remember to replace `<ParallelDownloadStack.SampleS3BucketName>` with the value from the AWS CDK outputs.	Cloud architect
Destroy the resources.	To destroy all the resources that were created for this pilot, run the following command:	Cloud architect

Task	Description	Skills required
	`cdk destroy`	

Troubleshooting

Issue	Solution
`'MemorySize' value failed to satisfy constraint: Member must have value less than or equal to 3008`	For new accounts, you might not be able to configure more than 3,008 MB in your Lambda functions. To test using AWS Lambda Power Tuning, add the following property at the input JSON when you are starting the Step Functions execution: `"powerValues": [ 512, 1024, 1536, 2048, 2560, 3008 ]`

Related resources

- Python – concurrent.futures.ThreadPoolExecutor
- Lambda quotas – Function configuration, deployment, and execution
- Working with the AWS CDK in Python
- Profiling functions with AWS Lambda Power Tuning

Additional information

Code

The following code snippet performs the parallel I/O processing:

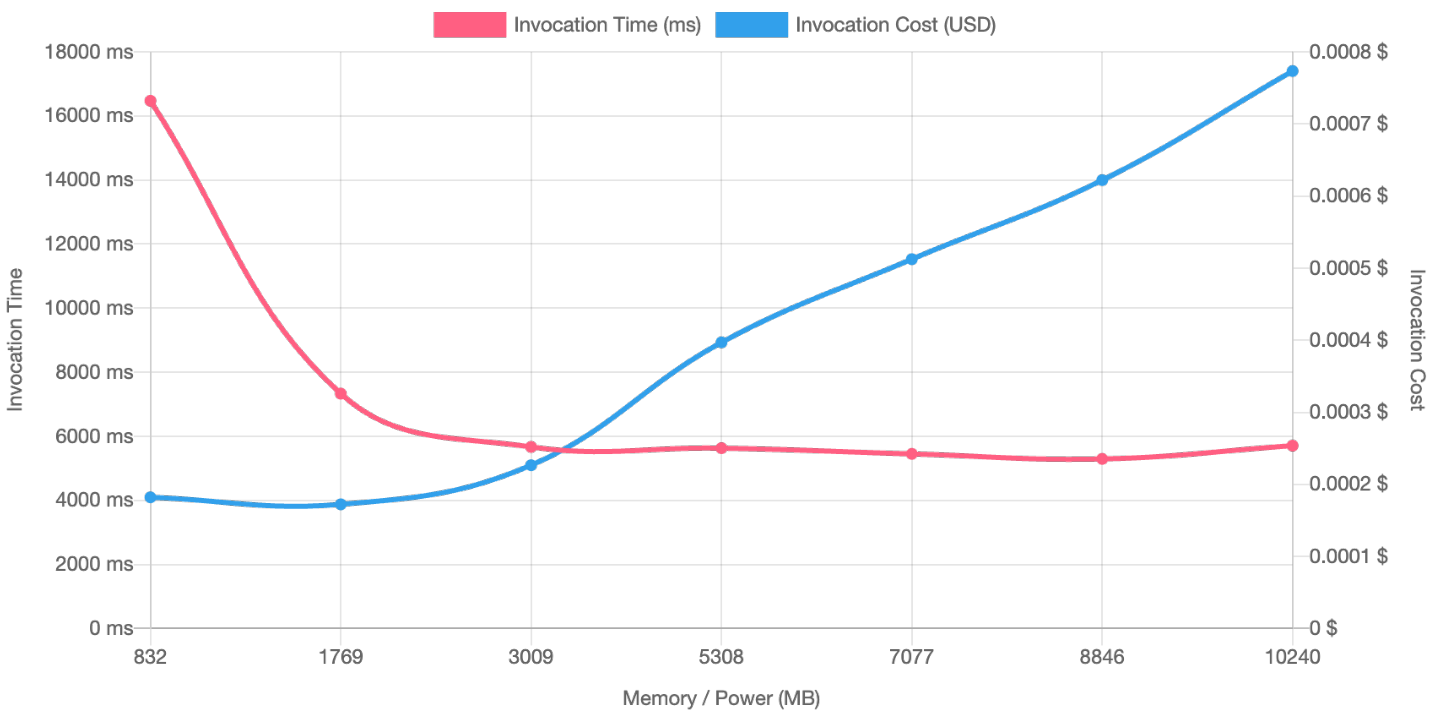
```
with ThreadPoolExecutor(max_workers=MAX_WORKERS) as executor:
  for result in executor.map(a_function, (the_arguments)):
    ...
```

The `ThreadPoolExecutor` reuses the threads when they become available.

Testing and results

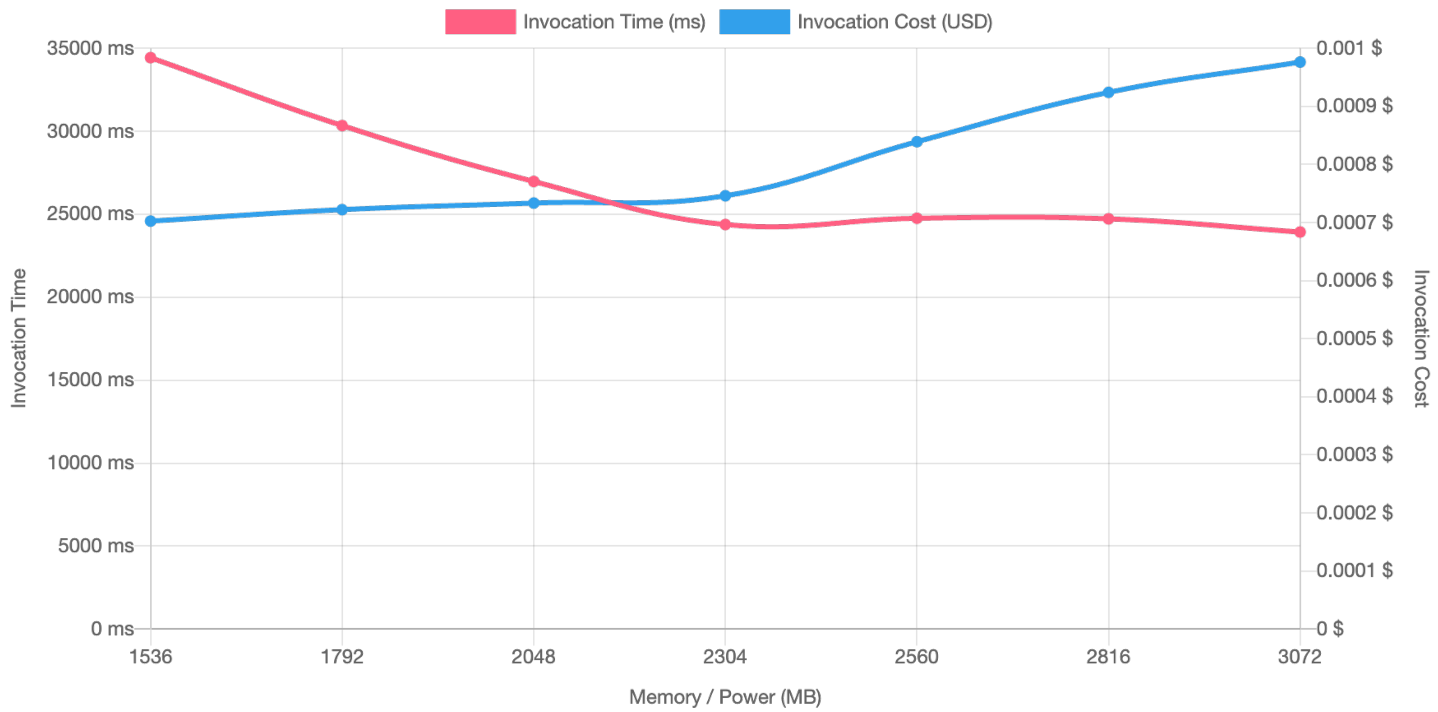
These tests were conducted in December 2024.

The first test processed 2,500 object reads, with the following result.



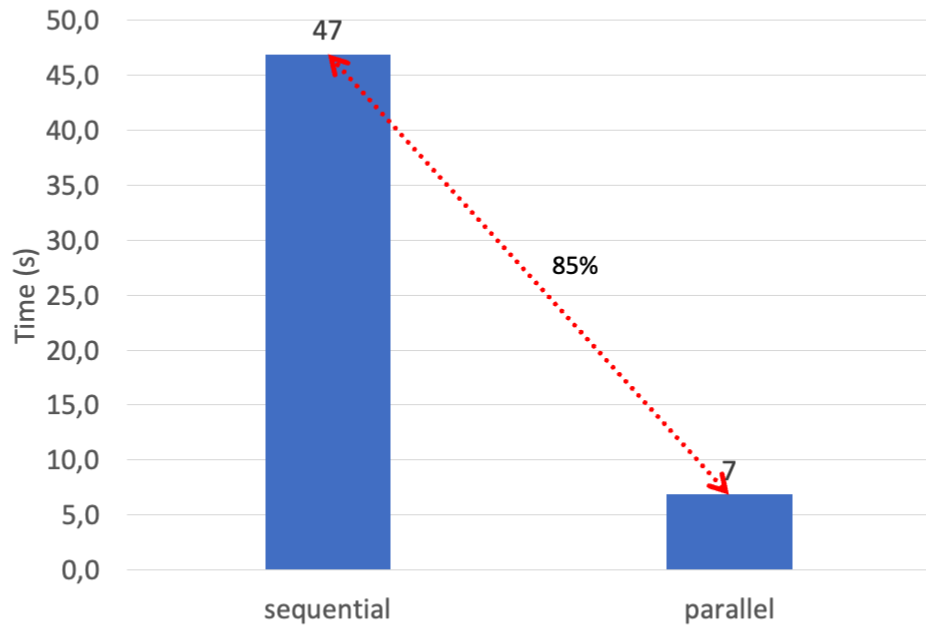
Starting at 3,009 MB, the processing-time level stayed almost the same for any memory increase, but the cost increased as the memory size increased.

Another test investigated the range between 1,536 MB and 3,072 MB of memory, using values that were multiples of 256 MB and processing 10,000 object reads, with the following results.



The best performance-to-cost ratio was with the 2,304 MB memory Lambda configuration.

For comparison, a sequential process of 2,500 object reads took 47 seconds. The parallel process using the 2,304 MB Lambda configuration took 7 seconds, which is 85 percent less.


50,0
45,0
40,0
35,0
30,0
25,0
20,0
15,0
10,0
5,0
0,0
Time (s)
47
85%
7
sequential
parallel

Send telemetry data from AWS Lambda to OpenSearch for real-time analytics and visualization

Tabby Ward, Guy Bachar, and David Kilzer, Amazon Web Services

Summary

Modern applications are becoming increasingly distributed and event-driven, which reinforces the need for real-time monitoring and observability. AWS Lambda is a serverless computing service that plays a crucial role in building scalable and event-driven architectures. However, monitoring and troubleshooting Lambda functions can be challenging if you rely solely on Amazon CloudWatch Logs, which can introduce latency and limited retention periods.

To address this challenge, AWS introduced the Lambda Telemetry API, which enables Lambda functions to send telemetry data directly to third-party monitoring and observability tools. This API supports real-time streaming of logs, metrics, and traces, and provides a comprehensive and timely view of the performance and health of your Lambda functions.

This pattern explains how to integrate the Lambda Telemetry API with [OpenSearch](), which is an open-source, distributed search and analytics engine. OpenSearch offers a powerful and scalable platform for ingesting, storing, and analyzing large volumes of data, which makes it an ideal choice for Lambda telemetry data. Specifically, this pattern demonstrates how to send logs from a Lambda function that's written in Python directly to an OpenSearch cluster by using a Lambda extension that's provided by AWS. This solution is flexible and customizable, so you can create your own Lambda extension or alter the sample source code to change the output format as desired.

The pattern explains how to set up and configure the Lambda Telemetry API integration with OpenSearch, and includes best practices for security, cost optimization, and scalability. The objective is to help you gain deeper insights into your Lambda functions and enhance the overall observability of your serverless applications.

> Note: This pattern focuses on integrating the Lambda Telemetry API with managed OpenSearc h. However, the principles and techniques discussed are also applicable to self-managed OpenSearch and Elasticsearch.

Prerequisites and limitations

Before you begin the integration process, make sure that you have the following prerequisites in place:

AWS account: An active AWS account with appropriate permissions to create and manage the following AWS resources:

- AWS Lambda
- AWS Identity and Access Management (IAM)
- Amazon OpenSearch Service (if you're using a managed OpenSearch cluster)

OpenSearch cluster:

- You can use an existing self-managed OpenSearch cluster or a managed service such as OpenSearch Service.
- If you're using OpenSearch Service, set up your OpenSearch cluster by following the instructions in Getting started with Amazon OpenSearch Service in the OpenSearch Service documentation.
- Make sure that the OpenSearch cluster is accessible from your Lambda function and is configured with the necessary security settings, such as access policies, encryption, and authentication.
- Configure the OpenSearch cluster with the necessary index mappings and settings to ingest the Lambda telemetry data. For more information, see Loading streaming data into Amazon OpenSearch Service in the OpenSearch Service documentation.

Network connectivity:

- Ensure that your Lambda function has the necessary network connectivity to access the OpenSearch cluster. For guidance on how to configure virtual private cloud (VPC) settings, see Launching your Amazon OpenSearch Service domains within a VPC in the OpenSearch Service documentation.

IAM roles and policies:

- Create an IAM role with the necessary permissions for your Lambda function to access the OpenSearch cluster and access your credentials stored in AWS Secrets Manager.

- Attach the appropriate IAM policies to the role, such as the `AWSLambdaBasicExecutionRole` policy and any additional permissions required to interact with OpenSearch.
- Verify that the IAM permissions granted to your Lambda function allow it to write data to the OpenSearch cluster. For information about managing IAM permissions, see Defining Lambda function permissions with an execution role in the Lambda documentation.

Programming language knowledge:

- You will need basic knowledge of Python (or the programming language of your choice) to understand and modify the sample code for the Lambda function and the Lambda extension.

Development environment:

- Set up a local development environment with the necessary tools and dependencies for building and deploying Lambda functions and extensions.

AWS CLI or AWS Management Console:

- Install and configure the AWS Command Line Interface (AWS CLI) or use the AWS Management Console with appropriate credentials to interact with the required AWS services.

Monitoring and logging:

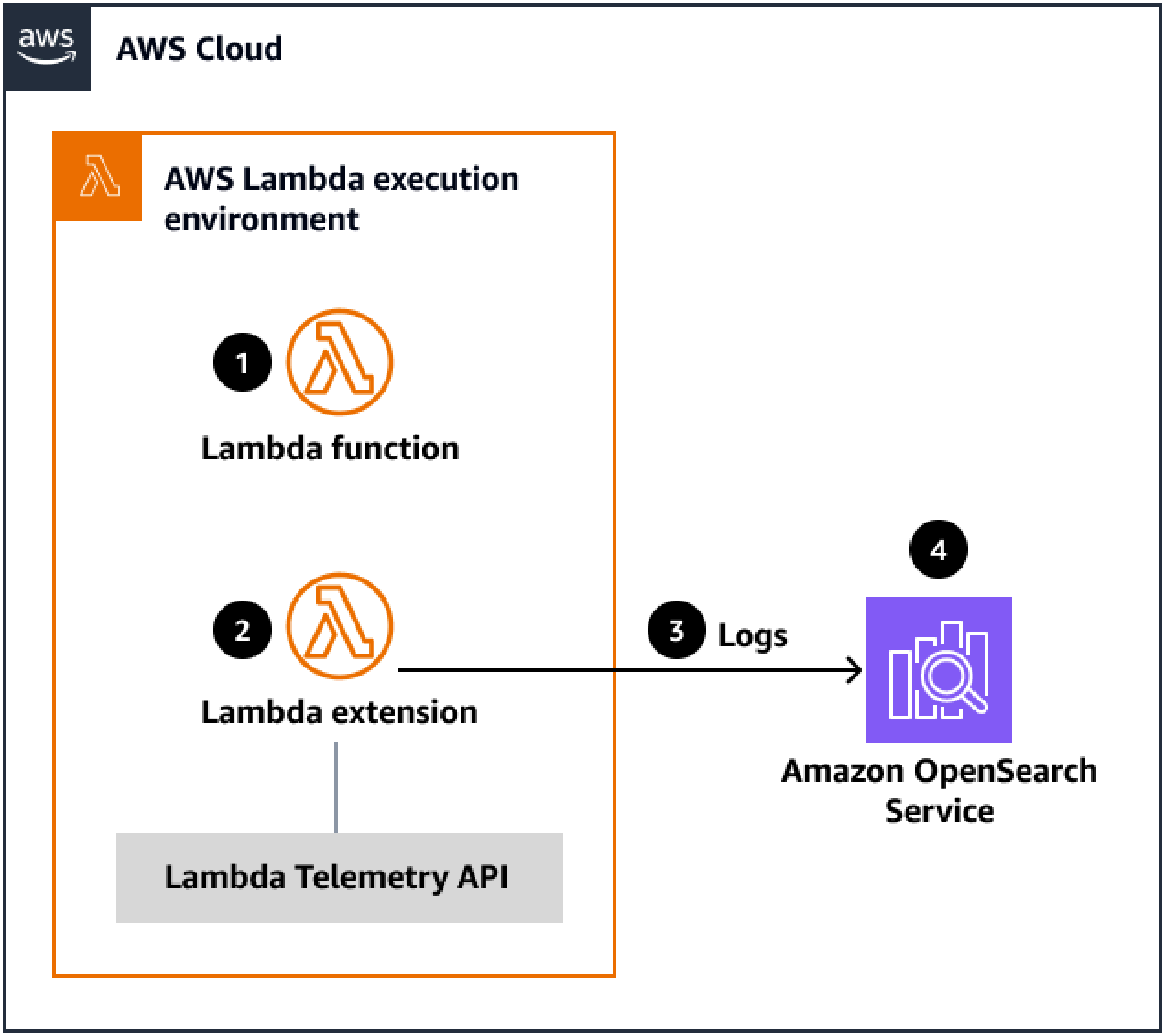
- Become familiar with monitoring and logging best practices on AWS, including services such as Amazon CloudWatch and AWS CloudTrail for monitoring and auditing purposes.
- Check CloudWatch Logs for your Lambda function to identify any errors or exceptions related to the Lambda Telemetry API integration. For troubleshooting guidance, see the Lambda Telemetry API documentation.

Architecture

This pattern uses OpenSearch Service to store logs and telemetry data that are generated by Lambda functions. This approach enables you to quickly stream logs directly to your OpenSearch cluster, which reduces the latency and costs associated with using CloudWatch Logs as an intermediary.

> **Note**
>
> Your Lambda extension code can push telemetry to OpenSearch Service, either by directly using the OpenSearch API or by using an OpenSearch client library. The Lambda extension can use the bulk operations supported by the OpenSearch API to batch telemetry events together and send them to OpenSearch Service in a single request.

The following workflow diagram illustrates the log workflow for Lambda functions when you use an OpenSearch cluster as the endpoint.



The architecture includes these components:

- Lambda function: The serverless function that generates logs and telemetry data during execution.
- Lambda extension: A Python-based extension that uses the Lambda Telemetry API to integrate directly with the OpenSearch cluster. This extension runs alongside the Lambda function in the same execution environment.
- Lambda Telemetry API: The API that enables Lambda extensions to send telemetry data, including logs, metrics, and traces, directly to third-party monitoring and observability tools.

- Amazon OpenSearch Service cluster: A managed OpenSearch cluster that's hosted on AWS. This cluster is responsible for ingesting, storing, and indexing the log data streamed from the Lambda function through the Lambda extension.

The workflow consists of these steps:

1. The Lambda function is called, and generates logs and telemetry data during its execution.
2. The Lambda extension runs alongside the function to capture the logs and telemetry data by using the Lambda Telemetry API.
3. The Lambda extension establishes a secure connection with the OpenSearch Service cluster and streams the log data in real time.
4. The OpenSearch Service cluster ingests, indexes, and stores the log data to make it available for search, analysis, and visualization through the use of tools such as Kibana or other compatible applications.

By circumventing CloudWatch Logs and sending log data directly to the OpenSearch cluster, this solution provides several benefits:

- Real-time log streaming and analysis, enabling faster troubleshooting and improved observability.
- Reduced latency and potential retention limitations associated with CloudWatch Logs.
- Flexibility to customize the Lambda extension or create your own extension for specific output formats or additional processing.
- Integration with the search, analytics, and visualization capabilities of OpenSearch Service for log analysis and monitoring.

The Epics section provides step-by-step instructions for setting up the Lambda extension, configuring the Lambda function, and integrating with the OpenSearch Service cluster. For security considerations, cost optimization strategies, and tips for monitoring and troubleshooting the solution, see the Best practices section.

Tools

AWS services

- AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon OpenSearch Service is a fully managed service provided by AWS that makes it easy to deploy, operate, and scale OpenSearch clusters in the cloud.
- Lambda extensions extends the functionality of your Lambda functions by running custom code alongside them. You can use Lambda extensions to integrate Lambda with various monitoring, observability, security, and governance tools.
- AWS Lambda Telemetry API enables you to use extensions to capture enhanced monitoring and observability data directly from Lambda and send it to a destination of your choice.
- AWS CloudFormation helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications.

Code repositories

- AWS Lambda Extensions includes demos and sample projects from AWS and AWS Partners to help you get started with building your own extensions.
- Example Lambda Telemetry Integrations for OpenSearch provides a sample Lambda extension that demonstrates how to send logs from a Lambda function to an OpenSearch cluster.

Other tools

- OpenSearch is an open-source distributed search and analytics engine that provides a powerful platform for ingesting, storing, and analyzing large volumes of data.
- Kibana is an open-source data visualization and exploration tool that you can use with OpenSearch. Note that the implementation of visualization and analytics is beyond the scope of this pattern. For more information, see the Kibana documentation and other resources.

Best practices

When you integrate the Lambda Telemetry API with OpenSearch, consider the following best practices.

Security and access control

- **Secure communication**: Encrypt all communications between your Lambda functions and the OpenSearch cluster by using HTTPS. Configure the necessary SSL/TLS settings in your Lambda extension and OpenSearch configuration.
- **IAM permissions**:
  - Extensions run in the same execution environment as the Lambda function, so they inherit the same level of access to resources such as the file system, networking, and environment variables.
  - Grant the minimum necessary IAM permissions to your Lambda functions to access the Lambda Telemetry API and write data to the OpenSearch cluster. Use the [principle of least privilege] to limit the scope of permissions.
- **OpenSearch access control**: Implement fine-grained access control in your OpenSearch cluster to restrict access to sensitive data. Use the built-in security features, such as user authentication, role-based access control, and index-level permissions, in OpenSearch.
- **Trusted extensions**: Always install extensions from a trusted source only. Use infrastructure as code (IaC) tools such as AWS CloudFormation to simplify the process of attaching the same extension configuration, including IAM permissions, to multiple Lambda functions. IaC tools also provide an audit record of the extensions and versions used previously.
- **Sensitive data handling**: When building extensions, avoid logging sensitive data. Sanitize payloads and metadata before logging or persisting them for audit purposes.

Cost optimization

- **Monitoring and alerting**: Set up monitoring and alerting mechanisms to track the volume of data being sent to OpenSearch from your Lambda functions. This will help you identify and address any potential cost overruns.
- **Data retention**: Carefully consider the appropriate data retention period for your Lambda telemetry data in OpenSearch. Longer retention periods can increase storage costs, so balance your observability needs with cost optimization.
- **Compression and indexing**: Enable data compression and optimize your OpenSearch indexing strategy to reduce the storage footprint of your Lambda telemetry data.
- **Reduced reliance on CloudWatch**: By integrating the Lambda Telemetry API directly with OpenSearch, you can potentially reduce your reliance on CloudWatch Logs, which can result in cost savings. This is because the Lambda Telemetry API enables you to send logs directly to OpenSearch, which bypasses the need to store and process the data in CloudWatch.

Scalability and reliability

- **Asynchronous processing**: Use asynchronous processing patterns, such as Amazon Simple Queue Service (Amazon SQS) or Amazon Kinesis, to decouple the Lambda function execution from the OpenSearch data ingestion. This helps maintain the responsiveness of your Lambda functions and improves the overall reliability of the system.
- **OpenSearch cluster scaling**: Monitor the performance and resource utilization of your OpenSearch cluster, and scale it up or down as needed to handle the increasing volume of Lambda telemetry data.
- **Failover and disaster recovery**: Implement a robust disaster recovery strategy for your OpenSearch cluster, including regular backups and the ability to quickly restore data in the event of a failure.

Observability and monitoring

- **Dashboards and visualizations**: Use Kibana or other dashboard tools to create custom dashboards and visualizations that provide insights into the performance and health of your Lambda functions based on the telemetry data in OpenSearch.
- **Alerting and notifications**: Set up alerts and notifications to proactively monitor for anomalies, errors, or performance issues in your Lambda functions. Integrate these alerts and notifications with your existing incident management processes.
- **Tracing and correlation**: Ensure that your Lambda telemetry data includes relevant tracing information, such as request IDs or correlation IDs, to enable end-to-end observability and troubleshooting across your distributed serverless applications.

By following these best practices, you can ensure that your integration of the Lambda Telemetry API with OpenSearch is secure, cost-effective, and scalable, and provides comprehensive observability for your serverless applications.

Epics

Build and deploy the Lambda extension layer

Task	Description	Skills required
Download the source code.	Download the sample extensions from the AWS	App developer, Cloud architect

Task	Description	Skills required
	Lambda Extensions repository.	
Navigate to the `python-example-telemetry-opensearch-extension` folder.	The AWS Lambda Extensions repository that you downloaded contains numerous examples for several use cases and language runtimes. Navigate to the python-example-telemetry-opensearch-extension folder to use the Python OpenSearch extension, which sends logs to OpenSearch.	App developer, Cloud architect
Add permissions to execute the extension endpoint.	Run the following command to make the extension endpoint executable: `chmod +x python-example-telemetry-opensearch-extension/extension.py`	App developer, Cloud architect

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install the extension dependencies locally.</td><td>Run the following command to install local dependencies for the Python code:
<pre>pip3 install -r python-example-telemetry-opensearch-extension/requirements.txt -t ./python-example-telemetry-opensearch-extension/</pre>
These dependencies will be mounted along with the extension code.</td><td>App developer, Cloud architect</td></tr>
</table>

Task	Description	Skills required

Create a .zip package for the extension to deploy it as a layer.	The extension .zip file should contain a root directory called `extensions/` , where the extension executable is located, and another root directory called `python-example-telemetry-opensearch-extension/` , where the core logic of the extension and its dependencies are located.
	
	Create the .zip package for the extension:
	
	`chmod +x extensions/python-example-telemetry-opensearch-extension`
	`zip -r extension.zip extensions python-example-telemetry-opensearch-extension`
		App developer, Cloud architect

Deploy the extension as a Lambda layer.	Publish the layer by using your extension .zip file and the following command:
	
	`aws lambda publish-layer-version \`
	`--layer-name "python-example-telemetry-opensearch-extension" \`
	`--zip-file "fileb://extension.zip"`
		App developer, Cloud architect

Integrate the extension into your function

Task	Description	Skills required
Add the layer to your function.	1. Sign in to the AWS Management Console and open the [Functions page](#) of the AWS Lambda console. 2. Select your function. 3. Under **Layers,** choose **Add a layer**. 4. Under **Choose a layer**, choose **Custom layers** as a layer source and add your layer. For more information about adding a layer to your Lambda function, see the [Lambda documentation](#).	App developer, Cloud architect
Set the environment variables for the function.	On the function page, choose the **Configuration** tab and add the following environme nt variables to your function: • URL – The URI of your OpenSearch endpoint where your logs will be sent. • `AUTH_SECRET` – The ARN of your OpenSearc h credentials stored in AWS Secrets Manager. This should be stored as a key-value pair and have	App developer, Cloud architect

Task	Description	Skills required
	two keys: `username` and `password`. • `PLATFORM_INDEX` , `FUNCTION_INDEX` , and `EXTENSION_INDEX` – The names of indexes that will store your telemetry data, function logs, and extension logs. Make sure that they adhere to the proper naming criteria. Otherwise, your indexes won't be created. • `DISPATCH_MIN_BATCH_SIZE` – The number of log events that you want to batch. However, when the function shuts down, your logs will be dispatched regardless of this setting.	

Add logging statements and test your function

Task	Description	Skills required
Add logging statements to your function.	Add logging statements to your function by using one of the built-in logging mechanisms or your logging module of choice. Here are examples of logging messages in Python:	App developer, Cloud architect

Task	Description	Skills required
	``` print("Your Log Message Here") logger = logging.getLogger(__name__) logger.info("Test Info Log.") logger.error("Test Error Log.") ```	
Test your function.	1. On the function page, choose the **Test** tab. 2. Create a test event for your function and run the test. For more information, see [Testing Lambda functions in the console](#) in the Lambda documentation. You should see **Executing function: succeeded** if everything works properly.	App developer, Cloud architect

View your logs in OpenSearch

Task	Description	Skills required
Query your indexes.	In OpenSearch, run the following command to query your indexes: ``` SELECT * FROM index-name ```	Cloud architect

Task	Description	Skills required
	Your logs should be displayed in the query results.	

Troubleshooting

Issue	Solution
Connectivity issues	• Confirm that your Lambda function has the necessary network connectivity to access the OpenSearch cluster. See the OpenSearch Service documentation for guidance on configuring VPC settings. • Verify that the IAM permissions granted to your Lambda function allow it to write data to the OpenSearch cluster. Review the Lambda documentation for information about managing IAM permissions.
Data ingestion errors	• Check CloudWatch Logs for your Lambda function to identify any errors or exceptions related to the Lambda Telemetry API integration. See the Lambda Telemetry API documentation for troubleshooting guidance. • Verify that the OpenSearch cluster is configured correctly and has the necessary index mappings and settings to ingest the Lambda telemetry data. Consult the OpenSearch documentation for more information.

Related resources

- Example Lambda Telemetry Integrations for OpenSearch (GitHub repository)
- Augment Lambda functions using Lambda extensions (Lambda documentation)
- Lambda Telemetry API (Lambda documentation)
- Introducing the AWS Lambda Telemetry API (AWS blog post)
- Integrating the AWS Lambda Telemetry API with Prometheus and OpenSearch (AWS blog post)

Additional information

Altering the log structure

The extension sends logs as a nested document to OpenSearch by default. This allows you to perform nested queries to retrieve individual column values.

If the default log output doesn't meet your specific needs, you can customize it by modifying the source code of the Lambda extension that's provided by AWS. AWS encourages customers to adapt the output to suit their business requirements. To change the log output, locate the `dispatch_to_opensearch` function in the `telemetry_dispatcher.py` file within the extension's source code and make the necessary alterations.

Set up a serverless cell router for a cell-based architecture

Mian Tariq and Ioannis Lioupras, Amazon Web Services

Summary

As the entry point to a global cell-based application's system, the cell router is responsible for efficiently assigning users to the appropriate cells and providing the endpoints to the users. The cell router handles functions such as storing user-to-cell mappings, monitoring cell capacity, and requesting new cells when needed. It's important to maintain cell-router functionality during potential disruptions.

The cell-router design framework in this pattern focuses on resilience, scalability, and overall performance optimization. The pattern uses static routing, where clients cache endpoints upon initial login and communicate directly with cells. This decoupling enhances system resilience by helping to ensure uninterrupted functionality of the cell-based application during a cell-router impairment.

This pattern uses an AWS CloudFormation template to deploy the architecture. For details about what the template deploys, or to deploy the same configuration by using the AWS Management Console, see the [Additional information](#) section.

> **Important**
>
> The demonstration, the code, and the AWS CloudFormation template presented in this pattern are intended for explanatory purposes only. The material provided is solely for the purpose of illustrating the design pattern and aiding in comprehension. The demo and code are not production-ready and should not be used for any live production activities. Any attempt to use the code or demo in a production environment is strongly discouraged and is at your own risk. We recommend consulting with appropriate professionals and performing thorough testing before implementing this pattern or any of its components in a production setting.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account

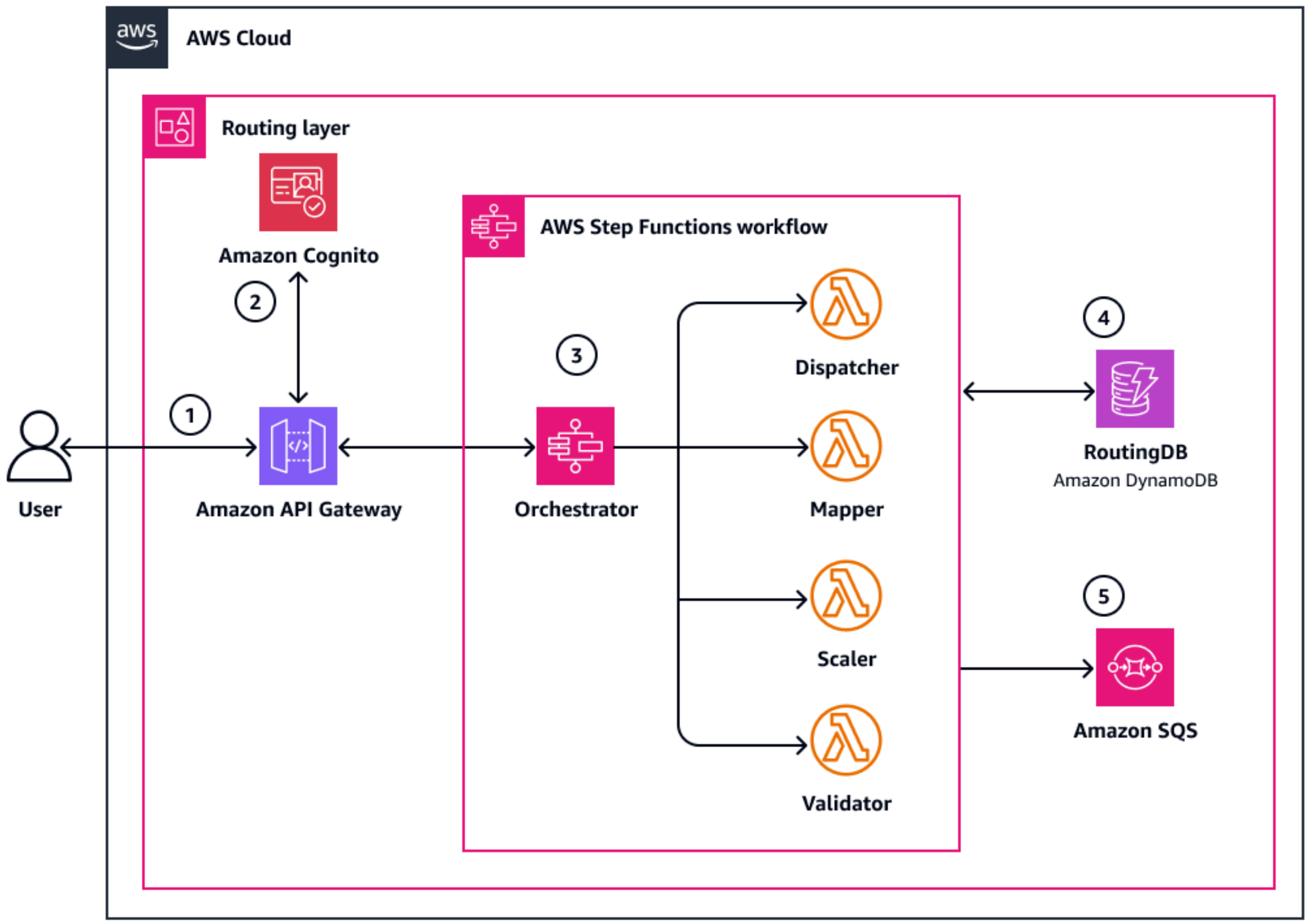
- The latest version of [AWS Command Line Interface (AWS CLI)]()
- [AWS credentials]() with the necessary permissions to create the AWS CloudFormation stack, AWS Lambda functions, and related resources

Product versions

- Python 3.12

Architecture

The following diagram shows a high-level design of the cell router.



The diagram steps through the following workflow:

1. The user contacts Amazon API Gateway, which serves as the front for the cell-router API endpoints.

2. Amazon Cognito handles the authentication and authorization.
3. The AWS Step Functions workflow consists of the following components:
   - **Orchestrator** – The `Orchestrator` uses AWS Step Functions to create a workflow, or state machine. The workflow is triggered by the cell router API. The `Orchestrator` executes Lambda functions based on the resource path.
   - **Dispatcher** – The `Dispatcher` Lambda function identifies and assigns one static cell per registered new user. The function searches for the cell with the least number of users, assigns it to the user, and returns the endpoints.
   - **Mapper** – The `Mapper` operation handles the user-to-cell mappings within the `RoutingDB` Amazon DynamoDB database that was created by the AWS CloudFormation template. When triggered, the `Mapper` function provides the already assigned users with their endpoints.
   - **Scaler** – The `Scaler` function keeps track of the cell occupancy and available capacity. When needed, the `Scaler` function can send a request through Amazon Simple Queue Service (Amazon SQS) to the Provision and Deploy layer to request new cells.
   - **Validator** – The `Validator` function validates the cell endpoints and detects any potential issues.
4. The `RoutingDB` stores cell information and attributes (API endpoints, AWS Region, state, metrics).
5. When the available capacity of cells exceeds a threshold, the cell router requests provisioning and deployment services through Amazon SQS to create new cells.

When new cells are created, `RoutingDB` gets updated from the Provision and Deploy layer. However, that process is beyond the scope of this pattern. For an overview of cell-based architecture design premises and details about the cell-router design used in this pattern, see the Additional information section.

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.

- Amazon Cognito provides authentication, authorization, and user management for web and mobile apps.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- AWS Step Functions is a serverless orchestration service that helps you combine Lambda functions and other AWS services to build business-critical applications.

Other tools

- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub Serverless-Cell-Router repository.

Best practices

For best practices when building cell-based architectures, see the following AWS Well-Architected guidance:

- Reducing the Scope of Impact with Cell-Based Architecture
- AWS Well-Architected Framework Reliability Pillar: REL10-BP04 Use bulkhead architectures to limit scope of impact

Epics

Prepare source files

Task	Description	Skills required
Clone the example code repository.	To clone the Serverless-Cell-Router GitHub repository to your computer, use the following command: `git clone https://github.com/aws-samples/Serverless-Cell-Router/`	Developer
Set up AWS CLI temporary credentials.	Configure the AWS CLI with credentials for your AWS account. This walkthrough uses temporary credentials provided by the AWS IAM Identity Center **Command line or programmatic access** option. This sets the `AWS_ACCESS_KEY_ID` , `AWS_SECRET_ACCESS_KEY` , and `AWS_SESSION_TOKEN` AWS environment variables with the appropriate credentials for use with the AWS CLI.	Developer
Create an S3 bucket.	Create an S3 bucket that will be used to store and access the Serverless-Cell-Router Lambda functions for deployment by the AWS CloudFormation template.	Developer

Task	Description	Skills required
	To create the S3 bucket, use following command: `aws s3api create-bucket --bucket <bucket name> --region eu-central-1 --create-bucket-configuration LocationConstraint=eu-central-1`	

Create .zip files.	Create one .zip file for each Lambda function located in the [Functions](#) directory. These .zip files will be used to deploy the Lambda functions . On a Mac, use the following `zip` commands:
	`zip -j mapper-scr.zip Functions/Mapper.py`
	`zip -j dispatcher-scr.zip Functions/Dispatcher.py`
	`zip -j scaler-scr.zip Functions/Scaler.py`
	`zip -j cp validator-scr.zip Functions/Validator.py`
	`zip -j dynamodbDummyData-scr.zip Functions/DynamodbDummyData.py`
		Developer

Task	Description	Skills required

Copy the .zip files to the S3 bucket.	To copy all the Lambda function .zip files to the S3 bucket, use the following commands:
	
	`aws s3 cp mapper-scr.zip s3://<bucket name>`
	`aws s3 cp dispatcher-scr.zip s3://<bucket name>`
	`aws s3 cp scaler-scr.zip s3://<bucket name>`
	`aws s3 cp validator-scr.zip s3://<bucket name>`
	`aws s3 cp dynamodbDummyData-scr.zip s3://<bucket name>`
		Developer

Create the AWS CloudFormation stack

Task	Description	Skills required

Deploy the AWS CloudFormation template.	To deploy the AWS CloudFormation template, run the following AWS CLI command:
	
	`aws cloudformation create-stack --stack-name serverless.cell-router \`
	`--template-body file://Serverless-Cell-Router-Stack-v10.yaml \`
		Developer

Task	Description	Skills required

```
--capabilities
 CAPABILITY_IAM \
--parameters Parameter
Key=LambdaFunction
MapperS3KeyParamet
erSCR,ParameterVal
ue=mapper-scr.zip \
ParameterKey=Lam
bdaFunctionDispatc
herS3KeyParameterS
CR,ParameterValue=
dispatcher-scr.zip \
ParameterKey=Lam
bdaFunctionScalerS
3KeyParameterSCR,P
arameterValue=scaler-
scr.zip \
ParameterKey=Lam
bdaFunctionAddDyna
moDBDummyItemsS3Ke
yParameterSCR,Para
meterValue=dynamod
bDummyData-scr.zip \
ParameterKey=Lam
bdaFunctionsS3Buck
etParameterSCR,Par
ameterValue=<S3 bucket
 storing lambda zip
 files> \
ParameterKey=Cog
nitoDomain,Paramet
erValue=<Cognito
 Domain Name> \
--region <enter your
 aws region id, e.g.
 "eu-central-1">
```

Task	Description	Skills required
Check progress.	Sign in to the AWS Management Console, open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/, and check the progress of stack development. When the status is `CREATE_COMPLETE`, the stack has been deployed successfully.	Developer

Assess and verify

Task	Description	Skills required

Assign cells to the user.	To initiate the `Orchestrator`, run the following curl command:
	`curl -X POST \`
	`-H "Authorization: Bearer {User id_token}" \`
	`https://xxxxxx.execute-api.eu-central-1.amazonaws.com/Cell_Router_Development/cells`
	
	The `Orchestrator` triggers the execution of the `Dispatcher` function. The `Dispatcher`, in turn, verifies the existence of the
		Developer

Task	Description	Skills required
	user. If the user is found, the `Dispatcher` returns the associated cell ID and endpoint URLs. If the user isn't found, the `Dispatcher` allocates a cell to the user and sends the cell ID to the `Scaler` function for assessment of the assigned cell's residual capacity. The `Scaler` function's response is the following: `"cellID : cell-0002 , endPoint_1 : https://xxxxx.execute-api.eu-north-1.amazonaws.com/ , endPoint_2 : https://xxxxxxx.execute-api.eu-central-1.amazonaws.com/"`	

Task	Description	Skills required

Retrieve user cells.	To use the `Orchestrator` to execute the `Mapper` function, run the following command:
	`curl -X POST \ -H "Authorization: Bearer {User id_token}" \ https://xxxxxxxxx.execute-api.eu-central-1.amazonaws.com/Cell_Router_Development/mapper`
	The `Orchestrator` searches for the cell assigned to the user and returns the cell ID and URLs in the following response:
	`"cellID : cell-0002 , endPoint_1 : https://xxxxx.execute-api.eu-north-1.amazonaws.com/ , endPoint_2 : https://xxxxxxx.execute-api.eu-central-1.amazonaws.com/"`
		Developer

Clean up

Task	Description	Skills required
Clean up the resources.	To avoid incurring additiona l charges in your account, do the following: 1. Empty the S3 bucket that you created for the Lambda functions. 2. Delete the bucket. 3. Delete the AWS CloudForm ation stack.	App developer

Related resources

References

- Static stability using Availability Zones
- AWS Fault Isolation Boundaries: Static stability

Video

Physalia: Cell-based Architecture to Provide Higher Availability on Amazon EBS

https://www.youtube-nocookie.com/embed/6IknqRZMFic?controls=0

Additional information

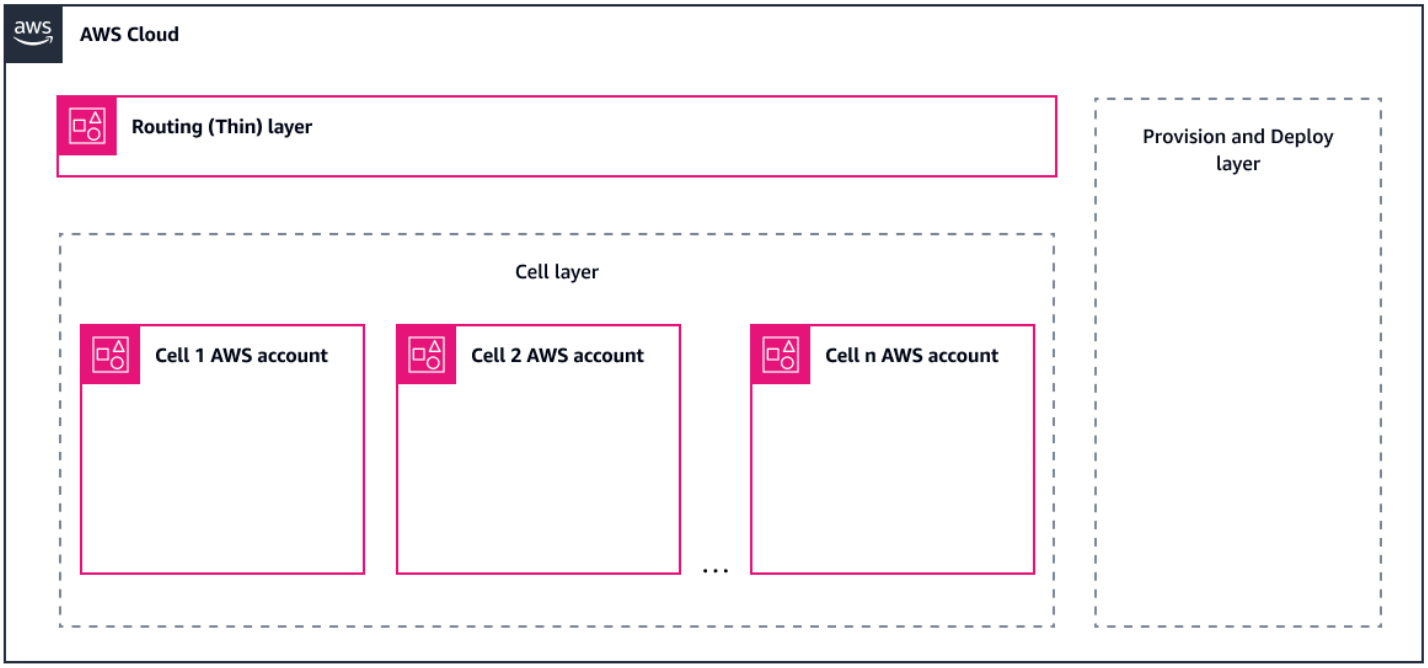
Cell-based architecture design premises

Although this pattern focuses on the cell router, it's important to understand the whole environment. The environment is structured into three discrete layers:

- The Routing layer, or Thin layer, which contains the cell router
- The Cell layer, comprising various cells
- The Provision and Deploy Layer, which provisions cells and deploys the application

Each layer sustains functionality even in the event of impairments affecting other layers. AWS accounts serve as a fault isolation boundary.

The following diagram shows the layers at a high level. The Cell layer and the Provision and Deploy layer are outside the scope of this pattern.



For more information about cell-based architecture, see [Reducing the Scope of Impact with Cell-Based Architecture: Cell routing](#).

Cell-router design pattern

The cell router is a shared component across cells. To mitigate potential impacts, it's important for the Routing layer to use a simplistic and horizontally scalable design that's as thin as possible. Serving as the system's entry point, the Routing layer consists of only the components that are required to efficiently assign users to the appropriate cells. Components within this layer don't engage in the management or creation of cells.

This pattern uses static routing, which means that the client caches the endpoints at the initial login and subsequently establishes direct communication with the cell. Periodic interactions between the client and the cell router are initiated to confirm the current status or retrieve any updates. This intentional decoupling enables uninterrupted operations for existing users in the event of cell-router downtime, and it provides continued functionality and resilience within the system.

In this pattern, the cell router supports the following functionalities:

- Retrieving cell data from the cell database in the Provision and Deploy layer and storing or updating the local database.
- Assigning a cell to each new registered user of the application by using the cell assignment algorithm.
- Storing the user-to-cells mapping in the local database.
- Checking the capacity of the cells during user assignment and raising an event for the vending machine to the Provision and Deploy layer to create cells.
- Using the cell creation criteria algorithm to provide this functionality.
- Responding to the newly registered user requests by providing the URLs of the static cells. These URLs will be cached on the client with a time to live (TTL).
- Responding to the existing user requests of an invalid URL by providing a new or updated URL.

To further understand the demonstration cell router that is set up by the AWS CloudFormation template, review the following components and steps:

1. Set up and configure the Amazon Cognito user pool.
2. Set up and configure the API Gateway API for the cell router.
3. Create a DynamoDB table.
4. Create and configure an SQS queue.
5. Implement the `Orchestrator`.
6. Implement the Lambda functions: `Dispatcher`, `Scaler`, `Mapper`, `Validator`.
7. Asses and verify.

The presupposition is that the Provision and Deploy layer is already established. Its implementation details fall beyond the scope of this artifact.

Because these components are set up and configured by an AWS CloudFormation template, the following steps are presented at a descriptive and high level. The assumption is that you have the required AWS skills to complete the setup and configuration.

1. Setup and configure the Amazon Cognito user pool

Sign in to the AWS Management Console, and open the Amazon Cognito console at https://console.aws.amazon.com/cognito/. Set up and configure an Amazon Cognito user pool named `CellRouterPool`, with app integration, hosted UI, and the necessary permissions.

2. Set up and configure the API Gateway API for the cell router

Open the API Gateway console at https://console.aws.amazon.com/apigateway/. Set up and configure an API named `CellRouter`, using an Amazon Cognito authorizer integrated with the Amazon Cognito user pool `CellRouterPool`. Implement the following elements:

- `CellRouter` API resources, including `POST` methods
- Integration with the Step Functions workflow implemented in step 5
- Authorization through the Amazon Cognito authorizer
- Integration request and response mappings
- Allocation of necessary permissions

3. Create a DynamoDB table

Open the DynamoDB console at https://console.aws.amazon.com/dynamodb/, and create a standard DynamoDB table called `tbl_router` with the following configuration:

- **Partition key** – `marketId`
- **Sort key** – `cellId`
- **Capacity mode** – Provisioned
- **Point-in-time recovery (PITR)** – Off

On the **Indexes** tab, create a global secondary index called `marketId-currentCapacity-index`. The `Scaler` Lambda function will use the index to conduct efficient searches for the cell with the lowest number of assigned users.

Create the table structure with the following attributes:

- `marketId` – Europe
- `cellId` – cell-0002
- `currentCapacity` – 2
- `endPoint_1` – <your endpoint for the first Region>
- `endPoint_2` – <your endpoint for the second Region>
- `IsHealthy` – True

- `maxCapacity` – 10
- `regionCode_1` – `eu-north-1`
- `regionCode_2` – `eu-central-1`
- `userIds` – <your email address>

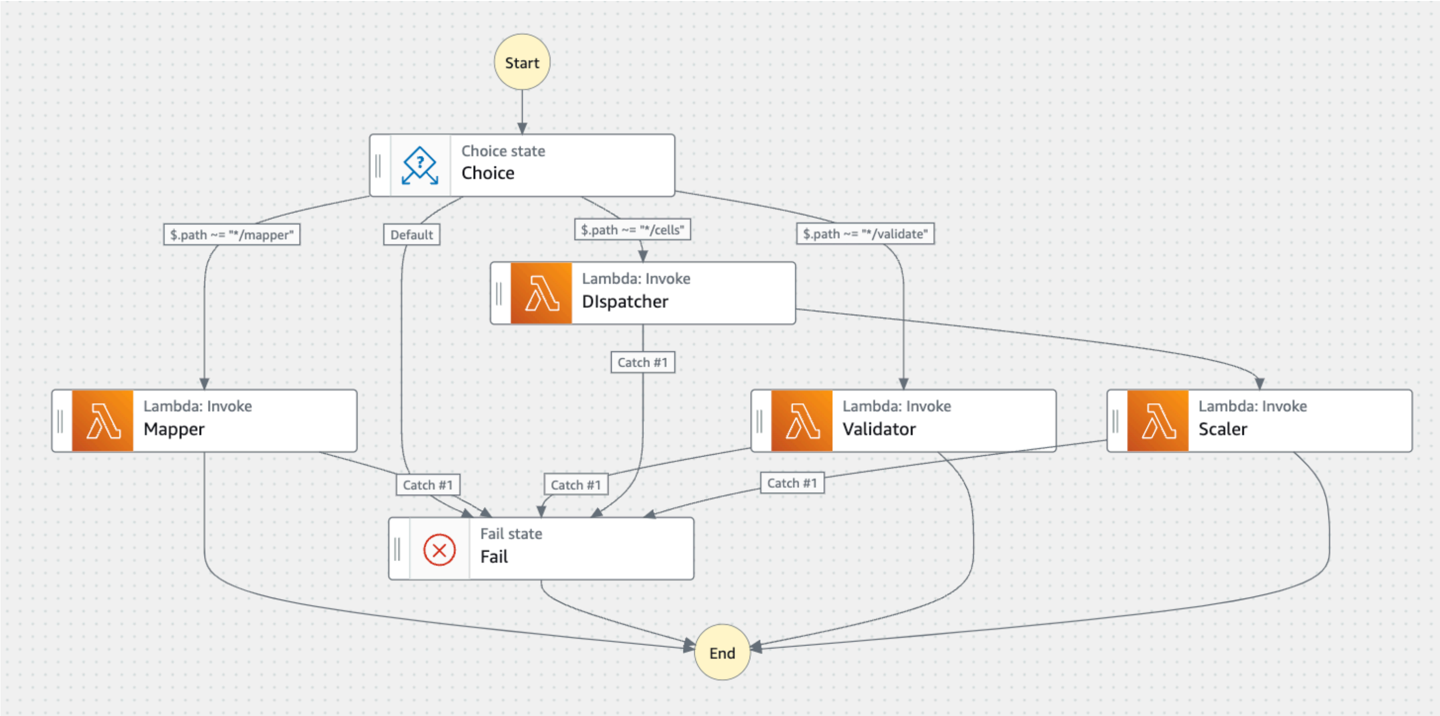
4. Create and configure an SQS queue

Open the Amazon SQS console at https://console.aws.amazon.com/sqs/, and create a standard SQS queue called `CellProvisioning` configured with **Amazon SQS key** encryption.

5. Implement the Orchestrator

Develop a Step Functions workflow to serve as the `Orchestrator` for the router. The workflow is callable through the cell router API. The workflow executes the designated Lambda functions based on the resource path. Integrate the step function with the API Gateway API for the cell router `CellRouter`, and configure the necessary permissions to invoke the Lambda functions.

The following diagram shows the workflow. The choice state invokes one of the Lambda functions. If the Lambda function is successful, the workflow ends. If the Lambda function fails, fail state is called.



6. Implement the Lambda functions

Implement the `Dispatcher`, `Mapper`, `Scaler`, and `Validator` functions. When you set up and configure each function in the demonstration, define a role for the function and assign the necessary permissions for performing required operations on the DynamoDB table `tbl_router`. Additionally, integrate each function into the workflow `Orchestrator`.

Dispatcher function

The `Dispatcher` function is responsible for identifying and assigning a single static cell for each new registered user. When a new user registers with the global application, the request goes to the `Dispatcher` function. The function processes the request by using predefined evaluation criteria such as the following:

1. **Region** – Select the cell in the market where the user is located. For example, if the user is accessing the global application from Europe, select a cell that uses AWS Regions in Europe.
2. **Proximity or latency** – Select the cell closest to the user For example, if the user is accessing the application from Holland, the function considers a cell that uses Frankfurt and Ireland. The decision regarding which cell is closest is based on metrics such as latency between the user's location and the cell Regions. For this example pattern, the information is statically fed from the Provision and Deploy layer.
3. **Health** – The `Dispatcher` function checks whether the selected cell is healthy based on the provided cell state (Healthy = true or false).
4. **Capacity** – The user distribution is based on *least number of users in a cell* logic, so the user is assigned to the cell that has least number of users.

Note

These criteria are presented to explain this example pattern only. For a real-life cell-router implementation, you can define more refined and use case–based criteria.

The `Orchestrator` invokes the Dispatcher function to assign users to cells. In this demo function, the market value is a static parameter defined as `europe`.

The `Dispatcher` function assesses whether a cell is already assigned to the user. If the cell is already assigned, the `Dispatcher` function returns the cell's endpoints. If no cell is assigned to the user, the function searches for the cell with the least number of users, assigns it to the user, and returns the endpoints. The efficiency of the cell search query is optimized by using the global secondary index.

Mapper function

The `Mapper` function oversees the storage and maintenance of user-to-cell mappings in the database. A singular cell is allocated to each registered user. Each cell has two distinct URLs—one for each AWS Region. Serving as API endpoints hosted on API Gateway, these URLs function as inbound points to the global application.

When the `Mapper` function receives a request from the client application, it runs a query on the DynamoDB table `tbl_router` to retrieve the user-to-cell mapping that is associated with the provided email ID. If it finds an assigned cell, the `Mapper` function promptly provides the cell's two URLs. The `Mapper` function also actively monitors alterations to the cell URLs, and it initiates notifications or updates to user settings.

Scaler function

The `Scaler` function manages the residual capacity of the cell. For each new user-registration request, the `Scaler` function assesses the available capacity of the cell that the `Dispatcher` function assigned to the user. If the cell has reached its predetermined limit according to the specified evaluation criteria, the function initiates a request through an Amazon SQS queue to the Provision and Deploy layer, soliciting the provisioning and deployment of new cells. The scaling of cells can be executed based on a set of evaluation criteria such as the following:

1. **Maximum users** – Each cell can have 500 maximum number of users.
2. **Buffer capacity** – The buffer capacity of each cell is 20 percent, which means that each cell can be assigned to 400 users at any time. The remaining 20 percent buffer capacity is reserved for future use cases and handling of unexpected scenarios (for example, when cell creation and provisioning services are unavailable).
3. **Cell creation** – As soon as an existing cell reaches 70 percent of capacity, a request is triggered to create an additional cell.

> **Note**
>
> These criteria are presented to explain this example pattern only. For a real-life cell-router implementation, you can define more refined and use case–based criteria.

The demonstration `Scaler` code is executed by the `Orchestrator` after the `Dispatcher` successfully assigns a cell to the newly registered user. The `Scaler`, upon receipt of the cell ID from the `Dispatcher`, evaluates whether the designated cell has adequate capacity to accommodate additional users, based on predefined evaluation criteria. If the cell's capacity is insufficient, the `Scaler` function dispatches a message to the Amazon SQS service. This message is retrieved by the service within the Provision and Deploy layer, initiating the provisioning of a new cell.

Validator function

The `Validator` function identifies and resolves issues pertaining to cell access. When a user signs in to the global application, the application retrieves the cell's URLs from the user profile settings and routes user requests to one of the two assigned Regions within the cell. If the URLs are inaccessible, the application can dispatch a validate URL request to the cell router. The cell-router `Orchestrator` invokes the `Validator`. The `Validator` initiates the validation process. Validation might include, among other checks, the following:

- Cross-referencing cell URLs in the request with URLs stored in database to identify and process potential updates
- Running a deep health check (for example, an `HTTP GET` request to the cell's endpoint)

In conclusion, the `Validator` function delivers responses to client application requests, furnishing validation status along with any required remediation steps.

The `Validator` is designed to enhance user experience. Consider a scenario where certain users encounter difficulty accessing the global application because an incident causes cells to be temporarily unavailable. Instead of presenting generic errors, the `Validator` function can provide instructive remediation steps. These steps might include the following actions:

- Inform users about the incident.
- Provide an approximate wait time before service availability.
- Provide a support contact number for obtaining additional information.

The demo code for the `Validator` function verifies that the user-supplied cell URLs in the request match the records stored in the `tbl_router` table. The `Validator` function also checks whether the cells are healthy.

Set up private access to an Amazon S3 bucket through a VPC endpoint

Martin Maritsch, Nicolas Jacob Baer, Gabriel Rodriguez Garcia, Shukhrat Khodjaev, Mohan Gowda Purushothama, and Joaquin Rinaudo, Amazon Web Services

Summary

In Amazon Simple Storage Service (Amazon S3), presigned URLs enable you to share files of arbitrary size with target users. By default, Amazon S3 presigned URLs are accessible from the internet within an expiration time window, which makes them convenient to use. However, corporate environments often require access to Amazon S3 presigned URLs to be limited to a private network only.

This pattern presents a serverless solution for securely interacting with S3 objects by using presigned URLs from a private network without internet traversal. In the architecture, users access an Application Load Balancer through an internal domain name. Traffic is routed internally through Amazon API Gateway and a virtual private cloud (VPC) endpoint for the S3 bucket. The AWS Lambda function generates presigned URLs for file downloads through the private VPC endpoint, which helps enhance security and privacy for sensitive data.

Prerequisites and limitations

Prerequisites

- A VPC that includes a subnet deployed in an AWS account that is connected to the corporate network (for example, through AWS Direct Connect).

Limitations

- The S3 bucket must have the same name as the domain, so we recommend that you check Amazon S3 bucket naming rules.
- This sample architecture doesn't include monitoring features for the deployed infrastructure. If your use case requires monitoring, consider adding AWS monitoring services.
- This sample architecture doesn't include input validation. If your use case requires input validation and an increased level of security, consider using AWS WAF to protect your API.
- This sample architecture doesn't include access logging with the Application Load Balancer. If your use case requires access logging, consider enabling load balancer access logs.

Versions

- Python version 3.11 or later
- Terraform version 1.6 or later

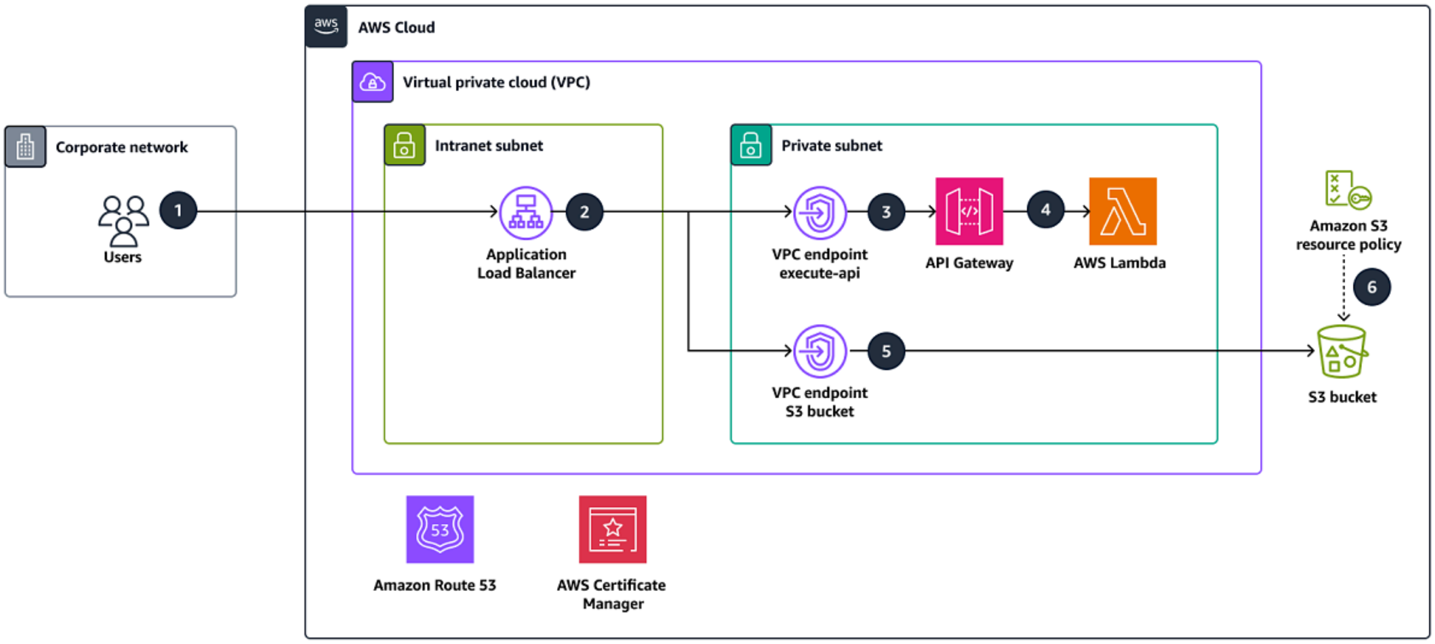
Architecture

Target technology stack

The following AWS services are used in the target technology stack:

- **Amazon S3** is the core storage service used for uploading, downloading, and storing files securely.
- **Amazon API Gateway** exposes resources and endpoints for interacting with the S3 bucket. This service plays a role in generating presigned URLs for downloading or uploading data.
- **AWS Lambda** generates presigned URLs for downloading files from Amazon S3. The Lambda function is called by API Gateway.
- **Amazon VPC** deploys resources within a VPC to provide network isolation. The VPC includes subnets and routing tables to control traffic flow.
- **Application Load Balancer** routes incoming traffic either to API Gateway or to the VPC endpoint of the S3 bucket. It allows users from the corporate network to access resources internally.
- **VPC endpoint for Amazon S3** enables direct, private communication between resources in the VPC and Amazon S3 without traversing the public internet.
- **AWS Identity and Access Management (IAM)** controls access to AWS resources. Permissions are set up to ensure secure interactions with the API and other services.

Target architecture



The diagram illustrates the following:

1. Users from the corporate network can access the Application Load Balancer through an internal domain name. We assume that a connection exists between the corporate network and the intranet subnet in the AWS account (for example, through a AWS Direct Connect connection).
2. The Application Load Balancer routes incoming traffic either to API Gateway to generate presigned URLs to download or upload data to Amazon S3, or to the VPC endpoint of the S3 bucket. In both scenarios, requests are routed internally and do not need to traverse the internet.
3. API Gateway exposes resources and endpoints to interact with the S3 bucket. In this example, we provide an endpoint to download files from the S3 bucket, but this could be extended to provide upload functionality as well.
4. The Lambda function generates the presigned URL to download a file from Amazon S3 by using the domain name of the Application Load Balancer instead of the public Amazon S3 domain.
5. The user receives the presigned URL and uses it to download the file from Amazon S3 by using the Application Load Balancer. The load balancer includes a default route to send traffic that's not intended for the API toward the VPC endpoint of the S3 bucket.
6. The VPC endpoint routes the presigned URL with the custom domain name to the S3 bucket. The S3 bucket must have the same name as the domain.

Automation and scale

This pattern uses Terraform to deploy the infrastructure from the code repository into an AWS account.

Tools

Tools

- Python is a general-purpose computer programming language.
- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.

Code repository

The code for this pattern is available in a GitHub repository at https://github.com/aws-samples/private-s3-vpce.

Best practices

The sample architecture for this pattern uses IAM permissions to control access to the API. Anyone who has valid IAM credentials can call the API. If your use case requires a more complex authorization model, you might want to use a different access control mechanism.

Epics

Deploy the solution in an AWS account

Task	Description	Skills required
Obtain AWS credentials.	Review your AWS credentials and your access to your account. For instructions, see Configuration and credential file settings in the AWS CLI documentation.	AWS DevOps, General AWS
Clone the repository.	Clone the GitHub repository provided with this pattern:	AWS DevOps, General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre>git clone https://g
ithub.com/aws-samp
les/private-s3-vpce</pre></td><td></td></tr>
<tr><td>Configure variables.</td><td>1. On your computer, in the GitHub repository, open the <code>terraform</code> folder:<pre>cd terraform</pre>2. Open the <code>example.tfvars</code> file and customize the parameters according to your needs.</td><td>AWS DevOps, General AWS</td></tr>
<tr><td>Deploy solution.</td><td>1. In the <code>terraform</code> folder, run Terraform and pass in the variables that you customized:<pre>terraform apply -
var-file="example
.tfvars"</pre>2. Confirm that the resources shown in the architecture diagram were deployed successfully.</td><td>AWS DevOps, General AWS</td></tr>
</table>

Test the solution

Task	Description	Skills required
Create a test file.	Upload a file to Amazon S3 to create a test scenario for the file download. You can use	AWS DevOps, General AWS

Task	Description	Skills required
	the [Amazon S3 console](https://console.aws.amazon.com/s3/) or the following AWS CLI command: `aws s3 cp /path/to/testfile s3://your-bucket-name/testfile`	
Test presigned URL functionality.	1. Send a request to the Application Load Balancer to create a presigned URL for the test file by using awscurl: `awscurl https://your-domain-name/api/get_url?key=testfile` This step creates a valid signature from your credentials, which will be validated by API Gateway. 2. Parse the link from the response you receive from the previous step, and open the presigned URL to download the file.	AWS DevOps, General AWS
Clean up.	Make sure to remove the resources when they are no longer required: `terraform destroy`	AWS DevOps, General AWS

Troubleshooting

Issue	Solution
S3 object key names with special characters such as number signs (#) break URL parameter s and lead to errors.	Encode URL parameters properly, and make sure that the S3 object key name follows Amazon S3 guidelines.

Related resources

Amazon S3:

- Sharing objects with presigned URLs
- Controlling access from VPC endpoints with bucket policies

Amazon API Gateway:

- Use VPC endpoint policies for private APIs in API Gateway

Application Load Balancer:

- Hosting Internal HTTPS Static Websites with ALB, S3, and PrivateLink (AWS blog post)

Troubleshoot states in AWS Step Functions by using Amazon Bedrock

Aniket Kurzadkar and Sangam Kushwaha, Amazon Web Services

Summary

AWS Step Functions error handling capabilities can help you see an error that occurs during a state in a workflow, but it can still be a challenge to find the root cause of an error and debug it. This pattern addresses that challenge and shows how Amazon Bedrock can help you resolve errors that occur during states in Step Functions.

Step Functions provides workflow orchestration, making it easier for developers to automate processes. Step Functions also provides error handling functionality that provides the following benefits:

- Developers can create more resilient applications that don't fail completely when something goes wrong.
- Workflows can include conditional logic to handle different types of errors differently.
- The system can automatically retry failed operations, perhaps with exponential backoff.
- Alternative execution paths can be defined for error scenarios, allowing the workflow to adapt and continue processing.

When an error occurs in a Step Functions workflow, this pattern shows how the error message and context can be sent to a foundation model (FM) like Claude 3 that's supported by Step Functions. The FM can analyze the error, categorize it, and suggest potential remediation steps.

Prerequisites and limitations

Prerequisites

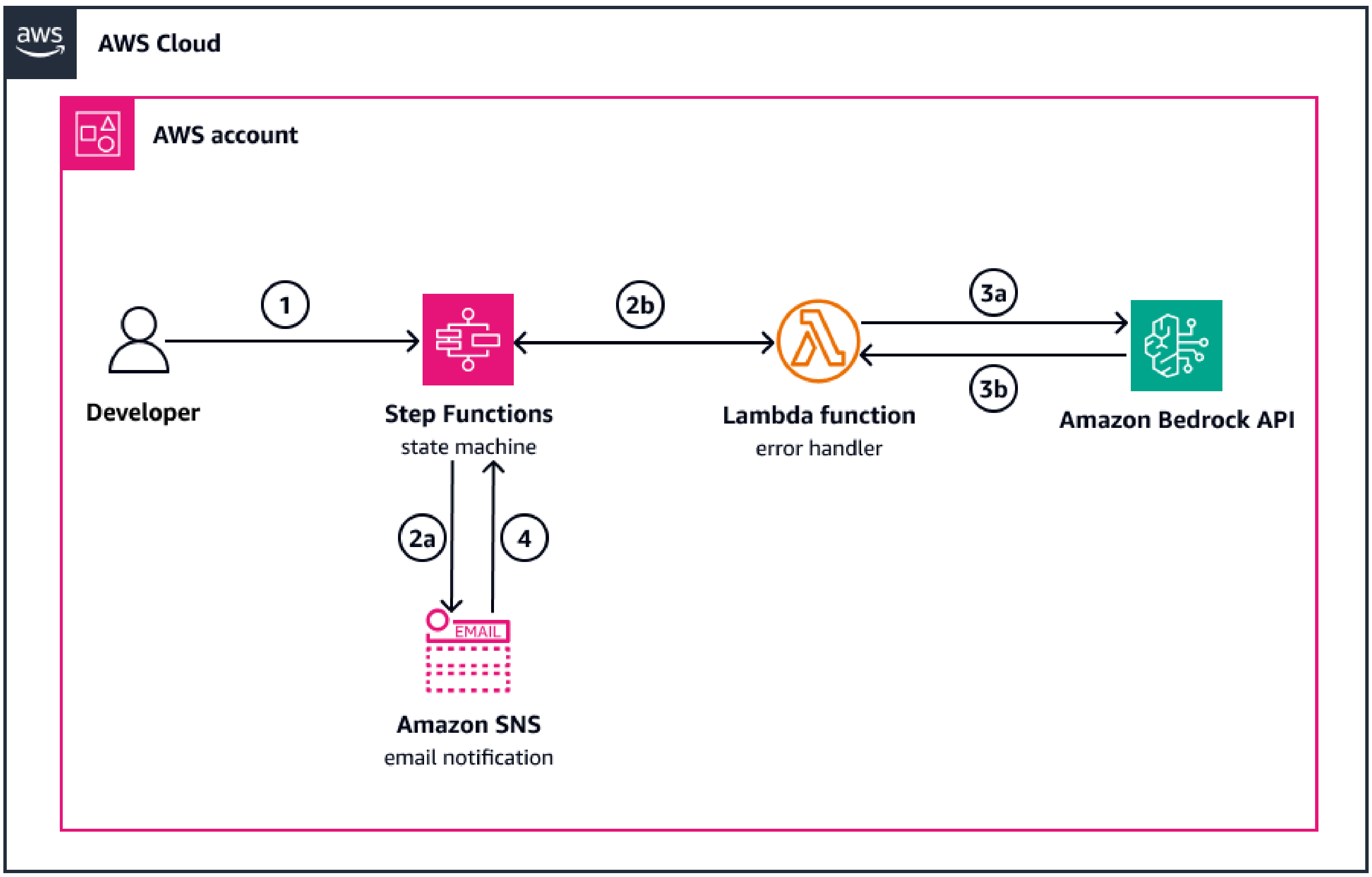
- An active AWS account
- Basic understanding of AWS Step Functions and workflows
- Amazon Bedrock API connectivity

Limitations

- You can use this pattern's approach for various AWS services. However, the results might vary according to the prompt created by AWS Lambda that's subsequently evaluated by Amazon Bedrock.
- Some AWS services aren't available in all AWS Regions. For Region availability, see [AWS services by Region](). For specific endpoints, see [Service endpoints and quotas](), and choose the link for the service.

Architecture

The following diagram shows the workflow and architecture components for this pattern.



The diagram shows the automated workflow for error handling and notification in a Step Functions state machine:

1. The developer starts a state machine's execution.
2. The Step Functions state machine begins processing its states. There are two possible outcomes:

  - (a) If all states execute successfully, the workflow proceeds directly to Amazon SNS for an email success notification.
  - (b) If any state fails, the workflow moves to the error handling Lambda function.

3. In case of an error, the following occurs:
  - (a) The Lambda function (error handler) is triggered. The Lambda function extracts the error message from the event data that the Step Functions state machine passed to it. Then the Lambda function prepares a prompt based on this error message and sends the prompt to Amazon Bedrock. The prompt requests solutions and suggestions related to the specific error encountered.
  - (b) Amazon Bedrock, which hosts the generative AI model, processes the input prompt. (This pattern uses the Anthropic Claude 3 foundation model (FM), which is one of many FMs that Amazon Bedrock supports.) The AI model analyses the error context. Then the model generates a response that can include explanations of why the error occurred, potential solutions to resolve the error, and suggestions to avoid making the same mistakes in the future.

    Amazon Bedrock returns its AI-generated response to the Lambda function. The Lambda function processes the response, potentially formatting it or extracting key information. Then the Lambda function sends the response to the state machine output.
4. After error handling or successful execution, the workflow concludes by triggering Amazon SNS to send an email notification.

Tools

AWS services

- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

- [AWS Step Functions](#) is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

Best practices

- Given that Amazon Bedrock is a generative AI model that learns from trained data, it also uses that data to train and generate context. As a best practice, conceal any private information that might lead to data leak problems.
- Although generative AI can provide valuable insights, critical error-handling decisions should still involve human oversight, especially in production environments.

Epics

Create a state machine for your workflow

Task	Description	Skills required
Create a state machine.	To create a state machine that's appropriate for your workflow, do the following: 1. Sign in to the AWS Management Console, and open the AWS Step Functions [console](#). 2. From the left navigatio n pane, choose **State machines**. 3. Choose **Create state machine**. 4. Choose a template according to your use case, or choose **Blank** to create a template according to your requirements.	AWS DevOps

Create a Lambda function

Task	Description	Skills required
Create a Lambda function.	To create a Lambda function, do the following: 1. In the AWS Management Console, navigate to the AWS Lambda console. 2. In the left navigation pane, choose **Functions** and then choose **Create function**. 3. On the **Create function** page, choose from the options to create a function. Then, enter a name in **Function name** and choose the appropria te language from the dropdown list in **Runtime**. 4. Choose **Create function**.	AWS DevOps
Set up the required logic in the Lambda code.	• To connect to the Amazon Bedrock API by using the [AWS SDK for Python (Boto3)](), use the following code. This code sets up a client for Amazon Bedrock, prepares the necessary parameters, and then sends a request to the Claude 3 model with a specified prompt.	AWS DevOps

Task	Description	Skills required

	This pattern invokes the Claude 3 model. For more information about all the supported foundation models including related model IDs, see [Supported foundation models in Amazon Bedrock](#) in the Amazon Bedrock documentation.
	
	``
	client = boto3.client(
	        service_name="bedrock-runtime", region_name="selected-region"
	    )
	
	    # Invoke Claude 3 with the text prompt
	    model_id = "your-model-id" # Select your Model ID, Based on the Model Id, Change the body format
	
	    try:
	        response = client.invoke_model(
	            modelId=model_id,
	            body=json.dumps(
	                {
	                    "anthropic_version": "bedrock-2023-05-31",
	``	

Task	Description	Skills required

	``
	"max_tokens": 1024,
	
	"messages": [
	
	{
	
	   "role": "user",
	
	         "content"
	: [{"type": "text",
	 "text": prompt}],
	
	}
	                  ],
	              }
	          ),
	      )
	``
	
	• (Optional) Replace the AWS account IDs with placeholder account IDs. For security purposes, this approach can be useful for sanitizing logs, error messages, or other output that might contain sensitive account information.
	
	The following code will find any occurrence of a 12-digit number enclosed in colons (which is the format of AWS account IDs in Amazon Resource Names (ARNs) and some other AWS identifiers) and replace it with the	

Task	Description	Skills required

	placeholder account ID `":123456789012:"` .
	`def replace_account_id(input_string):`
	`    # Use a regular expression to find the AWS account ID pattern`
	`    account_id_pattern = r'(:\d{12}:)'`
	`    # Replace the matched pattern with ":123456789012:"`
	`    modified_string = re.sub(account_id_pattern, ":123456789012:", input_string)`
	`    return modified_string`	

Integrate Step Functions with Lambda

Task	Description	Skills required
Set up Lambda to handle errors in Step Functions.	To set up Step Functions to handle errors without disrupting the workflow, do the following: 1. In the Step Functions console, navigate to the	AWS DevOps

Task	Description	Skills required
	state machine that you created earlier. 2. Choose **Edit**, and then choose the service that you want to set up error handling for and choose **Error Handling**. 3. Choose **Add new catcher**, and for **Fallback state**, choose **Lambda** and then choose the Lambda function that you created earlier. For more information, see Catch errors in the Step Functions documentation.	

Troubleshooting

Issue	Solution
Lambda cannot access the Amazon Bedrock API (Not authorized to perform)	This error occurs when the Lambda role doesn't have permission to access the Amazon Bedrock API. To resolve this issue, add the `AmazonBedrockFullAccess` policy for the Lambda role. For more information, see AmazonBedrockFullAccess in the *AWS Managed Policy Reference Guide*.
Lambda timeout error	Sometimes it might take more than 30 seconds to generate a response and send it back, depending on the prompt. To resolve this issue, increase the configuration time. For more information, see Configure

Issue	Solution
	Lambda function timeout in the *AWS Lambda Developer Guide*.

Related resources

- Amazon Bedrock
- Amazon Bedrock API access
- Create your first Lambda function
- Developing workflows with Step Functions
- AWS Step Functions

Chain AWS services together using a serverless approach

Aniket Braganza, Amazon Web Services

Summary

This pattern demonstrates a scalable, serverless approach for processing an uploaded file by chaining together Amazon Simple Storage Service (Amazon S3), Amazon Simple Notification Service (Amazon SNS), Amazon Simple Queue Service (Amazon SQS), and AWS Lambda. The uploaded file example is for demonstration purposes. You can use a serverless approach to complete other tasks by chaining together the combination of AWS services that are necessary to meet your business goals. The serverless approach employs an asynchronous workflow that relies on event-driven notifications, resilient storage, and function as a service (FaaS) computing to process requests. You can use the serverless approach to scale to meet demand while minimizing costs.

Note

There are several options for chaining AWS services together through a serverless approach. For example, you can use an approach that combines Lambda with Amazon S3 instead of Amazon SNS and Amazon SQS. However, this pattern uses Amazon SNS and Amazon SQS because this approach makes it possible to add multiple integration points into the Lambda invocation process during an event notification and to extend the implementation to include multiple listeners in a serverless orchestration while minimizing the amount of processing overhead.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Programmatic access to the AWS account. For more information, see:
  - Prerequisites in the AWS Cloud Development Kit (AWS CDK) documentation
  - Prerequisites in the AWS Command Line Interface (AWS CLI) documentation
- AWS CDK, installed
- AWS CLI, installed and configured

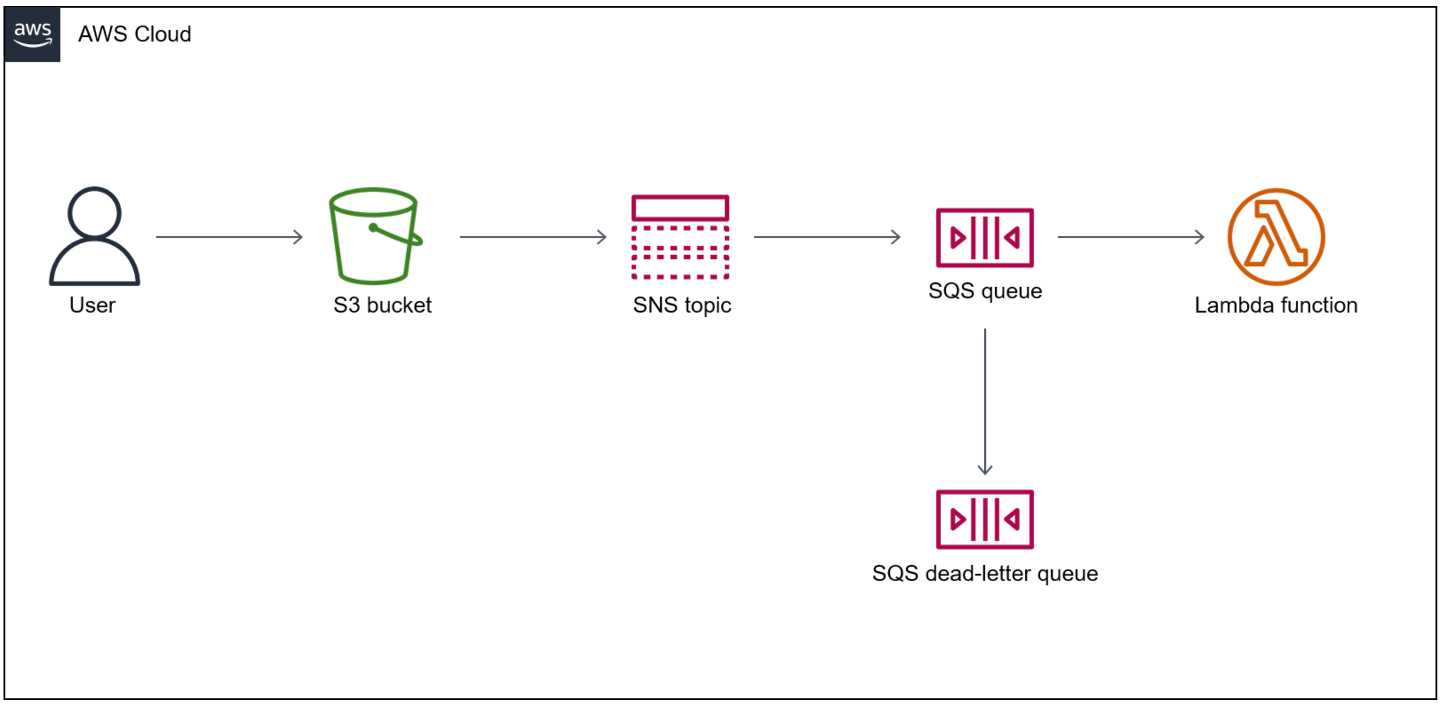
- [Python 3.9]

Product versions

- AWS CDK 2.x
- Python 3.9

Architecture

The following diagram illustrates how chained AWS services can enable a user to upload a file to an S3 bucket for processing.



The diagram shows the following workflow:

1. A user uploads a file to the S3 bucket.
2. The upload initiates an S3 event that publishes a message to an SNS topic. The message contains the details of the S3 event.
3. The message published to the SNS topic is inserted into an SQS queue, which is subscribed to and receives notifications for that topic.
4. A Lambda function polls the SQS queue (as its event source) and waits for messages to process.

5. When the Lambda function receives messages from the SQS queue, it processes them and acknowledges receipt of those messages.
6. If a message isn't processed by Lambda, then that message is returned to the SQS queue and is eventually transferred to an SQS dead-letter queue.

Technology stack

- Amazon S3
- Amazon SNS
- Amazon SQS
- AWS Lambda

Tools

AWS services

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Other tools

- AWS Cloud Development Kit (AWS CDK) is the primary tool for interacting with your AWS CDK app. It executes your app, interrogates the application model you defined, and produces and deploys the AWS CloudFormation templates generated by the AWS CDK.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Python is a high-level, interpreted general purpose programming language.

Code

The code for this pattern is available in the GitHub [Chaining S3 to SNS to SQS to Lambda]() repository.

Epics

Develop your serverless environment

Task	Description	Skills required
Clone the repository.	Clone the [repository]() and navigate to the `python/s3-sns-sqs-lambda-chain` folder.	App developer
Set up a virtual environment.	1. In the AWS CDK, run the `python3 -m venv .venv` command. 2. Run the `source .venv/bin/activate` command on MacOS/Linux or `.venv\Scripts\activate.bat` on Windows.	App developer
Install dependencies.	Run the `pip install -r requirements.txt` command.	App developer

Test the CloudFormation stack

Task	Description	Skills required
Run unit tests.	1. Run the `pip install -r requirements-dev.txt` command.	App developer, Test engineer

Task	Description	Skills required
	2. **Important** (Optional) Run the `cdk synth --no-staging > template.yml` command to generate the CloudFormation stack. You can inspect the stack, but avoid generating the staged resources and artifacts. 3. Run the `pytest` command to run all unit tests. 4. (Optional) Run the `pytest tests/unit/<test_filename>` command to run tests for a specific file.	

Deploy the CloudFormation stack

Task	Description	Skills required
Set up the bootstrap environment.	Follow the instructions in [Bootstrapping](#) in the AWS documentation to bootstrap the environment for AWS CDK deployment in each AWS Region where the CloudFormation stack will be deployed.	App developer, DevOps engineer, Data engineer

Task	Description	Skills required
	Note This step requires that you have credentials with programmatic access.	
Deploy the CloudFormation stack.	Run the `cdk deploy` command to build and deploy the stack to the AWS account.	App developer, DevOps engineer, AWS DevOps

Clean up your environment's resources

Task	Description	Skills required
Delete the CloudFormation stack and remove associated resources.	To delete the CloudForm ation stack that was created and remove all associated resources, run the `run cdk` destroy command.	App developer

More patterns

- Access, query, and join Amazon DynamoDB tables using Athena
- Aggregate data in Amazon DynamoDB for ML forecasting in Athena
- Automate Amazon CodeGuru reviews for AWS CDK Python applications by using GitHub Actions
- Automate AWS resource assessment
- Automate deployment of nested applications using AWS SAM
- Automate the replication of Amazon RDS instances across AWS accounts
- Automatically archive items to Amazon S3 using DynamoDB TTL
- Automatically detect changes and initiate different CodePipeline pipelines for a monorepo in CodeCommit
- Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9
- Build a multi-tenant serverless architecture in Amazon OpenSearch Service
- Build an advanced mainframe file viewer in the AWS Cloud
- Calculate value at risk (VaR) by using AWS services
- Copy AWS Service Catalog products across different AWS accounts and AWS Regions
- Create dynamic CI pipelines for Java and Python projects automatically
- Decompose monoliths into microservices by using CQRS and event sourcing
- Deploy a React-based single-page application to Amazon S3 and CloudFront
- Deploy an Amazon API Gateway API on an internal website using private endpoints and an Application Load Balancer
- Deploy and debug Amazon EKS clusters
- Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code
- Deploy a RAG use case on AWS by using Terraform and Amazon Bedrock
- Develop a fully automated chat-based assistant by using Amazon Bedrock agents and knowledge bases
- Develop advanced generative AI chat-based assistants by using RAG and ReAct prompting
- Dynamically generate an IAM policy with IAM Access Analyzer by using Step Functions
- Ensure Amazon EMR logging to Amazon S3 is enabled at launch
- Estimate the cost of a DynamoDB table for on-demand capacity
- Generate personalized and re-ranked recommendations using Amazon Personalize

- Generate test data using an AWS Glue job and Python
- Implement SHA1 hashing for PII data when migrating from SQL Server to PostgreSQL
- Implement the serverless saga pattern by using AWS Step Functions
- Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK
- Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function
- Migrate Apache Cassandra workloads to Amazon Keyspaces by using AWS Glue
- Monitor use of a shared Amazon Machine Image across multiple AWS accounts
- Optimize multi-account serverless deployments by using the AWS CDK and GitHub Actions workflows
- Orchestrate an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions
- Query Amazon DynamoDB tables with SQL by using Amazon Athena
- Run event-driven and scheduled workloads at scale with AWS Fargate
- Send custom attributes to Amazon Cognito and inject them into tokens
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront
- Streamline Amazon Lex bot development and deployment by using an automated workflow
- Structure a Python project in hexagonal architecture using AWS Lambda
- Translate natural language into query DSL for OpenSearch and Elasticsearch queries
- Turn off security standard controls across all Security Hub member accounts in a multi-account environment
- Unload data from an Amazon Redshift cluster across accounts to Amazon S3
- Coordinate resource dependency and task execution by using the AWS Fargate WaitCondition hook construct
- Use Amazon Bedrock agents to automate creation of access entry controls in Amazon EKS through text-based prompts

Networking

Topics

- Automate the setup of inter-Region peering with AWS Transit Gateway
- Centralize network connectivity using AWS Transit Gateway
- Configure HTTPS encryption for Oracle JD Edwards EnterpriseOne on Oracle WebLogic by using an Application Load Balancer
- Connect to Application Migration Service data and control planes over a private network
- Create Infoblox objects using AWS CloudFormation custom resources and Amazon SNS
- Customize Amazon CloudWatch alerts for AWS Network Firewall
- Deploy resources in an AWS Wavelength Zone by using Terraform
- Migrate DNS records in bulk to an Amazon Route 53 private hosted zone
- Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS
- Create a report of Network Access Analyzer findings for inbound internet access in multiple AWS accounts
- Set up DNS resolution for hybrid networks in a multi-account AWS environment
- Verify that ELB load balancers require TLS termination
- View AWS Network Firewall logs and metrics by using Splunk
- More patterns

Automate the setup of inter-Region peering with AWS Transit Gateway

Ram Kandaswamy, Amazon Web Services

Summary

AWS Transit Gateway connects virtual private clouds (VPCs) and on-premises networks through a central hub. Transit Gateway traffic always stays on the global Amazon Web Services (AWS) backbone and doesn't traverse the public internet, which reduces threat vectors, such as common exploits and distributed denial of service (DDoS) attacks.

If you need to communicate between two or more AWS Regions, you can use inter-Region Transit Gateway peering to establish peering connections between transit gateways in different Regions. However, manually configuring inter-Region peering with Transit Gateway can be a time-consuming process that has multiple steps. This pattern provides an automated process to remove these manual steps by using code to perform the peering. You can use this approach if you have to repeatedly configure several Regions and AWS accounts during a multi-Region organization setup.

This pattern uses an AWS CloudFormation stack that includes the AWS Step Functions workflow, AWS Lambda functions, AWS Identity and Access Management (IAM) roles, and log groups in Amazon CloudWatch Logs. You can then start a Step Functions execution and create the inter-Region peering connection for your transit gateways. To manually set up inter-Region peering, see [Peer VPCs in different AWS Regions by using AWS Transit Gateway](#).

Prerequisites and limitations

Prerequisites

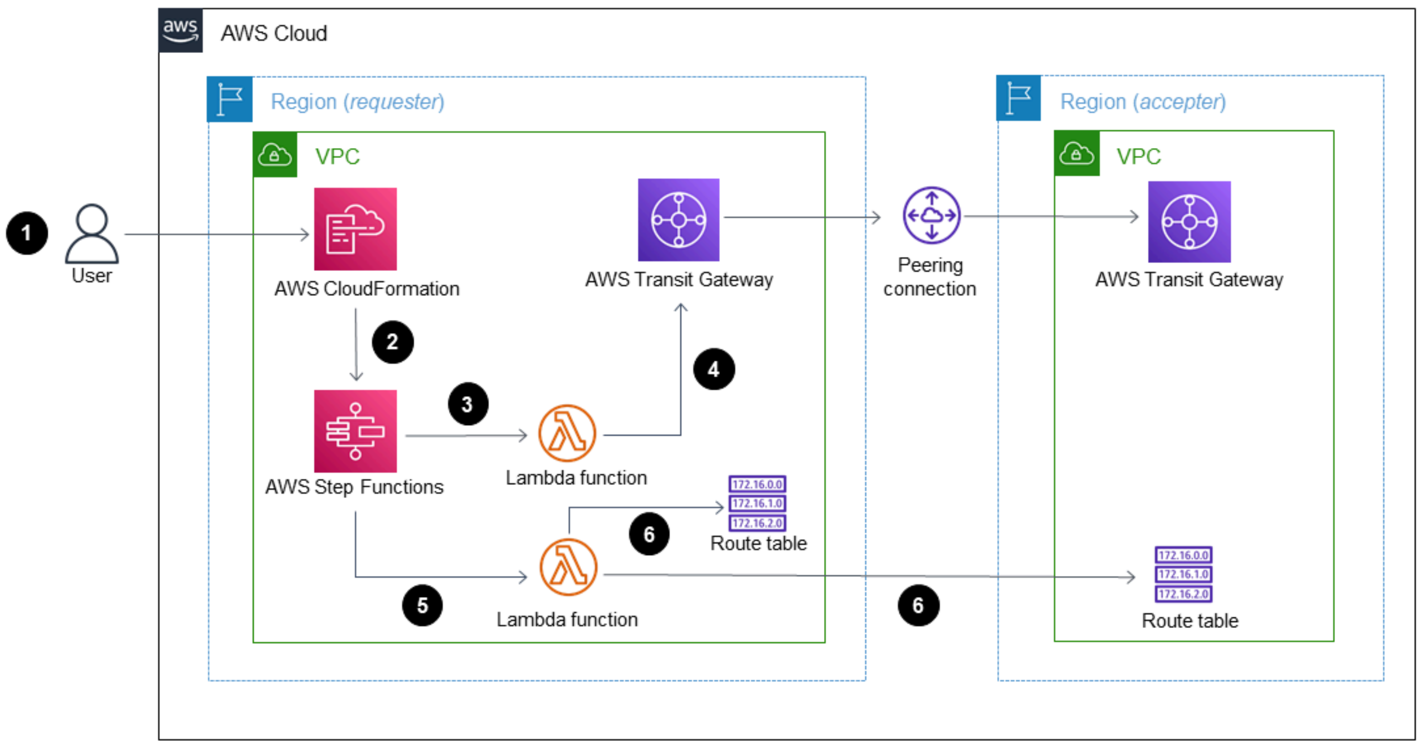
- An active AWS account.
- An existing Amazon Simple Storage Service (Amazon S3) bucket.
- Transit gateways, created and configured in the requestor Region and the acceptor Regions. The *requester* Region is where a peering request is originated and the *acceptor* Regions accept the peering request. For more information about this, see [Creating and accepting a VPC peering connection](#) in the Amazon VPC documentation.
- VPCs, installed and configured in the acceptor and requester Regions. For steps to create a VPC, see [Create the VPC](#) from [Get Started with Amazon VPC](#) in the Amazon VPC documentation.
- The VPCs must use the `addToTransitGateway` tag and `true` value.

- Security groups and network access control lists (ACLs) for your VPCs, configured according to your requirements. For more information about this, see Security groups for your VPC and Network ACLs in the Amazon VPC documentation.

AWS Regions and limitations

- Only certain AWS Regions support inter-Region peering. For a full list of Regions that support inter-Region peering, see the AWS Transit Gateway FAQs.
- In the attached sample code, the requestor Region is assumed to be `us-east-2`, and the acceptor Region is assumed to be `us-west-2`. If you want to configure different Regions, you must edit these values in all Python files. To implement a more complex setup that involves more than two Regions, you can change the Step Function to pass the Regions as a parameter to the Lambda function and run the function for each combination.

Architecture

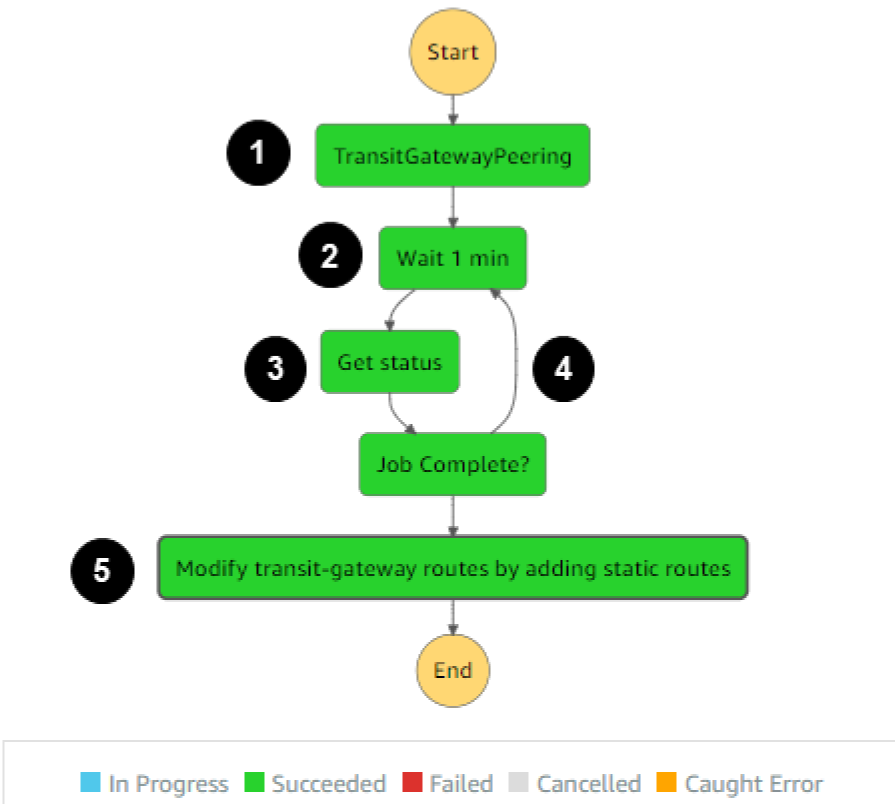


The diagram shows a workflow with the following steps:

1. The user creates an AWS CloudFormation stack.

2. AWS CloudFormation creates a Step Functions state machine that uses a Lambda function. For more information about this, see [Creating a Step Functions state machine that uses Lambda](#) in the AWS Step Functions documentation.
3. Step Functions calls a Lambda function for peering.
4. The Lambda function creates a peering connection between transit gateways.
5. Step Functions calls a Lambda function for route table modifications.
6. The Lambda function modifies the route tables by adding the Classless Inter-Domain Routing (CIDR) block of the VPCs.

Step Functions workflow



The diagram shows the following Step Functions workflow:

1. The Step Functions workflow calls the Lambda function for the transit gateway peering.
2. There is a timer call to wait for one minute.
3. The peering status is retrieved and sent to the condition block. The block is responsible for the looping.

4. If the success condition is not met, the workflow is coded to enter the timer stage.
5. If the success condition is met, a Lambda function is called to modify the route tables. After this call, the Step Functions workflow ends.

Tools

- AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your AWS resources.
- Amazon CloudWatch Logs – CloudWatch Logs helps you centralize the logs from all of your systems, applications, and AWS services that you use.
- AWS Identity and Access Management (IAM) – IAM is a web service for securely controlling access to AWS services.
- AWS Lambda – Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources.
- AWS Step Functions – Step Functions makes it easy to coordinate the components of distributed applications as a series of steps in a visual workflow.

Epics

Automate peering

Task	Description	Skills required
Upload the attached files to your S3 bucket.	Sign in to the AWS Management Console, open the Amazon S3 console, and then upload the `modify-transit-gateway-routes.zip` , `peer-transit-gateway.zip` , and `get-transit-gateway-peering-status.zip` files (attached) to your S3 bucket.	General AWS

Task	Description	Skills required
Create the AWS CloudFormation stack.	Run the following command to create an AWS CloudFormation stack using the `transit-gateway-peering.json` file (attached): `aws cloudformation create-stack --stack-name myteststack --template-body file://sampletemplate.json` The AWS CloudFormation stack creates the Step Functions workflow, the Lambda functions, IAM roles, and CloudWatch log groups. Make sure that the AWS CloudFormation template refers to the S3 bucket that contains the files that you uploaded earlier. **Note** You can also create a stack by using the AWS CloudFormation console. For more information about this, see [Creating a stack on the AWS CloudFormation console]() in the AWS	DevOps engineer

Task	Description	Skills required
	CloudFormation documentation.	
Start a new execution in Step Functions.	Open the Step Functions console and start a new execution. Step Functions calls the Lambda function and creates the peering connection for the transit gateways. You don't need an input JSON file. Verify that an attachment is available and that the connection type is **Peering**. For more information about this, see Start a new execution from Getting started with AWS Step Functions in the AWS Steps Functions documentation.	DevOps engineer, General AWS

Task	Description	Skills required
Verify the routes in the route tables.	Inter-Region peering is established between the transit gateways. The route tables are updated with the peer Region VPC's IPv4 CIDR block range. Open the Amazon VPC console and choose the **Associations** tab in the route table that corresponds to the transit gateway attachment. Verify the VPC CIDR block range of the peered Regions. For detailed steps and instructions, see [Associate a transit gateway route table] in the Amazon VPC documentation.	Network administrator

Related resources

- [Executions in Step Functions]
- [Transit gateway peering attachments]
- [Peer VPCs in different AWS Regions by using AWS Transit Gateway]
- [Interconnecting VPCs across AWS Regions using AWS Transit Gateway - Demo] (video)

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Centralize network connectivity using AWS Transit Gateway

Mydhili Palagummi and Nikhil Marrapu, Amazon Web Services

Summary

This pattern describes the simplest configuration in which AWS Transit Gateway can be used to connect an on-premises network to virtual private clouds (VPCs) in multiple AWS accounts within an AWS Region. Using this setup, you can establish a hybrid network that connects multiple VPC networks in a Region and an on-premises network. This is accomplished by using a transit gateway and a virtual private network (VPN) connection to the on-premises network.

Prerequisites and limitations

Prerequisites

- An account for hosting network services, managed as a member account of an organization in AWS Organizations
- VPCs in multiple AWS accounts, without overlapping Classless Inter-Domain Routing (CIDR) blocks

Limitations

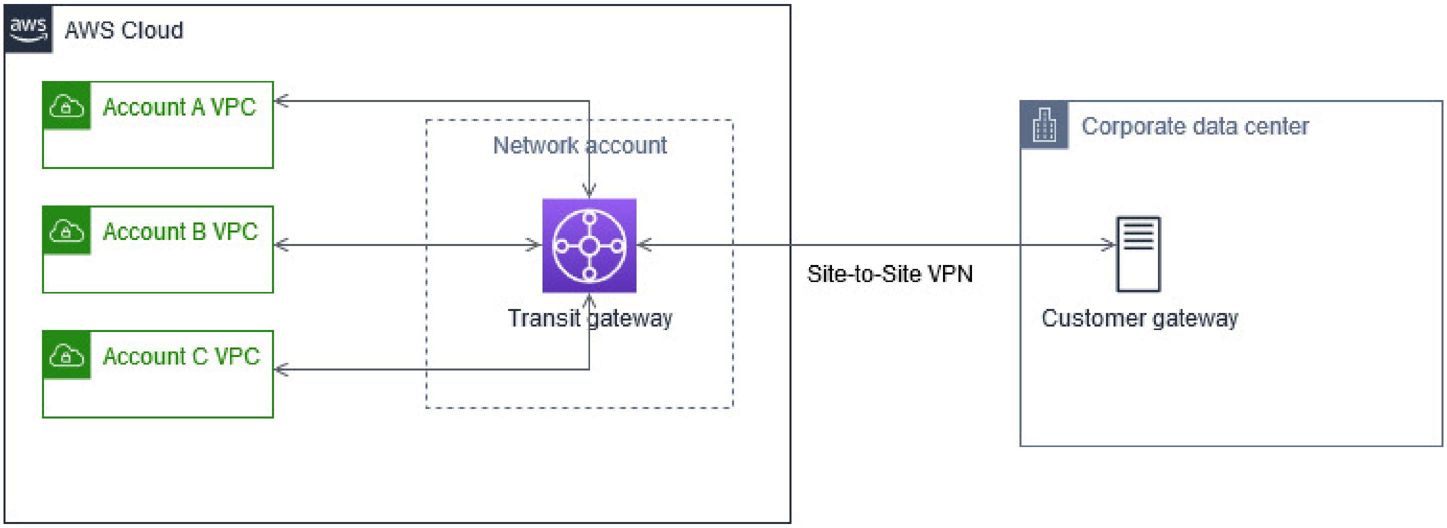
This pattern does not support the isolation of traffic between certain VPCs or the on-premises network. All the networks attached to the transit gateway will be able to reach each other. To isolate traffic, you need to use custom route tables on the transit gateway. This pattern only connects the VPCs and on-premises network by using a single default transit gateway route table, which is the simplest configuration.

Architecture

Target technology stack

- AWS Transit Gateway
- AWS Site-to-Site VPN
- VPC
- AWS Resource Access Manager (AWS RAM)

Target architecture



Tools

AWS services

- [AWS Resource Access Manager (AWS RAM)](#) helps you securely share your resources across your AWS accounts, organizational units, or your entire organization from AWS Organizations.
- [AWS Transit Gateway](#) is a central hub that connects virtual private clouds (VPCs) and on-premises networks.

Epics

Create a transit gateway in the network services account

Task	Description	Skills required
Create a transit gateway.	In the AWS account where you want to host network services, create a transit gateway in the target AWS Region. For instructions, see [Create a transit gateway](#). Note the following:	Network administrator

Task	Description	Skills required
	• Select **Default route table association**. • Select **Default route table propagation**.	

Connect the transit gateway to your on-premises network

Task	Description	Skills required
Set up a customer gateway device for the VPN connectio n.	The customer gateway device is attached on the on-premis es side of the Site-to-Site VPN connection between the transit gateway and your on-premises network. For more information, see [Your customer gateway device](#) in the AWS Site-to-Site VPN documentation. Identify or launch a supported on-premis es customer device and note its public IP address. VPN configuration is completed later in this epic.	Network administrator
In the network services account, create a VPN attachment to the transit gateway.	To set up a connection, create a VPN attachment for the transit gateway. For instructi ons, see [Transit gateway VPN attachments](#).	Network administrator
Configure the VPN on the customer gateway device in your on-premises network.	Download the configura tion file for the Site-to-Site VPN connection associated with the transit gateway and	Network administrator

Task	Description	Skills required
	configure VPN settings on the customer gateway device. For instructions, see Download the configuration file.	

Share the transit gateway in the network services account to other AWS accounts or your organization

Task	Description	Skills required
In the AWS Organizations management account, turn on sharing.	To share the transit gateway with your organization or with certain organizational units, turn on sharing in AWS Organizations. Otherwise , you would need to share the transit gateway for each account individually. For instructions, see Enable resource sharing within AWS Organizations.	AWS systems administrator
Create the transit gateway resource share in the network services account.	To allow VPCs in other AWS accounts within your organization to connect to the transit gateway, in the network services account, use the AWS RAM console to share the transit gateway resource. For instructions, see Create a resource share.	AWS systems administrator

Connect VPCs to the transit gateway

Task	Description	Skills required
Create VPC attachments in individual accounts.	In the accounts to which the transit gateway has been shared, create transit gateway VPC attachments. For instructions, see Create a transit gateway attachment to a VPC.	Network administrator
Accept the VPC attachment requests.	In the network services account, accept the transit gateway VPC attachment requests. For instructions, see Accept a shared attachment.	Network administrator

Configure routing

Task	Description	Skills required
Configure routes in individual account VPCs.	In each individual account VPC, add routes to the on-premises network and to other VPC networks, using the transit gateway as the target. For instructions, see Add and remove routes from a route table.	Network administrator
Configure routes in the transit gateway route table.	Routes from VPCs and the VPN connection should be propagated and should appear in the transit gateway default route table. If needed, create any static routes (one	Network administrator

Task	Description	Skills required
	example is static routes for the static VPN connection) in the transit gateway default route table. For instructions, see [Create a static route]().	
Add security group and network access control list (ACL) rules.	For the EC2 instances and other resources in the VPC, ensure that the security group rules and the network ACL rules allow traffic between VPCs as well as the on-premis es network. For instructi ons, see [Control traffic to resources using security groups]() and [Add and delete rules from an ACL]().	Network administrator

Test connectivity

Task	Description	Skills required
Test connectivity between VPCs.	Ensure that network ACL and security groups allow Internet Control Message Protocol (ICMP) traffic, and then ping from instances in a VPC to another VPC that is also connected to the transit gateway.	Network administrator
Test connectivity between VPCs and the on-premises network.	Ensure that network ACL rules, security group rules, and any firewalls allow ICMP traffic, and then ping	Network administrator

Task	Description	Skills required
	between the on-premises network and the EC2 instances in the VPCs. Network communication must be initiated from the on-premises network first to bring the VPN connection to UP status.	

Related resources

- Building a scalable and secure multi VPC AWS Network Infrastructure (AWS whitepaper)
- Working with shared resources (AWS RAM documentation)
- Working with transit gateways (AWS Transit Gateway documentation)

Configure HTTPS encryption for Oracle JD Edwards EnterpriseOne on Oracle WebLogic by using an Application Load Balancer

Thanigaivel Thirumalai, Amazon Web Services

Summary

This pattern explains how to configure HTTPS encryption for SSL offloading in Oracle JD Edwards EnterpriseOne on Oracle WebLogic workloads. This approach encrypts traffic between the user's browser and a load balancer to remove the encryption burden from the EnterpriseOne servers.

Many users scale the EnterpriseOne JAVA virtual machine (JVM) tier horizontally by using an AWS Application Load Balancer. The load balancer serves as the single point of contact for clients, and distributes incoming traffic across multiple JVMs. Optionally, the load balancer can distribute the traffic across multiple Availability Zones and increase the availability of EnterpriseOne.

The process described in this pattern configures encryption between the browser and the load balancer instead of encrypting the traffic between the load balancer and the EnterpriseOne JVMs. This approach is referred to as *SSL offloading*. Offloading the SSL decryption process from the EnterpriseOne web or application server to the Application Load Balancer reduces the burden on the application side. After SSL termination at the load balancer, the unencrypted traffic is routed to the application on AWS.

Oracle JD Edwards EnterpriseOne is an enterprise resource planning (ERP) solution for organizations that manufacture, construct, distribute, service, or manage products or physical assets. JD Edwards EnterpriseOne supports various hardware, operating systems, and database platforms.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS Identity and Access Management (IAM) role that has permissions to make AWS service calls and manage AWS resources
- An SSL certificate

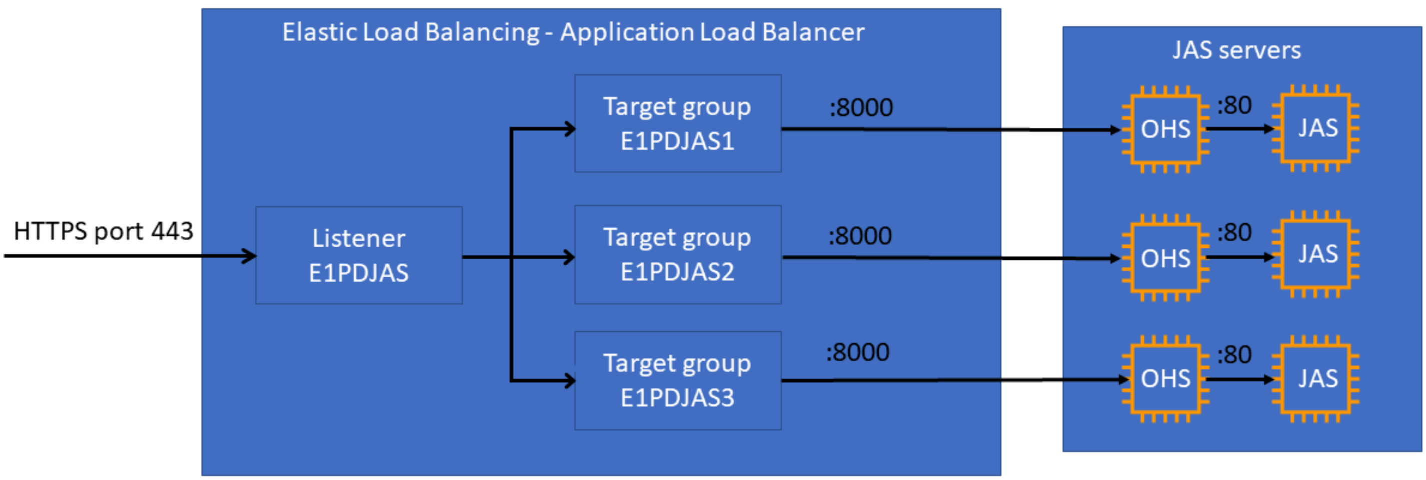
Product versions

- This pattern was tested with Oracle WebLogic 12c, but you can also use other versions.

Architecture

There are multiple approaches to perform SSL offloading. This pattern uses an Application Load Balancer and Oracle HTTP Server (OHS), as illustrated in the following diagram.

The following diagram shows the JD Edwards EnterpriseOne, Application Load Balancer, and Java Application Server (JAS) JVM layout.



Tools

AWS services

- Application Load Balancers distribute incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2 instances), in multiple Availability Zones.
- AWS Certificate Manager (ACM) helps you create, store, and renew public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications.
- Amazon Route 53 is a highly available and scalable DNS web service.

Best practices

- For ACM best practices, see the ACM documentation.

Epics

Set up WebLogic and OHS

Task	Description	Skills required
Install and configure Oracle components.	1. Install Fusion Middleware Infrastructure by following the standard installation process. This program helps you install and configure a WebLogic domain. For instructions, see the [Oracle documentation](). 2. Install OHS by following the standard installation process. For instructions, see the [Oracle documentation](). 3. When installation is complete, start the configuration wizard (`config.sh` file) to configure OHS. • You can update an existing domain or create a new domain. This pattern assumes that you're updating an existing domain. • For **Available Templates**, choose **Oracle Enterprise Manager-Restricted JRF** and **Oracle HTTP Server (Restricted**	JDE CNC, WebLogic administrator

Task	Description	Skills required
	JRF). Selecting these Java Required Files (JRF) options eliminates the connection to an external database. • For **Managed Servers**, **Clusters**, **Server Templates**, **Coherence Clusters**, **Machines**, **Assign Servers to Machines**, **Virtual targets**, and **Partitions**, accept the default configuration values and choose **Next** to move to the next category. • Complete the configuration details (for example, administrator host and port, listen address and port, server name) for the OHS instance (for example, `ohs1`).	

Task	Description	Skills required
Enable the WebLogic plugin at the domain level.	The WebLogic plugin is required for load balancing. To enable the plugin: 1. Log in to the WebLogic administration console by using the link: `http://<WeblogicServer>:<Adminport>/console` 2. Choose **Lock & Edit**, and then choose **Configuration**, **Web Applications**. 3. Choose the **WebLogic Plugin Enabled** (check box or dropdown option). 4. Choose **Save and Activate Changes**.	JDE CNC, WebLogic administrator

Task	Description	Skills required

Edit the configuration file.	The `mod_wl_ohs.conf` file configures proxy requests from OHS to WebLogic.
	
	1. Edit this file. It's located at:
	`$ORACLE_HOME/user_projects/domains/`
	For example:
	`/home/oracle/Oracle/Middleware/Oracle_Home/user_projects/domains/base_domain/config/fmwconfig/components/OHS/instances/ohs1`
	2. Add the WebLogic host (`WebLogicHost`) and port (`WebLogicPort`) values (This pattern assumes localhost and port 8000.)
	3. Add `WLProxySSL` and `WLProxySSLPassThrough` values as follows:
	
	`<VirtualHost *:8000>`
	`<Location /jde>`
	`WLSRequest On`
	`SetHandler weblogic-handler`
	`WebLogicHost localhost`
	`WebLogicPort 8000`
		JDE CNC, WebLogic administrator

Task	Description	Skills required
	``` WLProxySSL On WLProxySSLPassThrough On </Location> </VirtualHost> ```	

Task	Description	Skills required
Start OHS by using the Enterprise Manager.	1. Log in to Enterprise Manager Fusion Middleware by using the link: `http://<WeblogicServer>:<Adminport>/em/` 2. In **Target Navigation**, under **HTTP Server**, select the OHS instance (for example, `ohs1`). 3. Choose **Shut Down** and **Start Up** to restart the OHS instance. 4. When OHS setup is complete, you can connect to the EnterpriseOne HTML client by using your HTTP server host name with port 8000 instead of the EnterpriseOne server host name. • Old link: `http://<Webserver>:80/jde/owhtml` • New link: `http://<HTTP server or web server>:8000/jde/owhtml` If you use a port other than the default Oracle HTTP port, edit the `httpd.con`	JDE CNC, WebLogic administrator

Task	Description	Skills required

	f file to add a listener for that port in two places:
	`#[Listen] OHS_LISTEN_PORT`
	`Listen 8000`
	and:
	`#`
	`ServerName <Weblogic Server1>:8000`	

Configure the Application Load Balancer

Task	Description	Skills required
Set up a target group.	1. Create a target group for the HTTP server port 8000. 2. Register the targets under the target group with the same port. 3. Check the status of the targets to confirm that they are healthy. 4. Configure the health check settings as necessary. For detailed instructions, see the [Elastic Load Balancing documentation]().	AWS administrator
Set up the load balancer.	1. Create an Application Load Balancer with default	AWS administrator

Task	Description	Skills required
	attributes and the required virtual private cloud (VPC), security groups, and subnets. For instructi ons, see the Elastic Load Balancing documentation. 2. Add a listener entry for HTTPS 443 and forward it to the target group that you created in the previous step. (For instructions, see the Elastic Load Balancing documentation.) An HTTPS listener requires an SSL certificate. You can choose a certificate from ACM or upload one. 3. For both listeners, enable stickiness by following the instructions in the Elastic Load Balancing documenta tion.	
Add a Route 53 (DNS) record.	(Optional) You can add an Amazon Route 53 DNS record for the subdomain. This record would point to your Application Load Balancer. For instructions, see the Route 53 documentation.	AWS administrator

Troubleshooting

Issue	Solution
HTTP server doesn't appear.	If **HTTP Server** doesn't appear in the **Target Navigation** list on the Enterprise Manager console, follow these steps: 1. Under **WebLogic Domain**, **Administration**, choose **OHS Instances**. 2. Choose **Create** to create a new OHS instance. 3. Provide an instance name, and then choose **OK** to create the instance. When the instance has been created and changes have been activated, you will be able to see the HTTP server in the **Target Navigation** panel.

Related resources

AWS documentation

- Application Load Balancers
- Working with public hosted zones
- Working with private hosted zones

Oracle documentation:

- Overview of Oracle WebLogic Server Proxy Plug-In
- Installing WebLogic Server using the Infrastructure Installer
- Installing and Configuring Oracle HTTP Server

Connect to Application Migration Service data and control planes over a private network

Dipin Jain and Mike Kuznetsov, Amazon Web Services

Summary

This pattern explains how you can connect to an AWS Application Migration Service data plane and control plane on a private, secured network by using interface VPC endpoints.

Application Migration Service is a highly automated lift-and-shift (rehost) solution that simplifies, expedites, and reduces the cost of migrating applications to AWS. It enables companies to rehost a large number of physical, virtual, or cloud servers without compatibility issues, performance disruption, or long cutover windows. Application Migration Service is available from the AWS Management Console. This enables seamless integration with other AWS services, such as AWS CloudTrail, Amazon CloudWatch, and AWS Identity and Access Management (IAM).

You can connect from a source data center to a data plane—that is, to a subnet that serves as a staging area for data replication in the destination VPC—over a private connection by using AWS VPN services, AWS Direct Connect, or VPC peering in Application Migration Service. You can also use interface VPC endpoints powered by AWS PrivateLink to connect to an Application Migration Service control plane over a private network.

Prerequisites and limitations

Prerequisites

- **Staging area subnet** – Before you set up Application Migration Service, create a subnet to be used as a staging area for data replicated from your source servers to AWS (that is, a data plane). You must specify this subnet in the Replication Settings template when you first access the Application Migration Service console. You can override this subnet for specific source servers in the Replication Settings template. Although you can use an existing subnet in your AWS account, we recommend that you create a new, dedicated subnet for this purpose.
- **Network requirements** – The replication servers that are launched by Application Migration Service in your staging area subnet have to be able to send data to the Application Migration Service API endpoint at `https://mgn.<region>.amazonaws.com/`, where `<region>` is the code for the AWS Region you are replicating to (for example, `https://mgn.us-east-1.amazonaws.com`). Amazon Simple Storage Service (Amazon S3) service URLs are required for downloading Application Migration Service software.

- The AWS Replication Agent installer should have access to the Amazon Simple Storage Service (Amazon S3) bucket URL of the AWS Region you are using with Application Migration Service.
- The staging area subnet should have access to Amazon S3.
- The source servers on which the AWS Replication Agent is installed must be able to send data to the replication servers in the staging area subnet and to the Application Migration Service API endpoint at `https://mgn.<region>.amazonaws.com/`.

The following table lists the required ports.

Source	Destination	Port	For more informati on, see
Source data center	Amazon S3 service URLs	443 (TCP)	Communication over TCP port 443
Source data center	AWS Region-specific console address for Application Migration Service	443 (TCP)	Communication between the source servers and Applicati on Migration Service over TCP port 443
Source data center	Staging area subnet	1500 (TCP)	Communication between the source servers and the staging area subnet over TCP port 1500
Staging area subnet	AWS Region-specific console address for Application Migration Service	443 (TCP)	Communication between the staging area subnet and Application Migration Service over TCP port 443
Staging area subnet	Amazon S3 service URLs	443 (TCP)	Communication over TCP port 443

Staging area subnet	Amazon Elastic Compute Cloud (Amazon EC2) endpoint of the subnet's AWS Region	443 (TCP)	Communication over TCP port 443

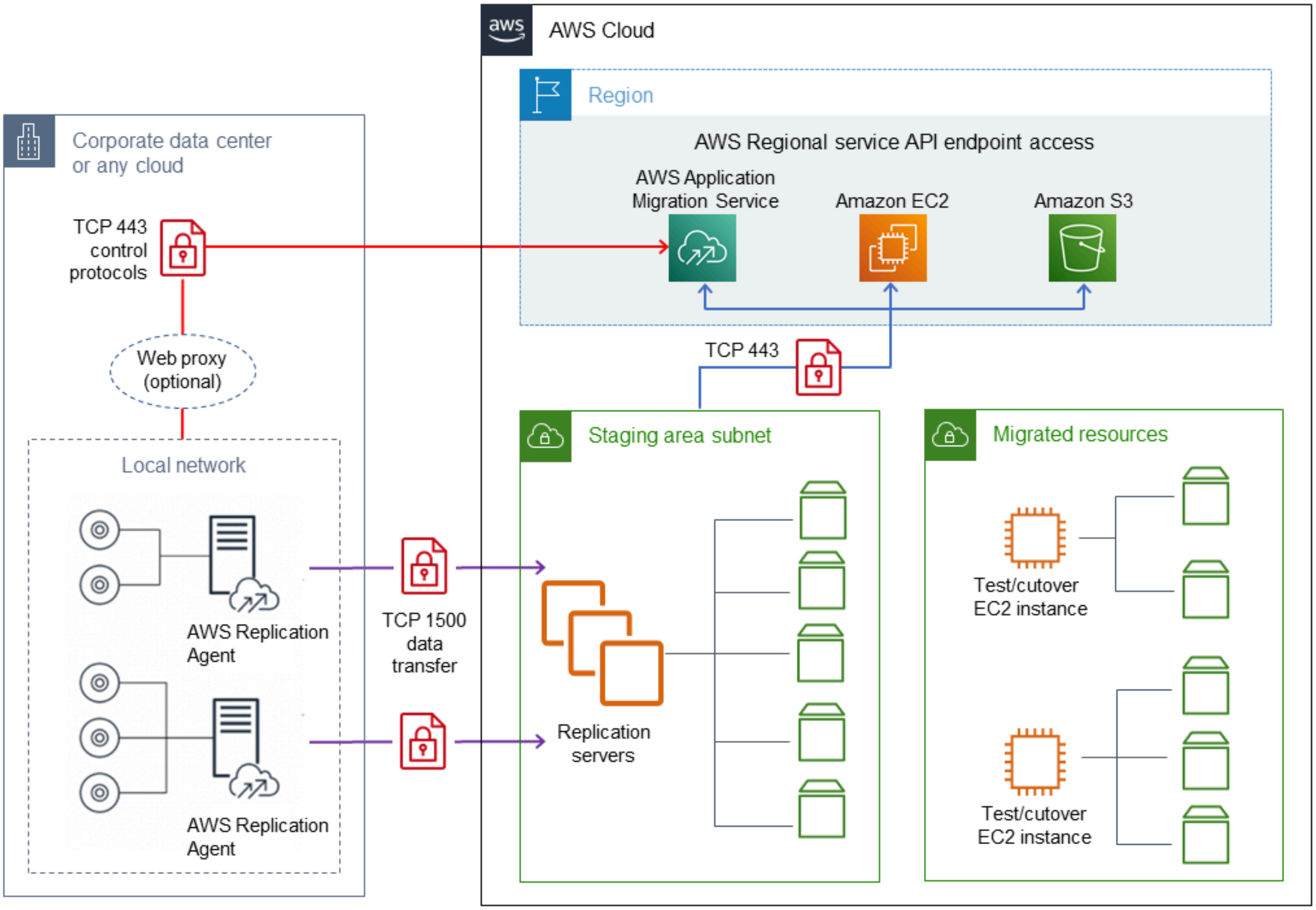
Limitations

Application Migration Service isn't currently available in all AWS Regions and operating systems.

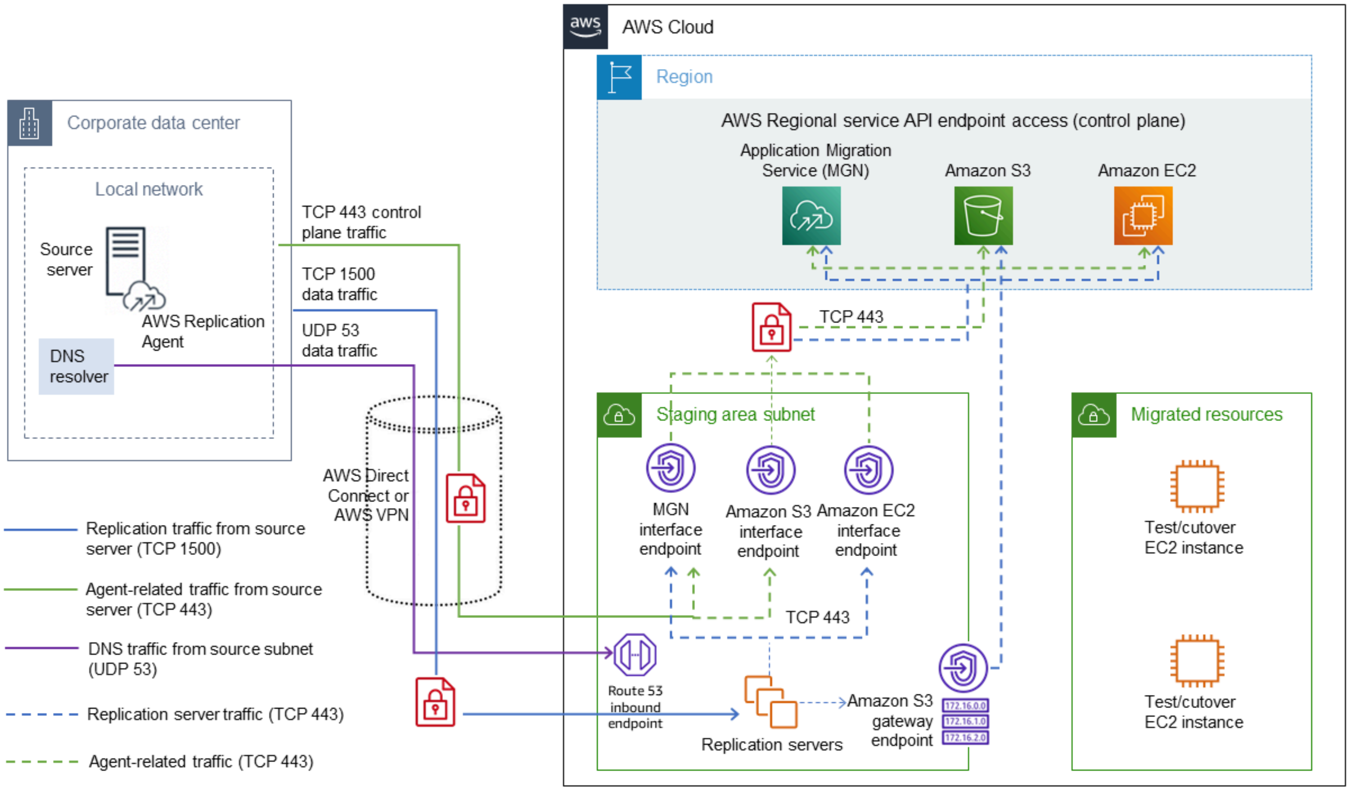
- Supported AWS Regions
- Supported operating systems

Architecture

The following diagram illustrates the network architecture for a typical migration. For more information about this architecture, see the Application Migration Service documentation and the Application Migration Service service architecture and network architecture video.



The following detailed view shows the configuration of interface VPC endpoints in the staging area VPC to connect Amazon S3 and Application Migration Service.



Tools

- [AWS Application Migration Service](#) simplifies, expedites, and reduces the cost of rehosting applications on AWS.
- [Interface VPC endpoints](#) enable you to connect to services that are powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Epics

Create endpoints for Application Migration Service, Amazon EC2, and Amazon S3

Task	Description	Skills required
Configure the interface endpoint for Application Migration Service.	The source data center and staging area VPC connect privately to the Application	Migration lead

Task	Description	Skills required
	Migration Service control plane through the interface endpoint that you create in the target staging area VPC. To create the endpoint: 1. Open the [Amazon Virtual Private Cloud (Amazon VPC) console](#). 2. In the navigation pane, choose **Endpoints**, **Create Endpoint**. 3. For **Service category**, choose **AWS services**. 4. For **Service Name**, enter `com.amazonaws.<region>.mgn` . For **Type**, choose **Interface**. 5. For **VPC**, select a target staging area VPC to create the endpoint. 6. For **Subnets**, select the subnets in which to create the endpoint network interfaces. 7. To turn on private DNS for the interface endpoint, in the **Additional settings** section, select **Enable DNS Name**. 8. Select a security group that allows ingress from the staging area VPC subnet over TCP 443.	

Task	Description	Skills required
	9. Choose **Create endpoint**. For more information, see [Access an AWS service using an interface VPC endpoint in the Amazon VPC](#) documenta tion.	
Configure the interface endpoint for Amazon EC2.	The staging area VPC connects privately to the Amazon EC2 API through the interface endpoint that you create in the target staging area VPC. To create the endpoint, follow the instructi ons provided in the previous story. • For service name, enter `com.amazonaws.<reg ion>.ec2` . For **Type**, choose **Interface**. • The security group must allow inbound HTTPS traffic from the staging area VPC subnet over port 443. • In the **Additional settings** section, select **Enable DNS name**.	Migration lead

Task	Description	Skills required
Configure the interface endpoint for Amazon S3.	The source data center and staging area VPC connect privately to the Amazon S3 API through the interface endpoint that you create in the target staging area VPC. To create the endpoint, follow the instructions provided in the first story. • For **Service Name**, enter `com.amazonaws.<region>.s3` . For **Type**, choose **Interface**. • The VPC security group must allow inbound HTTPS traffic from the staging area VPC subnet over port 443. • In the **Additional settings** section, clear **Enable DNS name**. Amazon S3 interface endpoints do not support private DNS names. **Note** You use an interface endpoint because gateway endpoint connections cannot be extended out of a VPC. (For details, see	Migration lead

Task	Description	Skills required
	the AWS PrivateLink documentation.)	
Configure the Amazon S3 Gateway endpoint.	During the configuration phase, the replication server has to connect to an S3 bucket to download the AWS Replication Server's software updates. However, Amazon S3 interface endpoints do not support private DNS names, and there is no way to supply an Amazon S3 endpoint DNS name to a replication server. To mitigate this issue, you create an Amazon S3 gateway endpoint in the VPC that the staging area subnet belongs to, and update the staging subnet's route tables with the relevant routes. For more information, see Create a gateway endpoint in the AWS PrivateLink documentation.	Cloud administrator

Task	Description	Skills required
Configure on-premises DNS to resolve private DNS names for endpoints.	The interface endpoints for Application Migration Service and Amazon EC2 have private DNS names that can be resolved in the VPC. However, you also need to configure on-premises servers to resolve private DNS names for these interface endpoints. There are multiple ways to configure these servers. In this pattern, we tested this functionality by forwarding on-premises DNS queries to the Amazon Route 53 Resolver inbound endpoint in the staging area VPC. For more information, see [Resolving DNS queries between VPCs and your network] in the Route 53 documentation.	Migration engineer

Connect to the Application Migration Service control plane over a private link

Task	Description	Skills required
Install AWS Replication Agent by using AWS PrivateLink.	1. Download the AWS Replication Agent to a private S3 bucket in the destination Region. 2. Log in to the source servers to be migrated.	Migration engineer

Task	Description	Skills required
	The AWS Replication Agent installer needs network access to the Application Migration Service and Amazon S3 endpoints. Because your on-premises network isn't open to Application Migration Service and Amazon S3 public endpoints, you must install the Agent with the help of the interface endpoints you created in the previous steps by using AWS PrivateLink. Here's an example for Linux: 1. Download the Agent by using the command: `wget -O ./aws-replication-installer-init.py \ https://aws-application-migration-service-<aws_region>.bucket.<s3-endpoint-DNS-name>/latest/linux/aws-replication-installer-init.py` For example, if the DNS name of the Amazon S3 interface endpoint is	

Task	Description	Skills required
	`vpce-009c8b07adb052a11-qgf8q50y.s3.us-west-1.vpce.amazonaws.com` and the AWS Region is `us-west-1`, you would use the command: `wget -O ./aws-replication-installer-init.py \ https://aws-application-migration-service-us-west-1.bucket.vpce-009c8b07adb052a11-qgf8q50y.s3.us-west-1.vpce.amazonaws.com/latest/linux/aws-replication-installer-init.py` **Note** `bucket` is a static keyword that you must add before the Amazon S3 interface endpoint DNS name. For more information, see the [Amazon S3 documentation](). 2. Install the Agent: • If you selected **Enable DNS name** when you	

Task	Description	Skills required

	created an interface endpoint for Application Migration Service, run the command:
	`sudo python3 aws-replication-installer-init.py \ --region <aws_region> \ --aws-access-key-id <access-key> \ --aws-secret-access-key <secret-key> \ --no-prompt \ --s3-endpoint <s3-endpoint-DNS-name>`
	• If you didn't select **Enable DNS name** when you created the interface endpoint for Application Migration Service, run the command:
	`sudo python3 aws-replication-installer-init.py \ --region <aws_region> \ --aws-access-key-id <access-key> \ --aws-secret-access-key <secret-key> \ --no-prompt \`	

Task	Description	Skills required

	``
	    --s3-endpoint
	 <s3-endpoint-DNS-n
	ame> \
	    --endpoint
	 <mgn-endpoint-DNS-
	name>
	``
	For more information, see [AWS Replication Agent installation instructions] in the Application Migration Service documentation.
	
	After you have establish ed your connection with Application Migration Service and installed the AWS Replication Agent, follow the instructions in the [Application Migration Service documenta tion] to migrate your source servers to your target VPC and subnet.	

Related resources

Application Migration Service documentation

- Concepts
- Migration workflow
- Quick start guide
- FAQ
- Troubleshooting

Additional resources

- Rehosting your applications in a multi-account architecture on AWS by using VPC interface endpoints (AWS Prescriptive Guidance guide)
- AWS Application Migration Service – A Technical Introduction (AWS Training and Certification walkthrough)
- AWS Application Migration Service architecture and network architecture (video)

Additional information

Troubleshooting *AWS* Replication Agent installations on Linux servers

If you get a **gcc** error on an Amazon Linux server, configure the package repository and use the following command:

```
## sudo yum groupinstall "Development Tools"
```

Create Infoblox objects using AWS CloudFormation custom resources and Amazon SNS

Tim Sutton, Amazon Web Services

Summary

Notice: AWS Cloud9 is no longer available to new customers. Existing customers of AWS Cloud9 can continue to use the service as normal. Learn more

Infoblox Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and IP address management (Infoblox DDI) enables you to centralize and efficiently control a complex hybrid environment. With Infoblox DDI, you can discover and record all network assets in one authoritative IP address management (IPAM) database, in addition to managing DNS on premises and on the Amazon Web Services (AWS) Cloud by using the same appliances.

This pattern describes how to use an AWS CloudFormation custom resource to create Infoblox objects (for example, DNS records or IPAM objects) by calling the Infoblox WAPI API. For more information about the Infoblox WAPI, see the WAPI documentation in the Infoblox documentation.

By using this pattern's approach, you can obtain a unified view of DNS records and IPAM configurations for your AWS and on-premises environments, in addition to removing manual processes that create records and provision your networks. You can use this pattern's approach for the following use cases:

- Adding an A record after creating an Amazon Elastic Compute Cloud (Amazon EC2) instance
- Adding a CNAME record after creating an Application Load Balancer
- Adding a network object after creating a virtual private cloud (VPC)
- Providing the next network range and using that range to create subnets

You can also extend this pattern and use other Infoblox device features such as adding different DNS record types or configuring Infoblox vDiscovery.

The pattern uses a hub-and-spoke design in which the hub requires connectivity to the Infoblox appliance on the AWS Cloud or on premises and uses AWS Lambda to call the Infoblox API. The spoke is in the same or a different account in the same organization in AWS Organizations, and calls the Lambda function by using an AWS CloudFormation custom resource.

Prerequisites and limitations

Prerequisites

- An existing Infoblox appliance or grid, installed on the AWS Cloud, on premises, or both, and configured with an admin user that can administer IPAM and DNS actions. For more information about this, see About admin accounts in the Infoblox documentation.
- An existing DNS authoritative zone that you want to add records on the Infoblox appliance. For more information about this, see Configuring authoritative zones in the Infoblox documentation.

- Two active AWS accounts in AWS Organizations. One account is the hub account and the other account is the spoke account.
- The hub and spoke accounts must be in the same AWS Region.
- The hub account's VPC must connect to the Infoblox appliance; for example, by using AWS Transit Gateway or VPC peering.
- AWS Serverless Application Model (AWS SAM), locally installed and configured with AWS Cloud9 or AWS CloudShell.
- The `Infoblox-Hub.zip` and `ClientTest.yaml` files (attached), downloaded to the local environment that contains AWS SAM.

Limitations

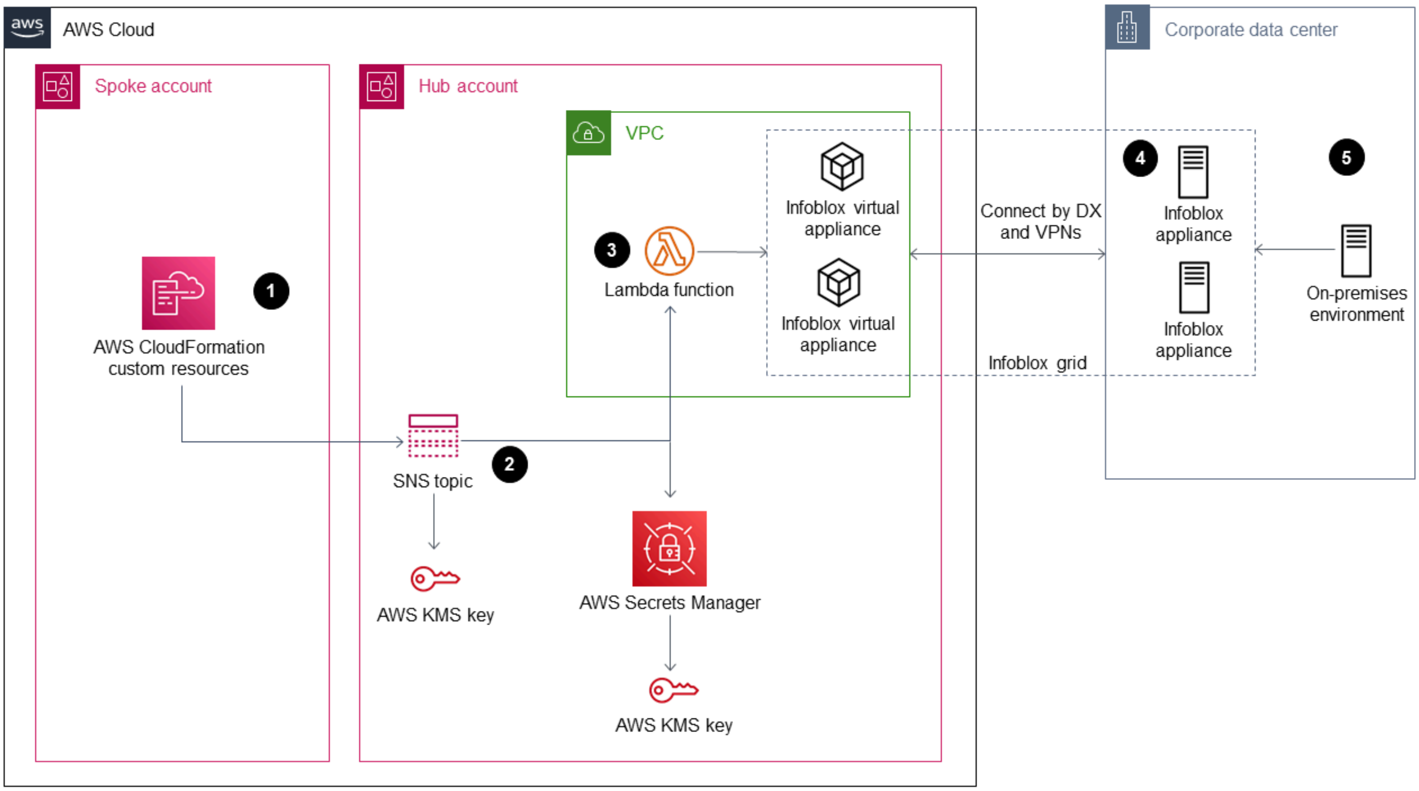
- The AWS CloudFormation custom resource's service token must be from the same Region where the stack is created. We recommend that you use a hub account in each Region, instead of creating an Amazon Simple Notification Service (Amazon SNS) topic in one Region and calling the Lambda function in another Region.

Product versions

- Infoblox WAPI version 2.7

Architecture

The following diagrams shows this pattern's workflow.



The diagram shows the following components for this pattern's solution:

1. AWS CloudFormation custom resources enable you to write custom provisioning logic in templates that AWS CloudFormation runs when you create, update, or delete stacks. When you create a stack, AWS CloudFormation sends a `create` request to an SNS topic that's monitored by an application running on an EC2 instance.
2. The Amazon SNS notification from the AWS CloudFormation custom resource is encrypted through a specific AWS Key Management Service (AWS KMS) key and access is restricted to accounts in your organization in Organizations. The SNS topic initiates the Lambda resource that calls the Infoblox WAPI API.
3. Amazon SNS invokes the following Lambda functions that take the Infoblox WAPI URL, the user name, and password AWS Secrets Manager Amazon Resource Names (ARNs) as environment variables:
   - `dnsapi.lambda_handler` – Receives the `DNSName`, `DNSType`, and `DNSValue` values from the AWS CloudFormation custom resource and uses these to create DNS A records and CNAMES.
   - `ipaddr.lambda_handler` – Receives the `VPCCIDR`, `Type`, `SubnetPrefix`, and `Network Name` values from the AWS CloudFormation custom resource and uses these to add the

network data into the Infoblox IPAM database or provide the custom resource with the next available network that can be used to create new subnets.

- `describeprefixes.lambda_handler` – Calls the `describe_managed_prefix_lists` AWS API by using the `"com.amazonaws."+Region+".s3"` filter to retrieve the required `prefix ID`.

> **Important**
>
> These Lambda functions are written in Python and are similar to each other but call different APIs.

4. You can deploy the Infoblox grid as physical, virtual, or cloud-based network appliances. It can be deployed on-premises or as a virtual appliance using a range of hypervisors, including VMware ESXi, Microsoft Hyper-V, Linux KVM, and Xen. You can also deploy the Infoblox grid on the AWS Cloud with an Amazon Machine Image (AMI).
5. The diagram shows a hybrid solution for the Infoblox grid that provides DNS and IPAM to resources on the AWS Cloud and on premises.

Technology stack

- AWS CloudFormation
- IAM
- AWS KMS
- AWS Lambda
- AWS SAM
- AWS Secrets Manager
- Amazon SNS
- Amazon VPC

Tools

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- AWS Serverless Application Model (AWS SAM) is an open-source framework that helps you build serverless applications in the AWS Cloud.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Code

You can use the `ClientTest.yaml` sample AWS CloudFormation template (attached) to test the Infoblox hub. You can customize the AWS CloudFormation template to include the custom resources from the following table.

Create an A record using the Infoblox spoke custom resource	Return values:
	
	`infobloxref` – Infoblox references
	
	Example resource:
	
	`ARECORDCustomResource:`
	
	`  Type: "Custom::InfobloxAPI"`
	
	`  Properties:`

```
    ServiceToken: !Sub  arn:aws:sns:
${AWS::Region}:${HubAccountID}:Ru
nInfobloxDNSFunction

    DNSName: 'arecordtest.compa
ny.com'

    DNSType: 'ARecord'

    DNSValue: '10.0.0.1'
```

Create a CNAME record using the Infoblox spoke custom resource

Return values:

`infobloxref` – Infoblox references

Example resource:

```
CNAMECustomResource:

  Type: "Custom::InfobloxAPI"

  Properties:

    ServiceToken: !Sub arn:aws:sns:
${AWS::Region}:${HubAccountID}:Ru
nInfoblox

    DNSFunction

    DNSName: 'cnametest.company.com'

    DNSType: 'cname'

    DNSValue: 'aws.amazon.com'
```

Create a network object using the Infoblox spoke custom resource

Return values:

`infobloxref` – Infoblox references

`network` – Network range (the same as `VPCCIDR`)

Example resource:

```
VPCCustomResource:

  Type: 'Custom::InfobloxAPI'

  Properties:

    ServiceToken: !Sub  arn:aws:sns:
${AWS::Region}:${HubAccountID}:Ru
nInfobloxNextSubnetFunction

    VPCCIDR: !Ref VpcCIDR

    Type: VPC

    NetworkName: My-VPC
```

Retrieve the next available subnet using the Infoblox spoke custom resource	**Return values:**
	`infobloxref` – Infoblox references
	`network` – The subnet's network range
	Example resource:
	`Subnet1CustomResource:`
	`  Type: 'Custom::InfobloxAPI'`
	`  DependsOn: VPCCustomResource`
	`  Properties:`
	`    ServiceToken: !Sub  arn:aws:sns:${AWS::Region}:${HubAccountID}:RunInfobloxNextSubnetFunction`
	`    VPCCIDR: !Ref VpcCIDR`
	`    Type: Subnet`
	`    SubnetPrefix: !Ref SubnetPrefix`
	`NetworkName: My-Subnet`

Epics

Create and configure the hub account's VPC

Task	Description	Skills required
Create a VPC with a connection to the Infoblox appliance.	Sign in to the AWS Management Console for your hub account and create a VPC by following the steps in the Amazon VPC on the AWS Cloud Quick Start reference	Network administrator, System administrator

Task	Description	Skills required
	deployment from AWS Quick Starts. **Important** The VPC must have HTTPS connectivity to the Infoblox appliance and we recommend that you use a private subnet for this connection.	

Task	Description	Skills required
(Optional) Create the VPC endpoints for private subnets.	VPC endpoints provide connectivity to public services for your private subnets. The following endpoints are required: • A gateway endpoint for Amazon Simple Storage Service (Amazon S3) to allow Lambda to communicate with AWS CloudFormation • An interface endpoint for Secrets Manager to enable connectivity with Secrets Manager • An interface endpoint for AWS KMS to allow the encryption of the SNS topic and Secrets Manager secret For more information about creating endpoints for private subnets, see [VPC endpoints](#) in the Amazon VPC documentation.	Network administrator, Systems administrator

Deploy the Infoblox hub

Task	Description	Skills required
Build the AWS SAM template.	1. Run the `unzip Infoblox-Hub.zip`	Developer, System administrator

Task	Description	Skills required
	command in the environment that contains AWS SAM. 2. Run the `cd Hub/` command to change your directory to the Hub directory. 3. Run the `sam build` command to process the AWS SAM template file, application code, and any language-specific files and dependencies. The `sam build` command also copies build artifacts in the format and location expected for the following story.	

Task	Description	Skills required

Deploy the AWS SAM template.	The `sam deploy` command takes the required parameters and saves them into the `samconfig.toml` file, stores the AWS CloudFormation template and Lambda functions in an S3 bucket, and then deploys the AWS CloudFormation template into your hub account.
	
	The following sample code shows how to deploy the AWS SAM template:
	
	`$ sam deploy --guided`
	
	`Configuring SAM deploy`
	`======================`
	`        Looking for config file [samconfig.toml] :  Found`
	`        Reading default arguments  :  Success`
	`        Setting default arguments for 'sam deploy'`
	`        =========================================`
	`        Stack Name [Infoblox-Hub]:`
	`        AWS Region [eu-west-1]:`
	`        Parameter InfobloxUsername:`
	`        Parameter InfobloxPassword:`
		Developer, System administrator

Task	Description	Skills required

	<pre> Parameter
	 InfobloxIPAddress
	 [xxx.xxx.xx.xxx]:
	        Parameter
	 AWSOrganisationID [o-
	xxxxxxxxx]:
	        Parameter VPCID
	 [vpc-xxxxxxxxx]:
	        Parameter
	 VPCCIDR [xxx.xxx.
	xxx.xxx/16]:
	        Parameter
	 VPCSubnetID1 [subnet-x
	xx]:
	        Parameter
	 VPCSubnetID2 [subnet-x
	xx]:
	        Parameter
	 VPCSubnetID3 [subnet-x
	xx]:
	        Parameter
	 VPCSubnetID4 []:
	        #Shows you
	 resources changes to be
	 deployed and require a
	 'Y' to initiate deploy
	        Confirm changes
	 before deploy [Y/n]: y
	        #SAM needs
	 permission to be
	 able to create roles
	 to connect to the
	 resources in your
	 template
	Allow SAM CLI IAM role
	 creation [Y/n]: n
	Capabilities [['CAPABI
	LITY_NAMED_IAM']]:
	        Save arguments
	 to configuration file
	 [Y/n]: y</pre>	

Task	Description	Skills required

	``
	        SAM configura
	tion file [samconfi
	g.toml]:
	        SAM configura
	tion environment
	 [default]:
	``
	
	Important
	
	You must use the `--guided` option each time because the Infoblox sign-in credentials are not stored in the `samconfig.toml` file.	

Related resources

- Getting started with WAPIs using Postman (Infoblox Blog)
- Provisioning vNIOS for AWS Using the BYOL Model (Infoblox documentation)
- quickstart-aws-vpc (GitHub repo)
- describe_managed_prefix_lists (AWS SDK for Python documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Customize Amazon CloudWatch alerts for AWS Network Firewall

Jason Owens, Amazon Web Services

Summary

The pattern helps you customize the Amazon CloudWatch alerts that are generated by AWS Network Firewall. You can use predefined rules or create custom rules that determine the message, metadata, and severity of the alerts. You can then act upon these alerts or automate responses by other Amazon services, such as Amazon EventBridge.

In this pattern, you generate Suricata-compatible firewall rules. Suricata is an open-source threat detection engine. You first create simple rules and then test them to confirm that the CloudWatch alerts are generated and logged. Once you have successfully tested the rules, you modify them to define custom messages, metadata, and severities, and you then test once more to confirm the updates.

Prerequisites and limitations

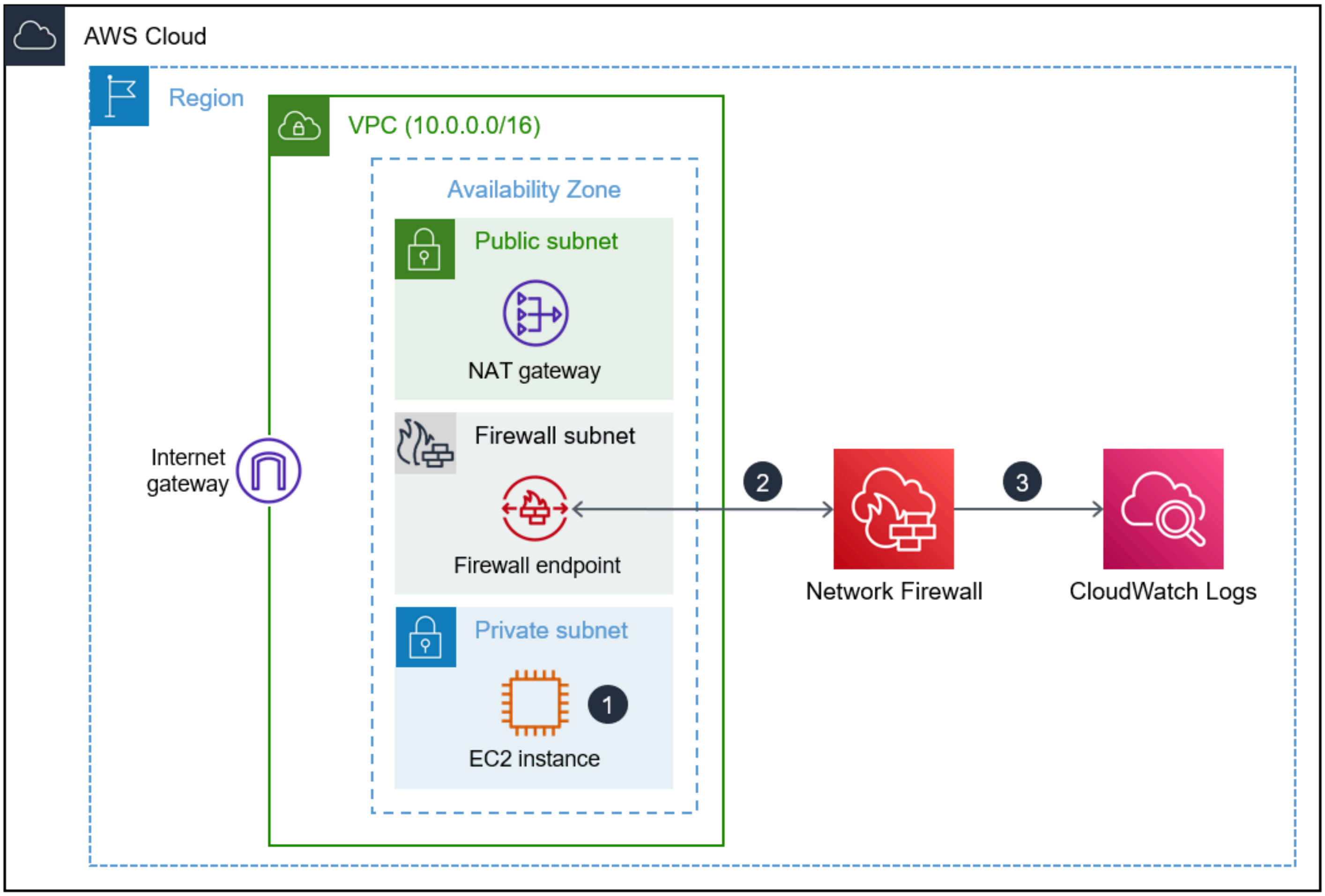
Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) installed and configured on your Linux, macOS, or Windows workstation. For more information, see Installing or updating the latest version of the AWS CLI.
- AWS Network Firewall installed and configured to use CloudWatch Logs. For more information, see Logging network traffic from AWS Network Firewall.
- An Amazon Elastic Compute Cloud (Amazon EC2) instance in a private subnet of a virtual private cloud (VPC) that is protected by Network Firewall.

Product versions

- For version 1 of AWS CLI, use 1.18.180 or later. For version 2 of AWS CLI, use 2.1.2 or later.
- The classification.config file from Suricata version 5.0.2. For a copy of this configuration file, see the Additional information section.

Architecture



The architecture diagram shows the following workflow:

1. An Amazon EC2 instance in a private subnet makes a request by using either curl or Wget.
2. Network Firewall processes the traffic and generates an alert.
3. Network Firewall sends the logged alerts to CloudWatch Logs.

Tools

AWS services

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service for virtual private clouds (VPCs) in the AWS Cloud.

Other tools

- curl is an open-source command line tool and library.
- GNU Wget is a free command line tool.

Epics

Create the firewall rules and rule group

Task	Description	Skills required
Create rules.	1. In a text editor, create a list of rules that you want to add to the firewall. Each rule must be on a separate line. The value in the `classtype` parameter is from the default Suricata classification configuration file. For the full configura tion file contents, see the Additional information section. The following are two examples of rules. `alert http any any -> any any (content:"badstuff"; classtype:misc-activity; sid:3; rev:1;)`	AWS systems administrator, Network administrator

Task	Description	Skills required
	`alert http any any -> any any (content: "morebadstuff"; classtype:bad-unknown; sid:4; rev:1;)` 2. Save the rules in a file named `custom.rules` .	

Task	Description	Skills required

Create the rule group.	In the AWS CLI, enter the following command. This creates the rule group.
	```# aws network-firewall create-rule-group \ --rule-group-name custom --type STATEFUL \ --capacity 10 --rules file://custom.rules \ --tags Key=environment,Value=development```
	The following is an example output. Make note of the `RuleGroupArn` , which you need in a later step.
	```{ "UpdateToken": "4f998d72-973c-490a-bed2-fc3460547e23", "RuleGroupResponse": { "RuleGroupArn": "arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom", "RuleGroupName": "custom", "RuleGroupId": "238a8259-9eaf-48bb-90af-5e690cf8c48b", "Type": "STATEFUL",```
		AWS systems administrator

Task	Description	Skills required

	``"Capacity": 10,
	        "RuleGrou
	pStatus": "ACTIVE",
	        "Tags": [
	            {
	                "Key":
	 "environment",
	                "Value":
	 "development"
	            }
	        ]
	    }```	

Update the firewall policy

Task	Description	Skills required
Get the ARN of the firewall policy.	In the AWS CLI, enter the following command. This returns the Amazon Resource Name (ARN) of the firewall policy. Record the ARN for use later in this pattern. `# aws network-firewall describe-firewall \ --firewall-name aws-network-firewall-anfw \ --query 'Firewall.FirewallPolicyArn'` The following is an example ARN that is returned by this command.	AWS systems administrator

Task	Description	Skills required
	`"arn:aws:network-firewall:us-east-2:1234567890:firewall-policy/firewall-policy-anfw"`	

Task	Description	Skills required
Update the firewall policy.	In a text editor, copy the paste the following code. Replace `<RuleGroupArn>` with the value you recorded in the previous epic. Save the file as `firewall-policy-anfw.json` . ```{ "StatelessDefaultActions": ["aws:forward_to_sfe"], "StatelessFragmentDefaultActions": ["aws:forward_to_sfe"], "StatefulRuleGroupReferences": [{ "ResourceArn": "<RuleGroupArn>" }] }``` Enter the following command in the AWS CLI. This command requires an [update token](https://docs.aws.amazon.com/network-firewall/latest/developerguide/resource-managing.html#nwfw-resource-updating) to add the new rules. The token is used to confirm that the policy hasn't changed since you last retrieved it.	AWS systems administrator

Task	Description	Skills required
	<pre>UPDATETOKEN=(`aws  network-firewall  describe-firewall- policy \               -- firewall-policy-name  firewall-policy-anfw  \               --output  text --query UpdateTok en`) aws network-firewall update-firewall-po licy \ --update-token $UPDATETOKEN \ --firewall-policy- name firewall-policy- anfw \ --firewall-policy file://firewall-po licy-anfw.json</pre>	

Task	Description	Skills required

Confirm the policy updates.	(Optional) If you would like to confirm the rules were added and view the policy format, enter the following command in the AWS CLI.
	`# aws network-firewall describe-firewall-policy \ --firewall-policy-name firewall-policy-anfw \ --query FirewallPolicy`
	The following is an example output.
	`{ "StatelessDefaultActions": [ "aws:forward_to_sfe" ], "StatelessFragmentDefaultActions": [ "aws:forward_to_sfe" ], "StatefulRuleGroupReferences": [ { "ResourceArn": "arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom" }`
		AWS systems administrator

Task	Description	Skills required

	``
	    ]
	}
	``	

Test alert functionality

Task	Description	Skills required

Generate alerts for testing.	1. Log in to a test workstation within the firewall subnet.
	2. Enter commands that should generate alerts. For example, you can use `wget` or `curl`.
	`wget -U "badstuff" http://www.amazon.com -o /dev/null`
	`curl -A "morebadstuff" http://www.amazon.com -o /dev/null`
		AWS systems administrator

Validate that the alerts are logged.	1. Open the [CloudWatch console](#). 2. Navigate to the correct log group and stream. For more information, see [View log data sent to CloudWatch Logs](#) (CloudWatch Logs documentation). 3. Confirm the logged events are similar to the following examples. The examples	AWS systems administrator

Task	Description	Skills required

	show only the relevant portion of the alert.
	
	Example 1
	
	`"alert": {`
	`"action": "allowed",`
	`"signature_id": 3,`
	`"rev": 1,`
	`"signature": "",`
	`"category": "Misc activity",`
	`"severity": 3`
	`}`
	
	Example 2
	
	`"alert": {`
	`"action": "allowed",`
	`"signature_id": 4,`
	`"rev": 1,`
	`"signature": "",`
	`"category": "Potentially Bad Traffic",`
	`"severity": 2`
	`}`	

Update the firewall rules and rule group

Task	Description	Skills required
Update the firewall rules.	1. In a text editor, open the `custom.rules` file. 2. Change the first rule to be similar to the following. This rule must be entered on a single line in the file. `alert http any any -> any any (msg:"Watch out - Bad Stuff!!"; content:"badstuff"; classtype:misc-activity; priority: 2; sid:3; rev:2; metadata:custom-field-2 Danger!, custom-field More Info;)` This makes the following changes to the rule: • Adds a msg (Suricata website) string that provides text information about the signature or alert. In the generated alert, this maps to the signature. • Adjusts the default priority (Suricata website) of `misc-activity` from 3 to 2. For the default values of the various `classtype`	AWS systems administrator

Task	Description	Skills required
	s , see the Additional information section. • Adds custom metadata (Suricata website) to the alert. This is additional information that that is added to the signature. It is recommended that you use key-value pairs. • Changes the rev (Suricata website) from 1 to 2. This represents the version of the signature.	

Task	Description	Skills required

Update the rule group.	In the AWS CLI, run the following commands. Use the ARN of your firewall policy. These commands get an update token and update the rule group with the rule changes.
	`# UPDATETOKEN=(`aws network-firewall \ describe-rule-group \ --rule-group-arn arn:aws:network-firewall:us-east-2:123457890:stateful-rulegroup/custom \ --output text --query UpdateToken`)`
	`# aws network-firewall update-rule-group \ --rule-group-arn arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom \ --rules file://custom.rules \ --update-token $UPDATETOKEN`
	The following is an example output.
	`{`
		AWS systems administrator

Task	Description	Skills required
	"UpdateToken": "7536939f-6a1d-414c-96d1-bb28110996ed", "RuleGroupResponse": { "RuleGroupArn": "arn:aws:network-firewall:us-east-2:1234567890:stateful-rulegroup/custom", "RuleGroupName": "custom", "RuleGroupId": "238a8259-9eaf-48bb-90af-5e690cf8c48b", "Type": "STATEFUL", "Capacity": 10, "RuleGroupStatus": "ACTIVE", "Tags": [{ "Key": "environment", "Value": "development" }] } }	

Test the updated alert functionality

Task	Description	Skills required
Generate an alert for testing.	1. Log in to a test workstation within the firewall subnet. 2. Enter a command that should generate an alert.	AWS systems administrator

Task	Description	Skills required
	For example, you can use `curl`. `curl -A "badstuff" http://www.amazon.com -o /dev/null`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Validate the alert changed.</td><td>

1. Open the [CloudWatch console](https://console.aws.amazon.com/cloudwatch/).
2. Navigate to the correct log group and stream.
3. Confirm the logged event is similar to the following example. The example shows only the relevant portion of the alert.

```
"alert": {
    "action":
 "allowed",
    "signature_id":
 3,
    "rev": 2,
    "signature":
 "Watch out - Bad
 Stuff!!",
    "category": "Misc
 activity",
    "severity": 2,
    "metadata": {
        "custom-f
ield": [
            "More
 Info"
        ],
        "custom-f
ield-2": [
            "Danger!"
        ]
    }
}
```

</td><td>AWS systems administrator</td></tr>
</table>

Related resources

References

- Send alerts from AWS Network Firewall to a Slack channel (AWS Prescriptive Guidance)
- Scaling threat prevention on AWS with Suricata (AWS blog post)
- Deployment models for AWS Network Firewall (AWS blog post)
- Suricata meta keyworks (Suricata documentation)

Tutorials and videos

- AWS Network Firewall workshop

Additional information

The following is the classification configuration file from Suricata 5.0.2. These classifications are used when creating the firewall rules.

```
# config classification:shortname,short description,priority

config classification: not-suspicious,Not Suspicious Traffic,3
config classification: unknown,Unknown Traffic,3
config classification: bad-unknown,Potentially Bad Traffic, 2
config classification: attempted-recon,Attempted Information Leak,2
config classification: successful-recon-limited,Information Leak,2
config classification: successful-recon-largescale,Large Scale Information Leak,2
config classification: attempted-dos,Attempted Denial of Service,2
config classification: successful-dos,Denial of Service,2
config classification: attempted-user,Attempted User Privilege Gain,1
config classification: unsuccessful-user,Unsuccessful User Privilege Gain,1
config classification: successful-user,Successful User Privilege Gain,1
config classification: attempted-admin,Attempted Administrator Privilege Gain,1
config classification: successful-admin,Successful Administrator Privilege Gain,1

# NEW CLASSIFICATIONS
config classification: rpc-portmap-decode,Decode of an RPC Query,2
config classification: shellcode-detect,Executable code was detected,1
config classification: string-detect,A suspicious string was detected,3
config classification: suspicious-filename-detect,A suspicious filename was detected,2
```

```
config classification: suspicious-login,An attempted login using a suspicious username
 was detected,2
config classification: system-call-detect,A system call was detected,2
config classification: tcp-connection,A TCP connection was detected,4
config classification: trojan-activity,A Network Trojan was detected, 1
config classification: unusual-client-port-connection,A client was using an unusual
 port,2
config classification: network-scan,Detection of a Network Scan,3
config classification: denial-of-service,Detection of a Denial of Service Attack,2
config classification: non-standard-protocol,Detection of a non-standard protocol or
 event,2
config classification: protocol-command-decode,Generic Protocol Command Decode,3
config classification: web-application-activity,access to a potentially vulnerable web
 application,2
config classification: web-application-attack,Web Application Attack,1
config classification: misc-activity,Misc activity,3
config classification: misc-attack,Misc Attack,2
config classification: icmp-event,Generic ICMP event,3
config classification: inappropriate-content,Inappropriate Content was Detected,1
config classification: policy-violation,Potential Corporate Privacy Violation,1
config classification: default-login-attempt,Attempt to login by a default username and
 password,2

# Update
config classification: targeted-activity,Targeted Malicious Activity was Detected,1
config classification: exploit-kit,Exploit Kit Activity Detected,1
config classification: external-ip-check,Device Retrieving External IP Address
 Detected,2
config classification: domain-c2,Domain Observed Used for C2 Detected,1
config classification: pup-activity,Possibly Unwanted Program Detected,2
config classification: credential-theft,Successful Credential Theft Detected,1
config classification: social-engineering,Possible Social Engineering Attempted,2
config classification: coin-mining,Crypto Currency Mining Activity Detected,2
config classification: command-and-control,Malware Command and Control Activity
 Detected,1
```

Deploy resources in an AWS Wavelength Zone by using Terraform

Zahoor Chaudhrey and Luca Iannario, Amazon Web Services

Summary

AWS Wavelength helps you build infrastructure that is optimized for Multi-Access Edge Computing (MEC) applications. *Wavelength Zones* are AWS infrastructure deployments that embed AWS compute and storage services within communications service providers' (CSP) 5G networks. Application traffic from 5G devices reaches application servers running in Wavelength Zones without leaving the telecommunications network. The following facilitate network connectivity through Wavelength:

- **Virtual private clouds (VPCs) –** VPCs in an AWS account can extend to span multiple Availability Zones, including Wavelength Zones. Amazon Elastic Compute Cloud (Amazon EC2) instances and related services appear as part of your Regional VPC. VPCs are created and managed in Amazon Virtual Private Cloud (Amazon VPC).
- **Carrier gateway –** A carrier gateway enables connectivity from the subnet in the Wavelength Zone to the CSP network, the internet, or the AWS Region through the CSP's network. The carrier gateway serves two purposes. It allows inbound traffic from a CSP network in a specific location, and it allows outbound traffic to the telecommunications network and the internet.

This pattern and its associated Terraform code help you launch resources, such as Amazon EC2 instances, Amazon Elastic Block Store (Amazon EBS) volumes, VPCs, subnets, and a carrier gateway, in a Wavelength Zone.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An integrated development environment (IDE)
- Opt in to the target Wavelength Zone
- AWS Command Line Interface (AWS CLI), installed and configured
- Terraform version 1.8.4 or later, installed (Terraform documentation)
- Terraform AWS Provider version 5.32.1 or later, configured (Terraform documentation)
- Git, installed (GitHub)

- [Permissions](#) to create Amazon VPC, Wavelength, and Amazon EC2 resources

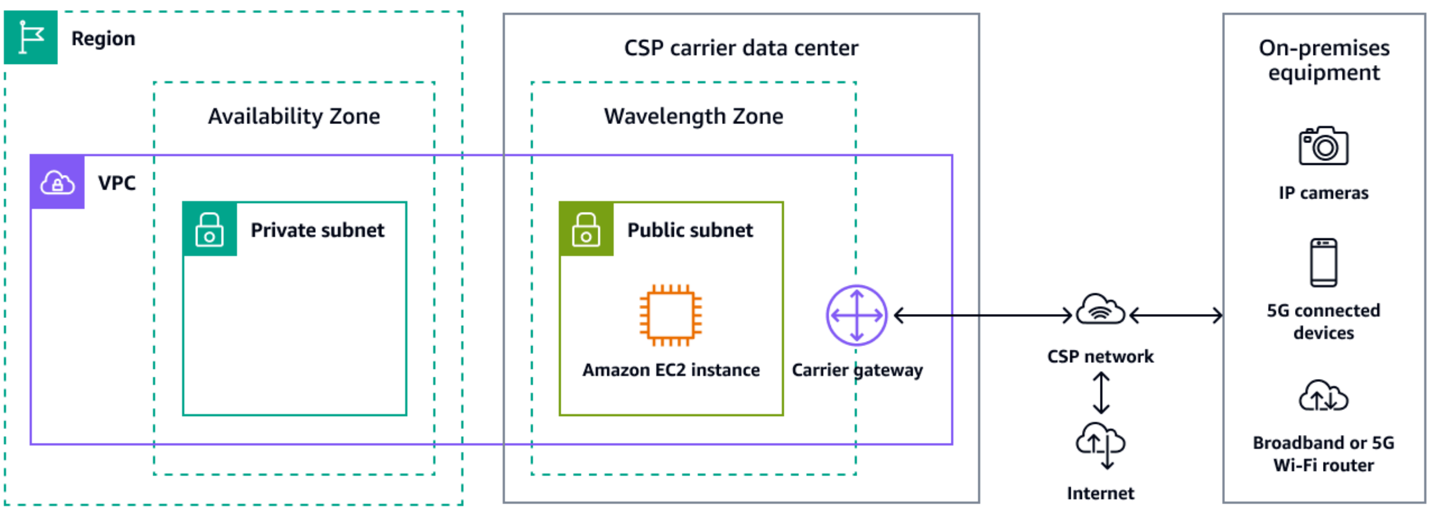
Limitations

Not all AWS Regions support Wavelength Zones. For more information, see Available Wavelength Zones in the Wavelength documentation.

Architecture

The following diagram shows how you can create a subnet and AWS resources in a Wavelength Zone. VPCs that contain a subnet in a Wavelength Zone can connect to a carrier gateway. A carrier gateway allows you to connect to the following resources:

- 4G/LTE and 5G devices on the telecommunication carrier's network.
- Fixed wireless access for select Wavelength Zone partners. For more information, see Multi-access AWS Wavelength.
- Outbound traffic to public internet resources.



Tools

AWS services

- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

- AWS Wavelength extends AWS Cloud infrastructure to telecommunication providers' 5G networks. This helps you build applications that deliver ultra-low latencies to mobile devices and end users.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub Creating AWS Wavelength Infrastructure using Terraform repository. The Terraform code deploys the following infrastructure and resources:

- A VPC
- A Wavelength Zone
- A pubic subnet in the Wavelength Zone
- A carrier gateway in the Wavelength Zone
- An Amazon EC2 instance in the Wavelength Zone

Best practices

- Before deploying, confirm that you're using the latest versions of Terraform and the AWS CLI.
- Use a continuous integration and continuous delivery (CI/CD) pipeline to deploy IaC. For more information, see Best practices for managing Terraform State files in AWS CI/CD Pipeline on AWS Blogs.

Epics

Provision the infrastructure

Task	Description	Skills required
Clone the repository.	Enter the following command to clone the Creating AWS Wavelength Infrastructure	DevOps engineer

Task	Description	Skills required
	using Terraform repository to your environment. `git clone git@github.com:aws-samples/terraform-wavelength-infrastructure.git`	

Task	Description	Skills required
Update the variables.	1. Navigate to the cloned repository. `cd terraform-wavelength-infrastructure` 2. Using any text editor, create a file called **terraform.tfvars** in the root directory. 3. Create the following variables and enter their values: • `region = <enter Region name>` • `vpc_cidr = <enter CIDR block used by VPC>` • `wavelength_subnet_cidr = <enter CIDR block for the subnet in the Wavelength Zone>` • `availabilityzone_wavelength = <enter Wavelength Zone name>` 4. Save the **terraform.tfvars** file.	DevOps engineer, Terraform

Task	Description	Skills required
Initialize the configuration.	Enter the following command to initialize the working directory. `terraform init`	DevOps engineer, Terraform
Preview the Terraform plan.	Enter the following command to compare the target state against the current state of your AWS environment. This command generates a preview of the resources that will be configured. `terraform plan`	DevOps engineer, Terraform

Task	Description	Skills required
Verify and deploy.	1. Review the configuration changes in the Terraform plan and confirm that you want to implement these changes. 2. Enter the following command to apply the plan and create the infrastructure. `terraform apply` 3. Enter `yes` to proceed. Terraform creates the architecture that is declared in the configura tion files. For more information about the architecture, see the Target architecture section of this pattern.	DevOps engineer, Terraform

Validate and clean up

Task	Description	Skills required
Verify the infrastructure deployment.	1. If you don't already have an Amazon EC2 instance in a public subnet in your AWS Region, create one. For instructions, see Launch your Linux instance or Launch your Windows instance. You will use	AWS DevOps, DevOps engineer

Task	Description	Skills required
	this instance to test the connectivity from the AWS Region to the Wavelength Zone. 2. Test the connectivity from the instance in the AWS Region to the instance in the Wavelength Zone. For instructions, see [Test the connectivity](#) in the Wavelength documentation.	
(Optional) Clean up the infrastructure.	If you need to delete all of the resources that were provisioned by Terraform, do the following: 1. Enter the following command. `terraform destroy` 2. Enter `yes` to confirm.	DevOps engineer, Terraform

Troubleshooting

Issue	Solution
Connectivity to Amazon EC2 instances in the AWS Region.	See Troubleshoot connecting to your Linux instance or Troubleshoot connecting to your Windows instance.

Issue	Solution
Connectivity to Amazon EC2 instances in the Wavelength Zone.	See Troubleshoot SSH or RDP connectivity to my EC2 instances launched in a Wavelength Zone.
Capacity in the Wavelength Zone.	See Quotas and considerations for Wavelength Zones.
Mobile or carrier connectivity from the carrier network to the AWS Region.	1. Verify that the carrier gateway is operational. Do the following: a. Open the Amazon VPC console. b. In the navigation pane, choose **Your VPCs**. c. Select the VPC that contains the Wavelength Zone. d. On the **Details** pane, for **Carrier gateway**, confirm that the value is attached. 2. Verify that any elastic IP addresses attached to instances in the Wavelength Zone are operational. Do the following: a. Open the Amazon EC2 console. b. In the navigation pane, choose **Instances**. c. Select the instance in the Wavelength Zone. d. Choose the **Network** tab. e. Confirm that the elastic network interface has an Elastic IP address attached. 3. Contact the carrier network support team.

Related resources

- What is AWS Wavelength?
- How AWS Wavelength works
- Resilience in AWS Wavelength

Migrate DNS records in bulk to an Amazon Route 53 private hosted zone

Ram Kandaswamy, Amazon Web Services

Summary

Network engineers and cloud administrators need an efficient and simple way to add Domain Name System (DNS) records to private hosted zones in Amazon Route 53. Using a manual approach to copy entries from a Microsoft Excel worksheet to appropriate locations in the Route 53 console is tedious and error prone. This pattern describes an automated approach that reduces the time and effort required to add multiple records. It also provides a repeatable set of steps for multiple hosted zone creation.

This pattern uses Amazon Simple Storage Service (Amazon S3) to store records. To work with data efficiently, the pattern uses the JSON format because of its simplicity and its ability to support a Python dictionary (`dict` data type).

> **Note**
>
> If you can generate a zone file from your system, consider using the Route 53 import feature instead.

Prerequisites and limitations

Prerequisites

- An Excel worksheet that contains private hosted zone records
- Familiarity with different types of DNS records such as A record, Name Authority Pointer (NAPTR) record, and SRV record (see Supported DNS record types)
- Familiarity with the Python language and its libraries

Limitations

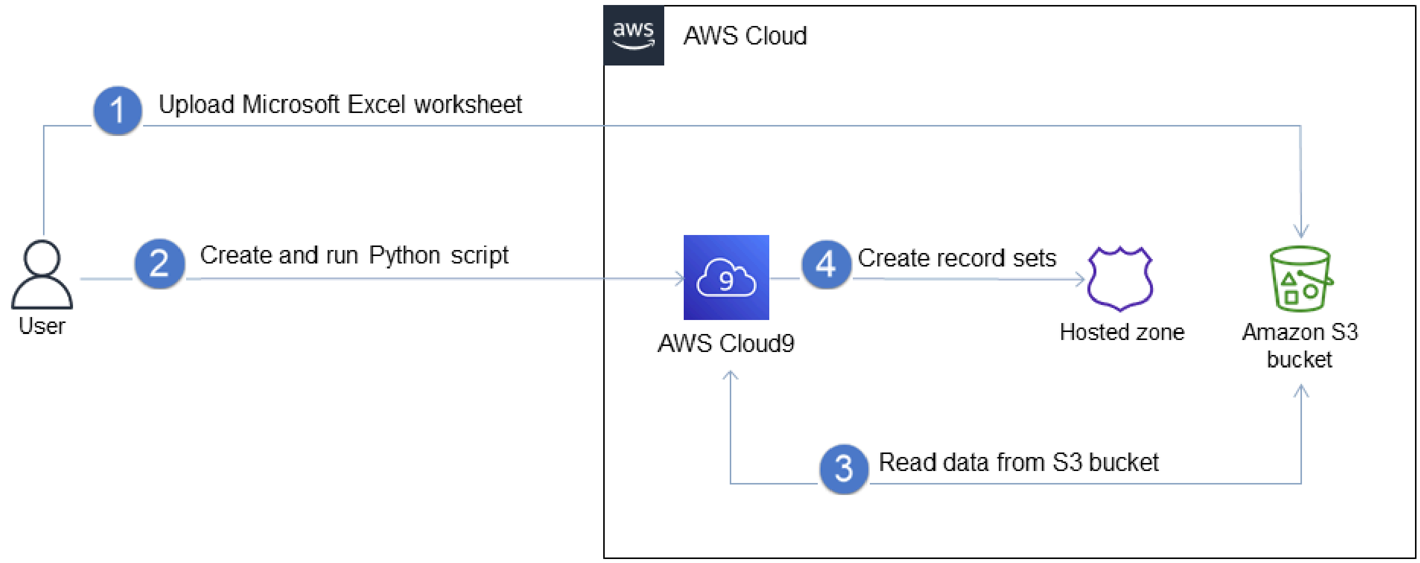
- The pattern doesn't provide extensive coverage for all use case scenarios. For example, the change_resource_record_sets call doesn't use all the available properties of the API.

- In the Excel worksheet, the value in each row is assumed to be unique. Multiple values for each fully qualified domain name (FQDN) are expected to appear in the same row. If that is not true, you should modify the code provided in this pattern to perform the necessary concatenation.
- The pattern uses the AWS SDK for Python (Boto3) to call the Route 53 service directly. You can enhance the code to use an AWS CloudFormation wrapper for the `create_stack` and `update_stack` commands, and use the JSON values to populate template resources.

Architecture

Technology stack

- Route 53 private hosted zones for routing traffic
- Amazon S3 for storing the output JSON file



The workflow consists of these steps, as illustrated in the previous diagram and discussed in the *Epics* section:

1. Upload an Excel worksheet that has the record set information to an S3 bucket.
2. Create and run a Python script that converts the Excel data to JSON format.
3. Read the records from the S3 bucket and clean the data.
4. Create record sets in your private hosted zone.

Tools

- [Route 53](#) – Amazon Route 53 is a highly available and scalable DNS web service that handles domain registration, DNS routing, and health checking.
- [Amazon S3](#) – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.

Epics

Prepare data for automation

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create an Excel file for your records.</td><td>Use the records you exported from your current system to create an Excel worksheet that has the required columns for a record, such as fully qualified domain name (FQDN), record type, Time to Live (TTL), and value. For NAPTR and SRV records, the value is a combination of multiple properties, so use Excel's <code>concat</code> method to combine these properties.
<table><tr><th>FqdnN</th><th>Recorc
e</th><th>Value</th><th>TTL</th></tr><tr><td>somet
.exam
org</td><td>A</td><td>1.1.1.</td><td>900</td></tr></table></td><td>Data engineer, Excel skills</td></tr>
<tr><td>Verify the working environme nt.</td><td>In your IDE, create a Python file to convert the Excel</td><td>General AWS</td></tr>
</table>

Task	Description	Skills required

	input worksheet to JSON format. (Instead of an IDE, you can also use an Amazon SageMaker notebook to work with Python code.)
	
	Verify that the Python version you're using is version 3.7 or later.
	
	`python3 --version`
	
	Install the **pandas** package.
	
	`pip3 install pandas --user`	

Convert the Excel worksheet data to JSON.	Create a Python file that contains the following code to convert from Excel to JSON.
	
	`import pandas as pd`
	`data=pd.read_excel('./Book1.xls')`
	`data.to_json(path_or_buf='my.json', orient='records')`
	
	where Book1 is the name of the Excel worksheet and `my.json` is the name of the output JSON file.
		Data engineer, Python skills

Task	Description	Skills required
Upload the JSON file to an S3 bucket.	Upload the `my.json` file to an S3 bucket. For more information, see [Creating a bucket](#) in the Amazon S3 documentation.	App developer

Insert records

Task	Description	Skills required

Create a private hosted zone.	Use the [create_hosted_zone](#) API and the following Python sample code to create a private hosted zone. Replace the parameters `hostedZoneName`, `vpcRegion`, and `vpcId` with your own values.
	`import boto3`
	`import random`
	`hostedZoneName ="xxx"`
	`vpcRegion = "us-east-1"`
	`vpcId="vpc-xxxx"`
	`route53_client = boto3.client('route53')`
	`response = route53_client.create_hosted_zone(`
	`        Name= hostedZoneName,`
	`        VPC={`
	`            'VPCRegion: vpcRegion,`
	`            'VPCId': vpcId`
		Cloud architect, Network administrator, Python skills

Task	Description	Skills required

	``
	        },
	        CallerRef
	erence=str(random.
	random()*100000),
	        HostedZon
	eConfig={
	            'Comment'
	: "private hosted zone
	 created by automatio
	n",
	            'PrivateZ
	one': True
	        }
	    )
	 print(response)
	``
	You can also use an infrastructure as code (IaC) tool such as AWS CloudFormation to replace these steps with a template that creates a stack with the appropriate resources and properties.	

Task	Description	Skills required

Retrieve details as a dictionary from Amazon S3.	Use the following code to read from the S3 bucket and to get the JSON values as a Python dictionary.
	`fileobj = s3_client.get_object(`
	`        Bucket=bucket_name,`
	`        Key='my.json'`
	`        )`
	`    filedata = fileobj['Body'].read()`
	`    contents = filedata.decode('utf-8')`
	`    json_content=json.loads(contents)`
	`    print(json_content)`
	where `json_content` contains the Python dictionary.
		App developer, Python skills

Task	Description	Skills required

Clean data values for spaces and Unicode characters.	As a safety measure to ensure the correctness of data, use the following code to perform a strip operation on the values in `json_content`. This code removes the space characters at the front and end of each string. It also uses the `replace` method to remove hard (non-breaking) spaces (the `\xa0` characters).
	
	`for item in json_content:`
	`    fqn_name = unicodedata.normalize("NFKD",item["FqdnName"].replace("u'", "'").replace('\xa0', '').strip())`
	`    rec_type = item["RecordType"].replace('\xa0', '').strip()`
	`    res_rec = {`
	`      'Value': item["Value"].replace('\xa0', '').strip()`
	`                }`
		App developer, Python skills

Task	Description	Skills required
Insert records.	Use the following code as part of the previous `for` loop. `change_response = route53_client.change_resource_record_sets( HostedZoneId="xxxxxxxx", ChangeBatch={ 'Comment': 'Created by automation', 'Changes': [ { 'Action': 'UPSERT', 'ResourceRecordSet': { 'Name': fqn_name, 'Type': rec_type, 'TTL': item["TTL"], 'ResourceRecords': res_rec } } ] } )`	App developer, Python skills

Task	Description	Skills required
	Where `xxxxxxx` is the hosted zone ID from the first step of this epic.	

Related resources

References

- Creating records by importing a zone file (Amazon Route 53 documentation)
- create_hosted_zone method (Boto3 documentation)
- change_resource_record_sets method (Boto3 documentation)

Tutorials and videos

- The Python Tutorial (Python documentation)
- DNS design using Amazon Route 53 (YouTube video, *AWS Online Tech Talks*)

Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS

Sachin Trivedi, Amazon Web Services

Summary

When you migrate an application that uses an F5 Load balancer to Amazon Web Services (AWS) and want to use an Application Load Balancer on AWS, migrating F5 rules for header modifications is a common problem. An Application Load Balancer doesn't support header modifications, but you can use Amazon CloudFront as a content delivery network (CDN) and Lambda@Edge to modify headers.

This pattern describes the required integrations and provides sample code for header modification by using AWS CloudFront and Lambda@Edge.

Prerequisites and limitations

Prerequisites

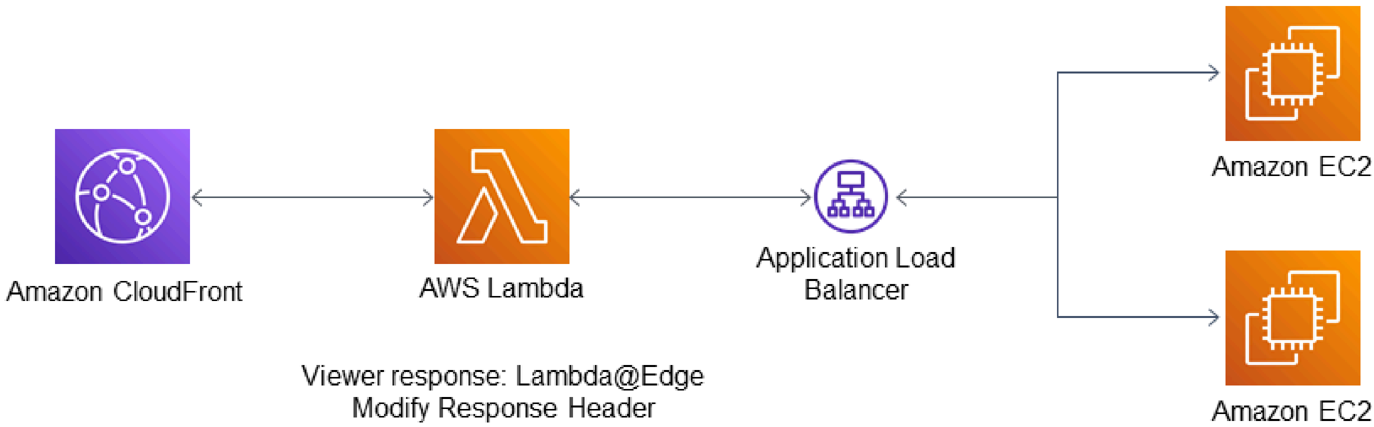
- An on-premises application that uses an F5 load balancer with a configuration that replaces the HTTP header value by using `if, else`. For more information about this configuration, see [HTTP::header](#) in the F5 product documentation.

Limitations

- This pattern applies to F5 load balancer header customization. For other third-party load balancers, please check the load balancer documentation for support information.
- The Lambda functions that you use for Lambda@Edge must be in the US East (N. Virginia) Region.

Architecture

The following diagram shows the architecture on AWS, including the integration flow between the CDN and other AWS components.



Tools

AWS services

- [Application Load Balancer](#) — An Application Load Balancer is an AWS fully managed load balancing service that functions at the seventh layer of the Open Systems Interconnection (OSI) model. It balances traffic across multiple targets and supports advanced routing requests based on HTTP headers and methods, query strings, and host-based or path-based routing.
- [Amazon CloudFront](#) – Amazon CloudFront is a web service that speeds up the distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations for lower latency and improved performance.
- [Lambda@Edge](#) — Lambda@Edge is an extension of AWS Lambda that lets you run functions to customize the content that CloudFront delivers. You can author functions in the US East (N. Virginia) Region, and then associate the function with a CloudFront distribution to automatically replicate your code around the world, without provisioning or managing servers. This reduces latency and improves the user experience.

Code

The following sample code provides a blueprint for modifying CloudFront response headers. Follow the instructions in the *Epics* section to deploy the code.

```
exports.handler = async (event, context) => {
    const response = event.Records[0].cf.response;
```

```
    const headers = response.headers;


    const headerNameSrc = 'content-security-policy';
    const headerNameValue = '*.xyz.com';


    if (headers[headerNameSrc.toLowerCase()]) {
        headers[headerNameSrc.toLowerCase()] = [{
            key: headerNameSrc,
            value: headerNameValue,
        }];
        console.log(`Response header "${headerNameSrc}" was set to ` +
                    `"${headers[headerNameSrc.toLowerCase()][0].value}"`);
    }
    else {
            headers[headerNameSrc.toLowerCase()] = [{
            key: headerNameSrc,
            value: headerNameValue,
            }];
    }
    return response;
};
```

Epics

Create a CDN distribution

Task	Description	Skills required
Create a CloudFront web distribution.	In this step, you create a CloudFront distribution to tell CloudFront where you want content to be delivered from, and the details about how to track and manage content delivery. To create a distribution by using the console, sign in to the AWS Management	Cloud administrator

Task	Description	Skills required
	Console, open the CloudFront console, and then follow the steps in the CloudFront documentation.	

Create and deploy the Lambda@Edge function

Task	Description	Skills required
Create and deploy a Lambda@Edge function.	You can create a Lambda@Edge function by using a blueprint for modifying CloudFront response headers. (Other bluePrints are available for different use cases; for more information, see Lambda@Edge example functions in the CloudFront documentation.) To create a Lambda@Edge function: 1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/. 2. Make sure that you're in the US East (N. Virginia) Region. CloudFront blueprints are available only in this Region. 3. Choose **Create function**.	AWS administrator

Task	Description	Skills required
	4. Choose **Use a blueprint**, and then enter **cloudfront** in the **Blueprints** search field. 5. Choose the **cloudfront-modify-response-header** blueprint, and then choose **Configure**. 6. On the **Basic information** page, enter the following information: a. Enter a function name. b. For **Execution role**, choose **Create a new role from AWS policy templates**. c. Associate the required AWS Identity and Access Management (IAM) role name. 7. Choose **Create function**. 8. In the **Designer** section of the page, choose your function name. 9. In the **Function code** section, replace the template code with the sample code provided previously in this pattern, in the *Code* section. 10. In the sample code, replace `xyz.com` with your domain name.	

Task	Description	Skills required
	11. Choose **Save**.	
Deploy the Lambda@Edge function.	Follow the instructions in step 4 of the *Tutorial: Creating a simple Lambda@Edge function* in the Amazon CloudFront documentation to configure the CloudFront trigger and deploy the function.	AWS administrator

Related resources

CloudFront documentation

- Request and response behavior for custom origins
- Working with distributions
- Lambda@Edge example functions
- Customizing at the edge with Lambda@Edge
- Tutorial: Creating a simple Lambda@Edge function

Create a report of Network Access Analyzer findings for inbound internet access in multiple AWS accounts

Mike Virgilio, Amazon Web Services

Summary

Unintentional inbound internet access to AWS resources can pose risks to an organization's data perimeter. Network Access Analyzer is an Amazon Virtual Private Cloud (Amazon VPC) feature that helps you identify unintended network access to your resources on Amazon Web Services (AWS). You can use Network Access Analyzer to specify your network access requirements and to identify potential network paths that do not meet your specified requirements. You can use Network Access Analyzer to do the following:

1. Identify AWS resources that are accessible to the internet through internet gateways.
2. Validate that your virtual private clouds (VPCs) are appropriately segmented, such as isolating production and development environments and separating transactional workloads.

Network Access Analyzer analyzes end-to-end network reachability conditions and not just a single component. To determine whether a resource is internet accessible, Network Access Analyzer evaluates the internet gateway, VPC route tables, network access control lists (ACLs), public IP addresses on elastic network interfaces, and security groups. If any of these components prevent internet access, Network Access Analyzer doesn't generate a finding. For example, if an Amazon Elastic Compute Cloud (Amazon EC2) instance has an open security group that allows traffic from `0/0` but the instance is in a private subnet that isn't routable from any internet gateway, then Network Access Analyzer wouldn't generate a finding. This provides high-fidelity results so that you can identify resources that are truly accessible from the internet.

When you run Network Access Analyzer, you use Network Access Scopes to specify your network access requirements. This solution identifies network paths between an internet gateway and an elastic network interface. In this pattern, you deploy the solution in a centralized AWS account in your organization, managed by AWS Organizations, and it analyzes all of the accounts, in any AWS Region, in the organization.

This solution was designed with the following in mind:

- The AWS CloudFormation templates reduce the effort required to deploy the AWS resources in this pattern.

- You can adjust the parameters in the CloudFormation templates and **naa-script.sh** script at the time of deployment to customize them for your environment.
- Bash scripting automatically provisions and analyzes the Network Access Scopes for multiple accounts, in parallel.
- A Python script processes the findings, extracts the data, and then consolidates the results. You can choose to review the consolidated report of Network Access Analyzer findings in CSV format or in AWS Security Hub. An example of the CSV report is available in the Additional information section of this pattern.
- You can remediate findings, or you can exclude them from future analyses by adding them to the **naa-exclusions.csv** file.

Prerequisites and limitations

Prerequisites

- An AWS account for hosting security services and tools, managed as a member account of an organization in AWS Organizations. In this pattern, this account is referred to as the security account.
- In the security account, you must have a private subnet with outbound internet access. For instructions, see Create a subnet in the Amazon VPC documentation. You can establish internet access by using an NAT gateway or an interface VPC endpoint.
- Access to the AWS Organizations management account or an account that has delegated administrator permissions for CloudFormation. For instructions, see Register a delegated administrator in the CloudFormation documentation.
- Enable trusted access between AWS Organizations and CloudFormation. For instructions, see Enable trusted access with AWS Organizations in the CloudFormation documentation.
- If you're uploading the findings to Security Hub, Security Hub must be enabled in the account and AWS Region where the Amazon EC2 instance is provisioned. For more information, see Setting up AWS Security Hub.

Limitations

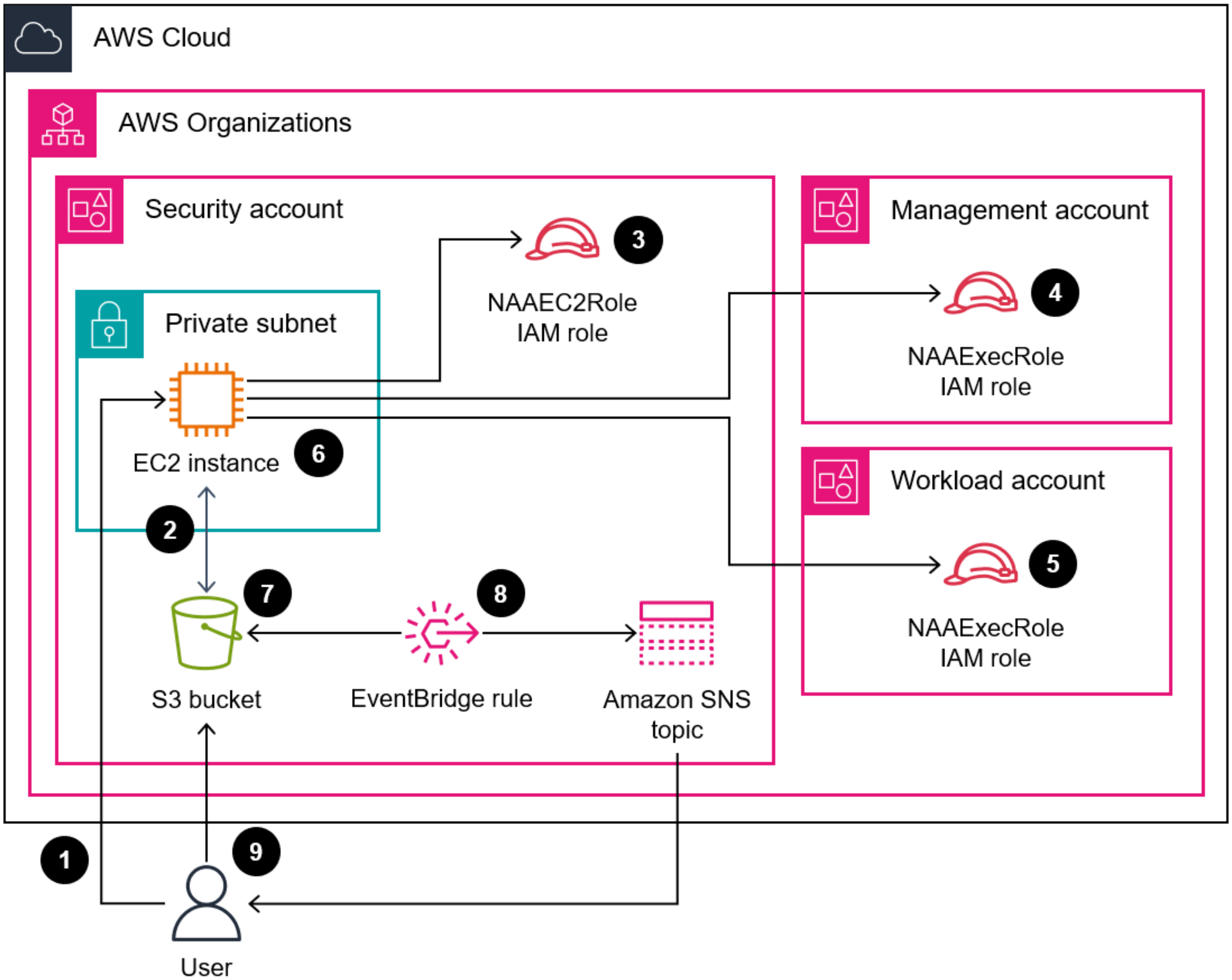
- Cross-account network paths are not currently analyzed due to limitations of the Network Access Analyzer feature.

- The target AWS accounts must be managed as an organization in AWS Organizations. If you are not using AWS Organizations, you can update the **naa-execrole.yaml** CloudFormation template and the **naa-script.sh** script for your environment. Instead, you provide a list of AWS account IDs and Regions where you want to run the script.
- The CloudFormation template is designed to deploy the Amazon EC2 instance in a private subnet that has outbound internet access. The AWS Systems Manager Agent (SSM Agent) requires outbound access to reach the Systems Manager service endpoint, and you need outbound access to clone the code repository and install dependencies. If you want to use a public subnet, you must modify the **naa-resources.yaml** template to associate an Elastic IP address with the Amazon EC2 instance.

Architecture

Target architecture

Option 1: Access findings in an Amazon S3 bucket



The diagram shows the following process:

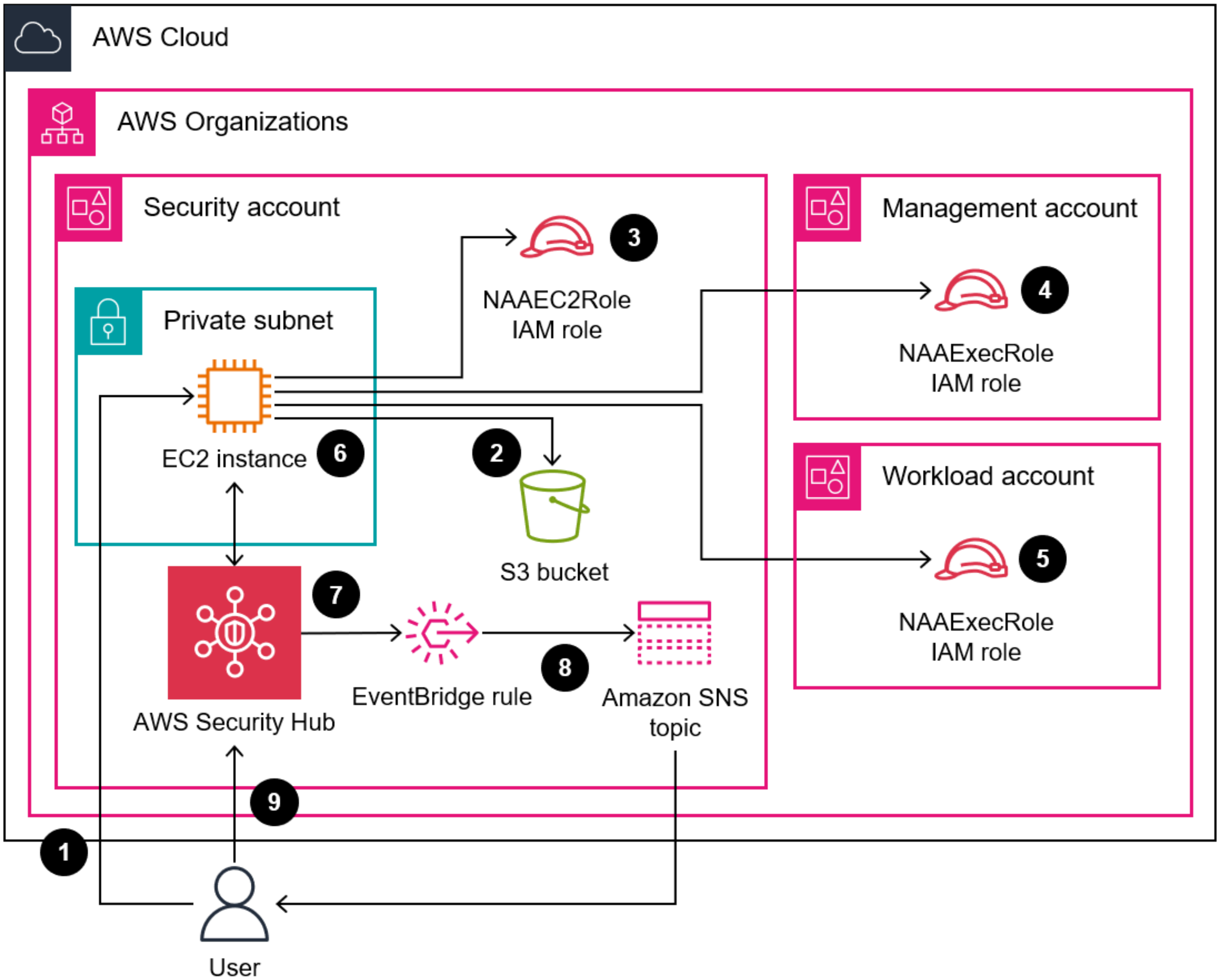
1. If you're manually running the solution, the user authenticates to the Amazon EC2 instance by using Session Manager and then runs the **naa-script.sh** script. This shell script performs steps 2–7.

   If you're automatically running the solution, the **naa-script.sh** script starts automatically on the schedule you defined in the cron expression. This shell script performs steps 2–7. For more information, see *Automation and scale* at the end of this section.

2. The Amazon EC2 instance downloads the latest **naa-exception.csv** file from the Amazon S3 bucket. This file is used later in the process when the Python script processes the exclusions.

3. The Amazon EC2 instance assumes the `NAAEC2Role` AWS Identity and Access Management (IAM) role, which grants permissions to access the Amazon S3 bucket and to assume the `NAAExecRole` IAM roles in the other accounts in the organization.
4. The Amazon EC2 instance assumes the `NAAExecRole` IAM role in the organization's management account and generates a list of the accounts in the organization.
5. The Amazon EC2 instance assumes the `NAAExecRole` IAM role in the organization's member accounts (called *workload accounts* in the architecture diagram) and performs a security assessment in each account. The findings are stored as JSON files on the Amazon EC2 instance.
6. The Amazon EC2 instance uses a Python script to process the JSON files, extract the data fields, and create a CSV report.
7. The Amazon EC2 instance uploads the CSV file to the Amazon S3 bucket.
8. An Amazon EventBridge rule detects the file upload and uses an Amazon SNS topic to send an email that notifies the user that the report is complete.
9. The user downloads the CSV file from the Amazon S3 bucket. The user imports the results into the Excel template and reviews the results.

Option 2: Access findings in AWS Security Hub



The diagram shows the following process:

1. If you're manually running the solution, the user authenticates to the Amazon EC2 instance by using Session Manager and then runs the **naa-script.sh** script. This shell script performs steps 2–7.

   If you're automatically running the solution, the **naa-script.sh** script starts automatically on the schedule you defined in the cron expression. This shell script performs steps 2–7. For more information, see *Automation and scale* at the end of this section.

2. The Amazon EC2 instance downloads the latest **naa-exception.csv** file from the Amazon S3 bucket. This file is used later in the process when the Python script processes the exclusions.

3. The Amazon EC2 instance assumes the `NAAEC2Role` IAM role, which grants permissions to access the Amazon S3 bucket and to assume the `NAAExecRole` IAM roles in the other accounts in the organization.
4. The Amazon EC2 instance assumes the `NAAExecRole` IAM role in the organization's management account and generates a list of the accounts in the organization.
5. The Amazon EC2 instance assumes the `NAAExecRole` IAM role in the organization's member accounts (called *workload accounts* in the architecture diagram) and performs a security assessment in each account. The findings are stored as JSON files on the Amazon EC2 instance.
6. The Amazon EC2 instance uses a Python script to process the JSON files and extract the data fields for import into Security Hub.
7. The Amazon EC2 instance imports the Network Access Analyzer findings to Security Hub.
8. An Amazon EventBridge rule detects the import and uses an Amazon SNS topic to send an email that notifies the user that the process is complete.
9. The user views the findings in Security Hub.

Automation and scale

You can schedule this solution to run the **naa-script.sh** script automatically on a custom schedule. To set a custom schedule, in the **naa-resources.yaml** CloudFormation template, modify the `CronScheduleExpression` parameter. For example, the default value of `0 0 * * 0` runs the solution at midnight on every Sunday. A value of `0 0 * 1-12 0` would run the solution at midnight on the first Sunday of every month. For more information about using cron expressions, see [Cron and rate expressions](#) in the Systems Manager documentation.

If you want adjust the schedule after the `NAA-Resources` stack has been deployed, you can manually edit the cron schedule in `/etc/cron.d/naa-schedule`.

Tools

AWS services

- [Amazon Elastic Compute Cloud (Amazon EC2)](#) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- [Amazon EventBridge](#) is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.

- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Security Hub provides a comprehensive view of your security state in AWS. It also helps you check your AWS environment against security industry standards and best practices.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale. This pattern uses Session Manager, a capability of Systems Manager.

Code repository

The code for this pattern is available in the GitHub Network Access Analyzer Multi-Account Analysis repository. The code repository contains the following files:

- **naa-script.sh** – This bash script is used to start a Network Access Analyzer analysis of multiple AWS accounts, in parallel. As defined in the **naa-resources.yaml** CloudFormation template, this script is automatically deployed to the `/usr/local/naa` folder on the Amazon EC2 instance.
- **naa-resources.yaml** – You use this CloudFormation template to create a stack in the security account in the organization. This template deploys all of the required resources for this account in order to support the solution. This stack must be deployed before the **naa-execrole.yaml** template.

  > **Note**
  >
  > If this stack is deleted and redeployed, you must rebuild the `NAAExecRole` stack set in order to rebuild the cross-account dependencies between the IAM roles.

- **naa-execrole.yaml** – You use this CloudFormation template to create a stack set that deploys the `NAAExecRole` IAM role in all accounts in the organization, including the management account.

- **naa-processfindings.py** – The **naa-script.sh** script automatically calls this Python script to process the Network Access Analyzer JSON outputs, exclude any known-good resources in the **naa-exclusions.csv** file, and then either generate a CSV file of the consolidated results or import the results into Security Hub.

Epics

Prepare for deployment

Task	Description	Skills required
Clone the code repository.	1. In a command-line interface, change your working directory to the location where you want to store the sample files. 2. Enter the following command: `git clone https://github.com/aws-samples/network-access-analyzer-multi-account-analysis.git`	AWS DevOps
Review the templates.	1. In the cloned repository, open the **naa-resources.yaml** and **naa-execrole.yaml** files. 2. Review the resources created by these templates and adjust the templates as needed for your environment. For more information, see [Working with templates](#) in	AWS DevOps

Task	Description	Skills required
	the CloudFormation documentation. 3. Save and close the **naa-resources.yaml** and **naa-execrole.yaml** files.	

Create the CloudFormation stacks

Task	Description	Skills required
Provision resources in the security account.	Using the **naa-resources.yaml** template, you create a CloudFormation stack that deploys all of the required resources in the security account. For instructions, see [Creating a stack](#) in the CloudFormation documentation. Note the following when deploying this template: 1. On the **Specify template** page, choose **Template is ready**, and then upload the **naa-resources.yaml** file. 2. On the **Specify stack details** page, in the **Stack name** box, enter `NAA-Resources`. 3. In the **Parameters** section, enter the following: • `VPCId` – Select a VPC in the account.	AWS DevOps

Task	Description	Skills required
	• `SubnetId` –– Select a private subnet that has internet access. **Note** If you select a public subnet, the Amazon EC2 instance might not be assigned a public IP address because the CloudFormation template, by default, doesn't provision and attach an Elastic IP address. • `InstanceType` – Leave the default instance type. • `InstanceImageId` – Leave the default. • `KeyPairName` – If you're using SSH for access, specify the name of an existing key pair. • `PermittedSSHInbound` – If you're using SSH for access, specify a permitted CIDR block. If you're not using SSH,	

Task	Description	Skills required
	keep the default value of `127.0.0.1` . • `BucketName` – The default value is `naa-<accountID>-<region>` . You can modify this as needed. If you specify a custom value, the account ID and Region are automatically appended to the specified value. • `EmailAddress` – Specify an email address for an Amazon SNS notification when the analysis is complete. **Note** The Amazon SNS subscription configuration must be confirmed prior to completion of the analysis, or a notification will not be sent. • `NAAEC2Role` – Keep the default unless your naming conventions require a different name for this IAM role.	

Task	Description	Skills required
	• `NAAExecRole` – Keep the default unless another name will be used when deploying the **naa-execrole.yaml** • `Parallelism` – Specify the number of parallel assessments to perform. • `Regions` – Specify the AWS Regions you want to analyze. • `ScopeNameValue` – Specify the tag that will be assigned to the scope. This tag is used to determine the Network Access Scope. • `ExclusionFile` – Specify the exclusion file name. Entries in this file will be excluded from findings. • `FindingsToCSV` – Specify whether findings should be output to CSV. Accepted values are `true` and `false`. • `FindingsToSecurityHub` – Specify whether findings should be imported into Security	

Task	Description	Skills required
	Hub. Accepted values are `true` and `false`. • `EmailNotificationsForSecurityHub` – Specify whether importing findings into Security Hub should generate email notifications. Accepted values are `true` and `false`. • `ScheduledAnalysis` – If you want the solution to run automatically on a schedule, enter `true`, and then customize the schedule in the `CronScheduleExpression` parameter. If you do not want to run the solution automatically, enter `false`. • `CronScheduleExpression` – If you're running the solution automatically, enter a cron expression to define the schedule. For more information, see *Automation and scale* in the [Architecture](#) section of this pattern.	

Task	Description	Skills required
	1. On the **Review** page, select **The following resource(s) require capabilities: [AWS::IAM::Role]**, and then choose **Create Stack**. 2. After the stack has been successfully created, in the CloudFormation console, on the **Outputs** tab, copy the `NAAEC2Role` Amazon Resource Name (ARN). You use this ARN later when deploying the **naa-execrole.yaml** file.	

Task	Description	Skills required
Provision the IAM role in the member accounts.	In the AWS Organizations management account or an account with delegated administrator permissions for CloudFormation, use the **naa-execrole.yaml** template to create a CloudFormation stack set. The stack set deploys the `NAAExecRole` IAM role in all member accounts in the organization. For instructions, see [Create a stack set with service-managed permissions](#) in the CloudFormation documentation. Note the following when deploying this template: 1. Under **Prepare template**, choose **Template is ready**, and then upload the **naa-execrole.yaml** file. 2. On the **Specify StackSet details** page, name the stack set `NAA-ExecRole` . 3. In the **Parameters** section, enter the following: • `AuthorizedARN` – Enter the `NAAEC2Role` ARN, which you copied when you created the `NAA-Resources` stack. • `NAARoleName` – Keep the default value of	AWS DevOps

Task	Description	Skills required
	`NAAExecRole` unless another name was used when deploying the **naa-resources.yaml** file. 4. Under **Permissions**, choose **Service-managed permissions**. 5. On the **Set deployment options** page, under **Deployment targets**, choose **Deploy to organization** and accept all defaults. **Note** If you want the stacks deployed to all member accounts simultaneously, set **Maximum concurrent accounts** and **Failure tolerance** to a high value, such as `100`. 6. Under **Deployment regions**, choose the Region where the Amazon EC2 instance for Network Access Analyzer is deployed. Because IAM resources are global and	

Task	Description	Skills required
	not Regional, this deploys the IAM role in all active Regions. 7. On the **Review** page, select **I acknowledge that AWS CloudFormation might create IAM resources with custom names**, and then choose **Create StackSet**. 8. Monitor the **Stack instances** tab (for individua l account status) and the **Operations** tab (for overall status) to determine when the deployment is complete.	

Task	Description	Skills required
Provision the IAM role in the management account.	Using the **naa-execrole.yaml** template, you create a CloudFormation stack that deploys the `NAAExecRole` IAM role in the managemen t account of the organizat ion. The stack set you created previously doesn't deploy the IAM role in the managemen t account. For instructions, see [Creating a stack](#) in the CloudFormation documenta tion. Note the following when deploying this template: 1. On the **Specify template** page, choose **Template is ready**, and then upload the **naa-execrole.yaml** file. 2. On the **Specify stack details** page, in the **Stack name** box, enter `NAA-ExecRole` . 3. In the **Parameters** section, enter the following: • `AuthorizedARN` – Enter the `NAAEC2Role` ARN, which you copied when you created the `NAA-Resources` stack. • `NAARoleName` – Keep the default value of `NAAExecRole` unless another name was used	AWS DevOps

Task	Description	Skills required
	when deploying the **naa-resources.yaml** file. 4. On the **Review** page, select **The following resource(s) require capabilities: [AWS::IAM::Role]**, and then choose **Create Stack**.	

Perform the analysis

Task	Description	Skills required

Customize the shell script.	1. Sign in to the security account in the organization.
	2. Using Session Manager, connect to the Amazon EC2 instance for Network Access Analyzer that you previously provisioned. For instructions, see [Connect to your Linux instance using Session Manager](). If you're unable to connect, see the [Troubleshooting]() section of this pattern.
	3. Enter the following commands to open the **naa-script.sh** file for editing:
	`sudo -i`
	`cd /usr/local/naa`
	`vi naa-script.sh`
		AWS DevOps

Task	Description	Skills required
	4. Review and modify the adjustable parameters and variables in this script as needed for your environment. For more information about customization options, see the comments at the beginning of the script. For example, instead of getting a list of all member accounts in the organization from the management account, you can modify the script to specify the AWS account IDs or AWS Regions that you want to scan, or you can reference an external file that contains these parameters. 5. Save and close the **naa-script.sh** file.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Analyze the target accounts.</td><td>
1. Enter the following commands. This runs the naa-script.sh script:
<pre><code>sudo -i
cd /usr/local/naa
screen
./naa-script.sh</code></pre>
Note the following:

The <code>screen</code> command permits the script to continue running in the event that the connectio n times out or you lose console access.
After the scan starts, you can force a screen detach by pressing Ctrl+A D. The screen detaches, and you can close the instance connectio n while the analysis proceeds.
To resume a detached session, connect to the instance, enter <code>sudo -i</code> then enter <code>screen -r</code>.

2. Monitor the output for any errors to make sure that the script is working properly. For a sample output, see the Additional
</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	information section of this pattern. 3. Wait for the analysis to complete. If you configured email notifications, you receive an email when the results have been uploaded to the Amazon S3 bucket or imported into Security Hub.	
Option 1 – Retrieve the results from the Amazon S3 bucket.	1. Download the CSV file from the `naa-<accountID>-<region>` bucket. For instructions, see Downloading an object in the Amazon S3 documentation. 2. Delete the CSV file from the Amazon S3 bucket. This is a best practice for cost optimization. For instructions, see Deleting objects in the Amazon S3 documentation.	AWS DevOps

Task	Description	Skills required
Option 2 – Review the results in Security Hub.	1. Open the [Security Hub console](). 2. Choose **Findings** from the navigation pane. 3. Review the Network Access Analyzer findings. For instructions, see [Viewing finding lists and details]() in the Security Hub documentation. **Note** You can search findings by adding a **Title starts with** filter and entering `Network Access Analyzer`.	AWS DevOps

Remediate and exclude findings

Task	Description	Skills required
Remediate findings.	Remediate any findings that you want to address. For more information and best practices about how to create a perimeter around your AWS identities, resources, and networks, see [Building a data perimeter on AWS]() (AWS Whitepaper).	AWS DevOps

Task	Description	Skills required
Exclude resources with known-good network paths.	If Network Access Analyzer generates findings for resources that should be accessible from the internet, then you can add these resources to an exclusion list. The next time Network Access Analyzer runs, it won't generate a finding for that resource. 1. Navigate to `/usr/local/naa` , and then open the **naa-script.sh** script. Make note of the value of the `S3_EXCLUSION_FILE` variable. 2. If the value of the `S3_EXCLUSION_FILE` variable is `true`, download the **naa-exclusions.csv** file from the `naa-<accountID>-<region>` bucket. For instructions, see [Downloading an object](#) in the Amazon S3 documentation. If the value of the `S3_EXCLUSION_FILE` variable is `false`, navigate to `/usr/local/naa` and then open the **naa-exclusions.csv** file.	AWS DevOps

Task	Description	Skills required

	Note
	If the value of the `S3_EXCLUSION_FILE` variable is `false`, the script uses a local version of the exclusions file. If you later change the value to `true`, then the script overwrites the local version with the file in the Amazon S3 bucket.
	3. In the **naa-exclusions.csv** file, enter the resources that you want to exclude. Enter one resource in each line, and use the following format.
	`<resource_id>,<secgroup_id>,<sgrule_cidr>,<sgrule_portrange>,<sgrule_protocol>`
	The following is an example resource.
	`eni-1111aaaaa2222bbbb,sg-3333ccccc44`	

Task	Description	Skills required
	`44dddd,0.0.0.0/0,80 to 80,tcp` 4. Save and close the **naa-exclusions.csv** file. 5. If you downloaded the **naa-exclusions.csv** file from the Amazon S3 bucket, upload the new version. For instructions, see [Uploading objects] in the Amazon S3 documentation.	

(Optional) Update the naa-script.sh script

Task	Description	Skills required
Update the naa-script.sh script.	If you want to update the **naa-script.sh** script to the latest version in the repo, do the following: 1. Connect to the Amazon EC2 instance by using Session Manager. For instructions, see [Connect to your Linux instance using Session Manager]. 2. Enter the following command: `sudo -i`	AWS DevOps

Task	Description	Skills required

	3. Navigate to the **naa-script.sh** script directory:
	`cd /usr/local/naa`
	4. Enter the following command to stash the local script so that you can merge custom changes into the newest version:
	`git stash`
	5. Enter the following command to get the latest version of the script:
	`git pull`
	6. Enter the following command to merge the custom script with the latest version of the script:
	`git stash pop`	

(Optional) Clean up

Task	Description	Skills required
Delete all deployed resources.	You can leave the resources deployed in the accounts. If you want to deprovision all resources, do the following:	AWS DevOps

Task	Description	Skills required
	1. Delete the `NAA-ExecRole` stack provisioned in the management account. For instructions, see [Deleting a stack](#) in the CloudFormation documentation. 2. Delete the `NAA-ExecRole` stack set provisioned in the organization's management account or in the delegated administrator account. For instructions, see [Delete a stack set](#) in the CloudFormation documentation. 3. Delete all objects in the `naa-<accountID>-<region>` Amazon S3 bucket. For instructions, see [Deleting objects](#) in the Amazon S3 documentation. 4. Delete the `NAA-Resources` stack provisioned in the security account. For instructions, see [Deleting a stack](#) in the CloudFormation documentation.	

Troubleshooting

Issue	Solution
Unable to connect to the Amazon EC2 instance by using Session Manager.	The SSM Agent must be able to communicate with the Systems Manager endpoint. Do the following: 1. Validate the subnet where the Amazon EC2 instance is deployed has internet access. 2. Reboot the Amazon EC2 instance.
When deploying the stack set, the CloudFormation console prompts you to `Enable trusted access with AWS Organizations to use service-managed permissions .`	This indicates that trusted access has not been enabled between AWS Organizations and CloudFormation. Trusted access is required to deploy the service-managed stack set. Choose the button to enable trusted access. For more information, see Enable trusted access in the CloudFormation documentation.

Related resources

- New – Amazon VPC Network Access Analyzer (AWS blog post)
- AWS re:Inforce 2022 - Validate effective network access controls on AWS (NIS202) (video)
- Demo - Organization-wide Internet Ingress Data Path Analysis Using Network Access Analyzer (video)

Additional information

Example console output

The following sample shows the output of generating the list of target accounts and analyzing the target accounts.

```
[root@ip-10-10-43-82 naa]# ./naa-script.sh
download: s3://naa-<account ID>-us-east-1/naa-exclusions.csv to ./naa-exclusions.csv
```

```
AWS Management Account: <Management account ID>

AWS Accounts being processed...
<Account ID 1> <Account ID 2> <Account ID 3>

Assessing AWS Account: <Account ID 1>, using Role: NAAExecRole
Assessing AWS Account: <Account ID 2>, using Role: NAAExecRole
Assessing AWS Account: <Account ID 3>, using Role: NAAExecRole
Processing account: <Account ID 1> / Region: us-east-1
Account: <Account ID 1> / Region: us-east-1 – Detecting Network Analyzer scope...
Processing account: <Account ID 2> / Region: us-east-1
Account: <Account ID 2> / Region: us-east-1 – Detecting Network Analyzer scope...
Processing account: <Account ID 3> / Region: us-east-1
Account: <Account ID 3> / Region: us-east-1 – Detecting Network Analyzer scope...
Account: <Account ID 1> / Region: us-east-1 – Network Access Analyzer scope detected.
Account: <Account ID 1> / Region: us-east-1 – Continuing analyses with Scope ID.
 Accounts with many resources may take up to one hour
Account: <Account ID 2> / Region: us-east-1 – Network Access Analyzer scope detected.
Account: <Account ID 2> / Region: us-east-1 – Continuing analyses with Scope ID.
 Accounts with many resources may take up to one hour
Account: <Account ID 3> / Region: us-east-1 – Network Access Analyzer scope detected.
Account: <Account ID 3> / Region: us-east-1 – Continuing analyses with Scope ID.
 Accounts with many resources may take up to one hour
```

CSV report examples

The following images are examples of the CSV output.

	A	B	C	D	E	F	G	H
1	account	region	vpc_id	subnet_id	instance_id	instance_arn	instance_name	resource_id
2	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	dev_server	eni-11111111111111111
3	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	dev_server	eni-11111111111111111
4	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	dev_server	eni-11111111111111111
5	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	Bastion-PublicSubnet	eni-11111111111111111
6	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	Bastion-PublicSubnet	eni-11111111111111111
7	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	lnx-sorter	eni-11111111111111111
8	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	lnx-sorter	eni-11111111111111111
9	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	lnx-sorter	eni-11111111111111111
10	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	Container_LZ-Public	eni-11111111111111111
11	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	Container_LZ-Public	eni-11111111111111111
12	111111111111	us-east-1	vpc-11111111111111111	subnet-11111111111111111	i-11111111111111111	arn:aws:ec2:us-east-1:111111111111:instance/i-11111111111111111	Container_LZ-Public	eni-11111111111111111
13	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	N/A	N/A	N/A	eni-22222222222222222
14	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	N/A	N/A	N/A	eni-22222222222222222
15	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	N/A	N/A	N/A	eni-22222222222222222
16	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	N/A	N/A	N/A	eni-22222222222222222
17	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	i-22222222222222222	arn:aws:ec2:us-east-1:222222222222:instance/i-22222222222222222	Test-FileGateway	eni-22222222222222222
18	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	i-22222222222222222	arn:aws:ec2:us-east-1:222222222222:instance/i-22222222222222222	Dev-LinuxGateway	eni-22222222222222222
19	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	N/A	N/A	N/A	eni-22222222222222222
20	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	i-22222222222222222	arn:aws:ec2:us-east-1:222222222222:instance/i-22222222222222222	testgw	eni-22222222222222222
21	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	N/A	N/A	N/A	eni-22222222222222222
22	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	i-22222222222222222	arn:aws:ec2:us-east-1:222222222222:instance/i-22222222222222222	FastProcessor	eni-22222222222222222
23	222222222222	us-east-1	vpc-22222222222222222	subnet-22222222222222222	i-22222222222222222	arn:aws:ec2:us-east-1:222222222222:instance/i-22222222222222222	FastProcessor	eni-22222222222222222
24	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	FastProcessor	eni-33333333333333333
25	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	FastProcessor	eni-33333333333333333
26	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	Liam_Solution	eni-33333333333333333
27	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	Liam_Solution	eni-33333333333333333
28	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	Test-FileGateway	eni-33333333333333333
29	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	Dev-LinuxGateway	eni-33333333333333333
30	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	Dev-LinuxGateway	eni-33333333333333333
31	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	FastProcessor	eni-33333333333333333
32	333333333333	us-east-1	vpc-33333333333333333	subnet-33333333333333333	i-33333333333333333	arn:aws:ec2:us-east-1:333333333333:instance/i-33333333333333333	FastProcessor	eni-33333333333333333

H	I	J	K	L	M	N
resource_id	resource_arn	secgroup_id	sgrule_direction	sgrule_cidr	sgrule_protocol	sgrule_portrange
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.27/30	all	all
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.0/24	tcp	443 to 443
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.0/24	tcp	5000 to 5000
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.0/24	tcp	3389 to 3389
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.15/32	tcp	3389 to 3389
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.93/32	tcp	3389 to 3389
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	0.0.0.0/0	tcp	80 to 80
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	0.0.0.0/0	tcp	443 to 443
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.15/32	tcp	22 to 22
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.0/24	tcp	22 to 22
eni-11111111111111111	arn:aws:ec2:us-east-1:111111111111:network-interface/eni-11111111111111111	sg-11111111111111111	ingress	192.0.2.51/30	tcp	22 to 22
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	192.0.2.15/32	tcp	80 to 80
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	sg-22222222222222222	tcp	80 to 80
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	sg-22222222222222222	tcp	80 to 80
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	sg-22222222222222222	tcp	80 to 80
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	0.0.0.0/0	all	all
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	0.0.0.0/0	all	all
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	0.0.0.0/0	tcp	1433 to 1433
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	0.0.0.0/0	tcp	22 to 22
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	0.0.0.0/0	tcp	1433 to 1433
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	0.0.0.0/0	tcp	22 to 22
eni-22222222222222222	arn:aws:ec2:us-east-1:222222222222:network-interface/eni-22222222222222222	sg-22222222222222222	ingress	0.0.0.0/0	tcp	80 to 80
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	tcp	443 to 443
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	tcp	8443 to 8443
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	tcp	22 to 22
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	tcp	80 to 80
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	all	all
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	all	all
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	tcp	2049 to 2049
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	tcp	22 to 22
eni-33333333333333333	arn:aws:ec2:us-east-1:333333333333:network-interface/eni-33333333333333333	sg-33333333333333333	ingress	0.0.0.0/0	tcp	80 to 80

Set up DNS resolution for hybrid networks in a multi-account AWS environment

Anvesh Koganti, Amazon Web Services

Summary

This pattern provides a comprehensive solution for setting up DNS resolution in hybrid network environments that include multiple Amazon Web Services (AWS) accounts. It enables bidirectional DNS resolution between on-premises networks and the AWS environment through Amazon Route 53 Resolver endpoints. The pattern presents two solutions to enable DNS resolution in a multi-account, centralized architecture:

- *Basic setup* doesn't use Route 53 Profiles. It helps optimize costs for small to medium deployments of lower complexity.
- *Enhanced setup* uses Route 53 Profiles to simplify operations. It is best for larger or more complex DNS deployments.

> **Note**
>
> Review the *Limitations* section for service limitations and quotas before implementation. Consider factors such as management overhead, costs, operational complexity, and team expertise when you make your decision.

Prerequisites and limitations

Prerequisites

- An AWS multi-account environment with Amazon Virtual Private Cloud (Amazon VPC) deployed across Shared Services and workload accounts (preferably set up through AWS Control Tower by following AWS best practices for account structure).
- Existing hybrid connectivity (AWS Direct Connect or AWS Site-to-Site VPN) between your on-premises network and the AWS environment.
- Amazon VPC peering, AWS Transit Gateway, or AWS Cloud WAN for Layer 3 network connectivity between VPCs. (This connectivity is required for application traffic. It is not required for DNS

resolution to work. DNS resolution operates independently of network connectivity between the VPCs.)

- DNS servers running in the on-premises environment.

Limitations

- Route 53 Resolver endpoints, rules, and Profiles are Regional constructs and might require replication in multiple AWS Regions for global organizations.
- For a comprehensive list of service quotas for Route 53 Resolver, private hosted zones, and Profiles, see Quotas in the Route 53 documentation.

Architecture

Target technology stack

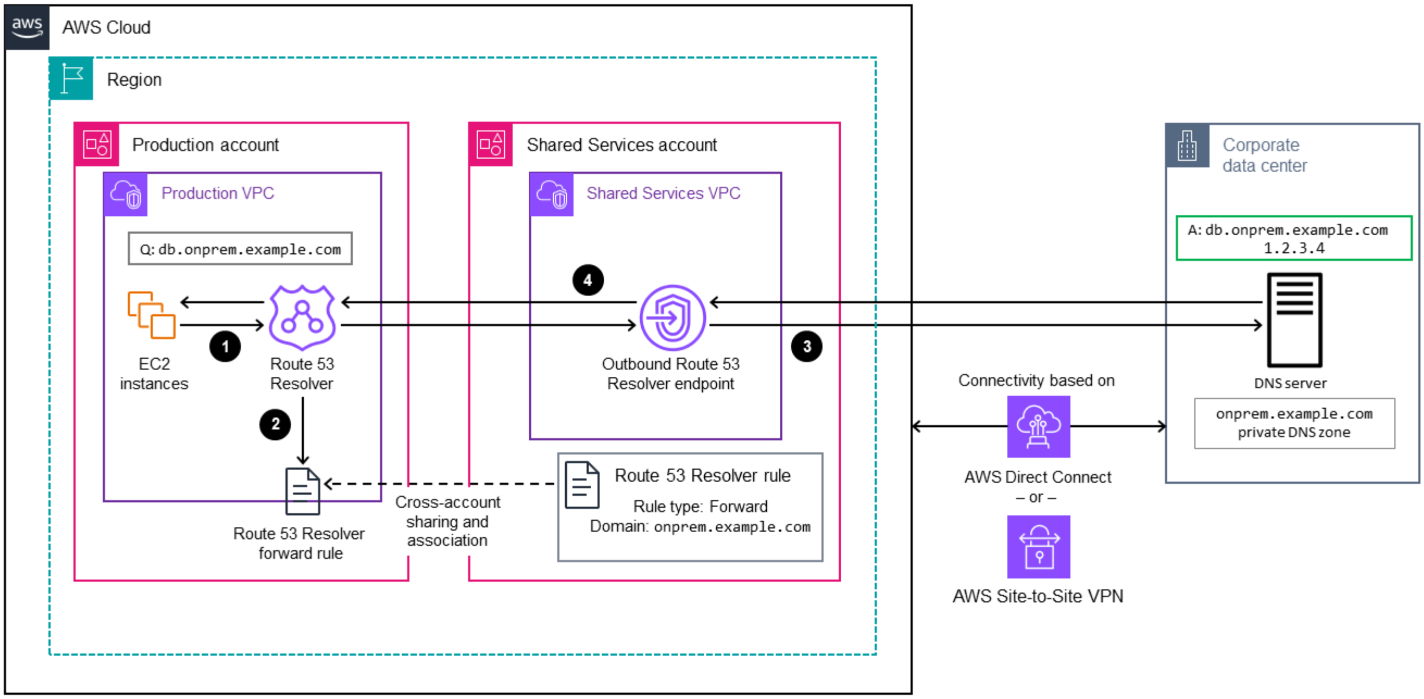
- Route 53 outbound and inbound endpoints
- Route 53 Resolver rules for conditional forwarding
- AWS Resource Access Manager (AWS RAM)
- Route 53 private hosted zone

Target architecture

Outbound and inbound endpoints

The following diagram shows the DNS resolution flow from AWS to on premises. This is the connectivity setup for outbound resolutions where the domain is hosted on premises. Here is a high-level overview of the process involved in setting this up. For details, see the Epics section.

1. Deploy outbound Route 53 Resolver endpoints in the Shared Services VPC.
2. Create Route 53 Resolver rules (forwarding rules) in the Shared Services account for domains that are hosted on premises.
3. Share and associate the rules with VPCs in other accounts that host resources that need to resolve on-premises hosted domains. This can be done in different ways depending on your use case, as described later in this section.



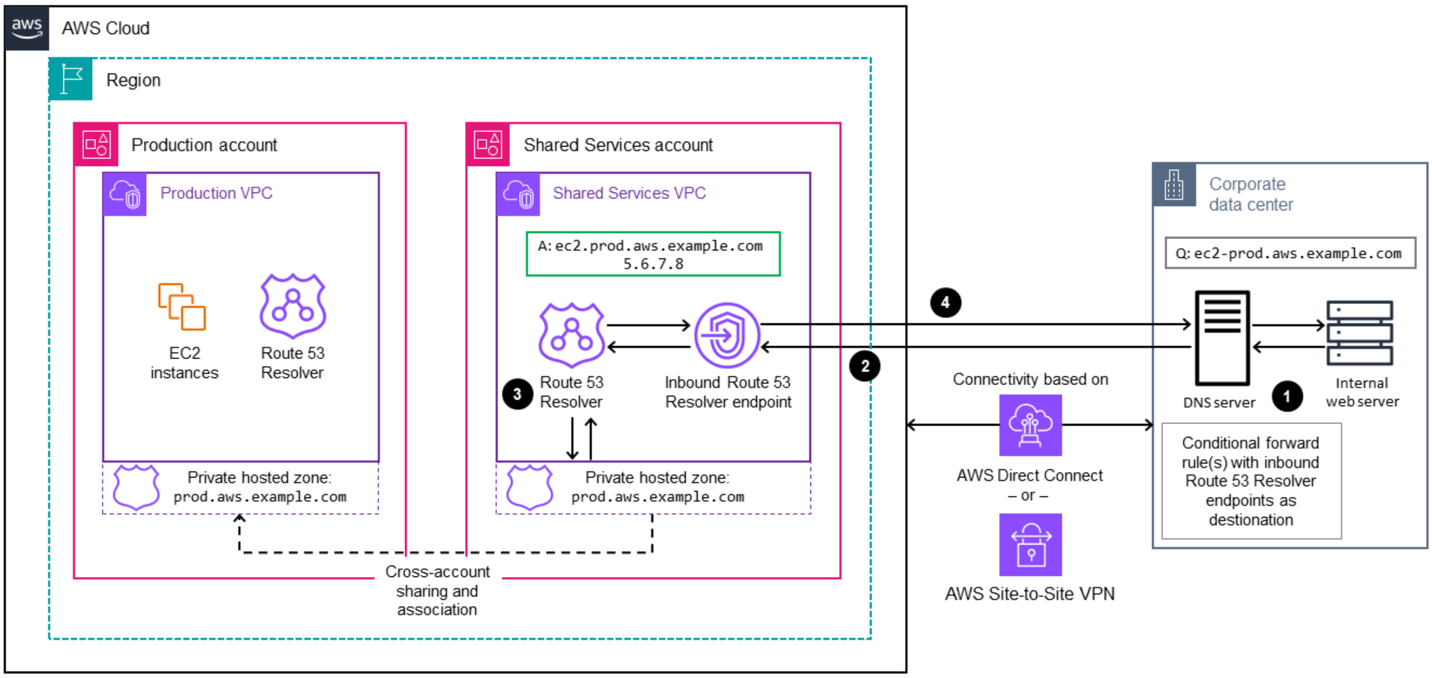
After you set up connectivity, the steps involved in the outbound resolution are as follows:

1. The Amazon Elastic Compute Cloud (Amazon EC2) instance sends a DNS resolution request for `db.onprem.example.com` to the VPC's Route 53 Resolver at the VPC+2 address.
2. Route 53 Resolver checks the Resolver rules and forwards the request to the on-premises DNS server IPs by using the outbound endpoint.
3. The outbound endpoint forwards the request to the on-premises DNS IPs. The traffic goes over the established hybrid network connectivity between the Shared Services VPC and the on-premises data center.
4. The on-premises DNS server responds back to the outbound endpoint, which then forwards the response back to the VPC's Route 53 Resolver. The Resolver returns the response to the EC2 instance.

The next diagram shows the DNS resolution flow from the on-premises environment to AWS. This is the connectivity setup for inbound resolutions where the domain is hosted on AWS. Here is a high-level overview of the process involved in setting this up. For details, see the Epics section.

1. Deploy inbound Resolver endpoints in the Shared Services VPC.
2. Create private hosted zones in the Shared Services account (centralized approach).

3. Associate the private hosted zones with the Shared Services VPC. Share and associate these zones with cross-account VPCs for VPC-to-VPC DNS resolution. This can be done in different ways depending on your use case, as described later in this section.



After you set up connectivity, the steps involved in the inbound resolution are as follows:

1. The on-premises resource sends a DNS resolution request for `ec2.prod.aws.example.com` to the on-premises DNS server.
2. The on-premises DNS server forwards the request to the inbound Resolver endpoint in the Shared Services VPC over the hybrid network connection.
3. The inbound Resolver endpoint looks up the request in the associated private hosted zone with the help of the VPC Route 53 Resolver and gets the appropriate IP address.
4. These IP addresses are sent back to the on-premises DNS server, which returns the response to the on-premises resource.

This configuration enables on-premises resources to resolve AWS private domain names by routing queries through the inbound endpoints to the appropriate private hosted zone. In this architecture, private hosted zones are centralized in a Shared Services VPC, which allows for central DNS management by a single team. These zones can be associated with many VPCs to address the VPC-to-VPC DNS resolution use case. Alternatively, you might want to delegate DNS domain ownership and management to each AWS account. In that case, each account manages its own private hosted zones and associates each zone with the central Shared Services VPC for a unified resolution with the on-premises environment. This decentralized approach is outside the scope of this pattern. For more information, see [Scaling DNS management across multiple accounts and VPCs]() in the *Hybrid Cloud DNS Options for Amazon VPC* whitepaper.

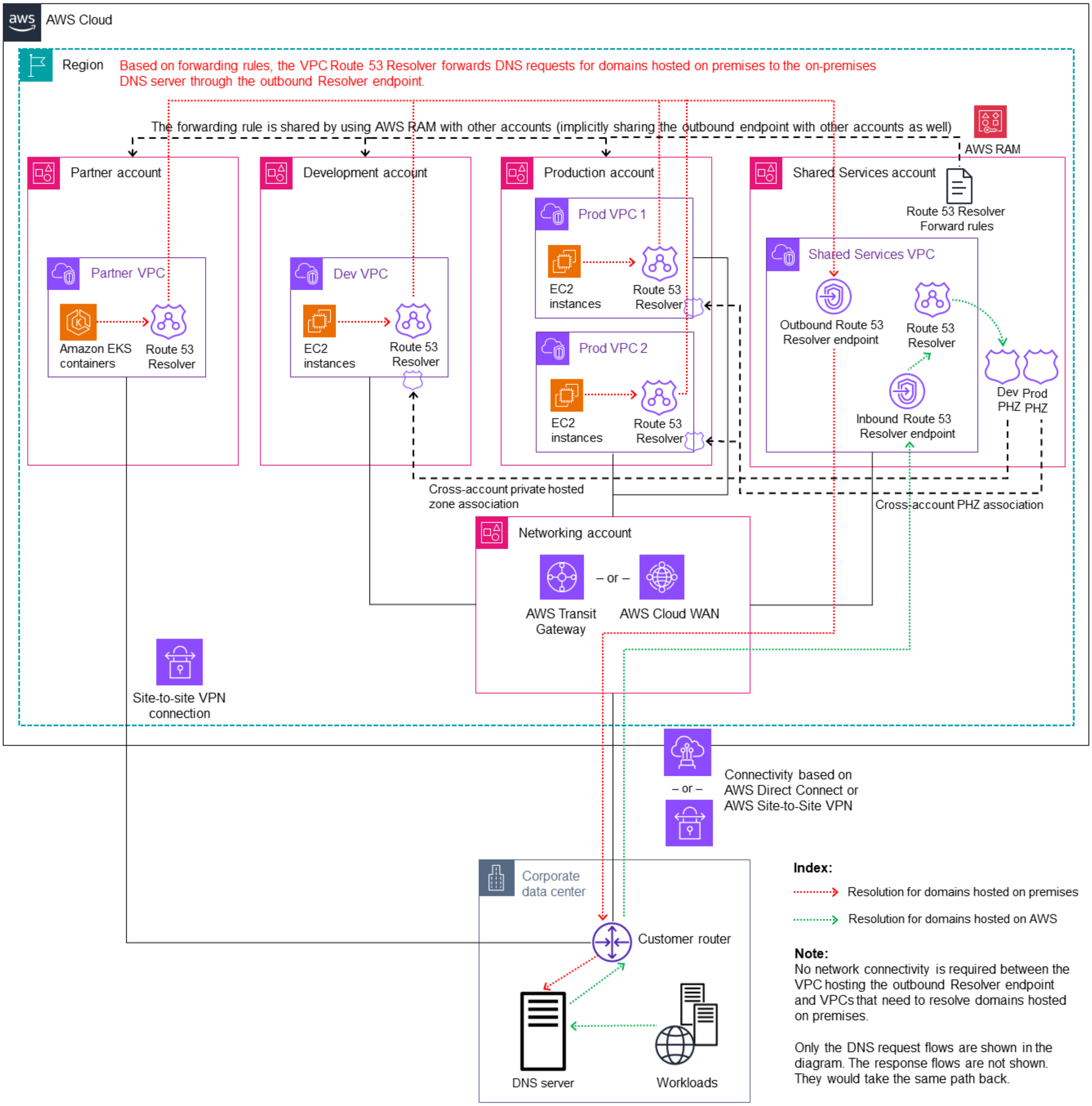
When you establish the fundamental DNS resolution flows by using Resolver endpoints, you need to determine how to manage the sharing and association of Resolver rules and private hosted zones across your AWS accounts. You can approach this in two ways: through self-managed sharing by using AWS RAM to share Resolver rules and direct private hosted zone associations, as detailed in the *Basic setup* section, or through Route 53 Profiles, as discussed in the *Enhanced setup* section. The choice depends on your organization's DNS management preferences and operational requirements. The following architecture diagrams illustrate a scaled environment that includes multiple VPCs across different accounts, which represents a typical enterprise deployment.

Basic setup

In basic setup, the implementation for hybrid DNS resolution in a multi-account AWS environment uses AWS RAM to share Resolver forwarding rules and private hosted zone associations to manage DNS queries between on-premises and AWS resources. This method uses centralized Route 53 Resolver endpoints in a Shared Services VPC that's connected to your on-premises network to handle both inbound and outbound DNS resolution efficiently.

- For outbound resolution, Resolver forwarding rules are created in the Shared Services account and then shared with other AWS accounts by using AWS RAM. This sharing is limited to accounts within the same Region. The target accounts can then associate these rules with their VPCs and enable the resources in those VPCs to resolve on-premises domain names.
- For inbound resolution, private hosted zones are created in the Shared Services account and associated with the Shared Services VPC. These zones can then be associated with VPCs in other accounts by using the Route 53 API, AWS SDKs, or the AWS Command Line Interface (AWS CLI). The resources in associated VPCs can then resolve DNS records defined in the private hosted zones, which creates a unified DNS view across your AWS environment.

The following diagram shows DNS resolution flows in this basic setup.

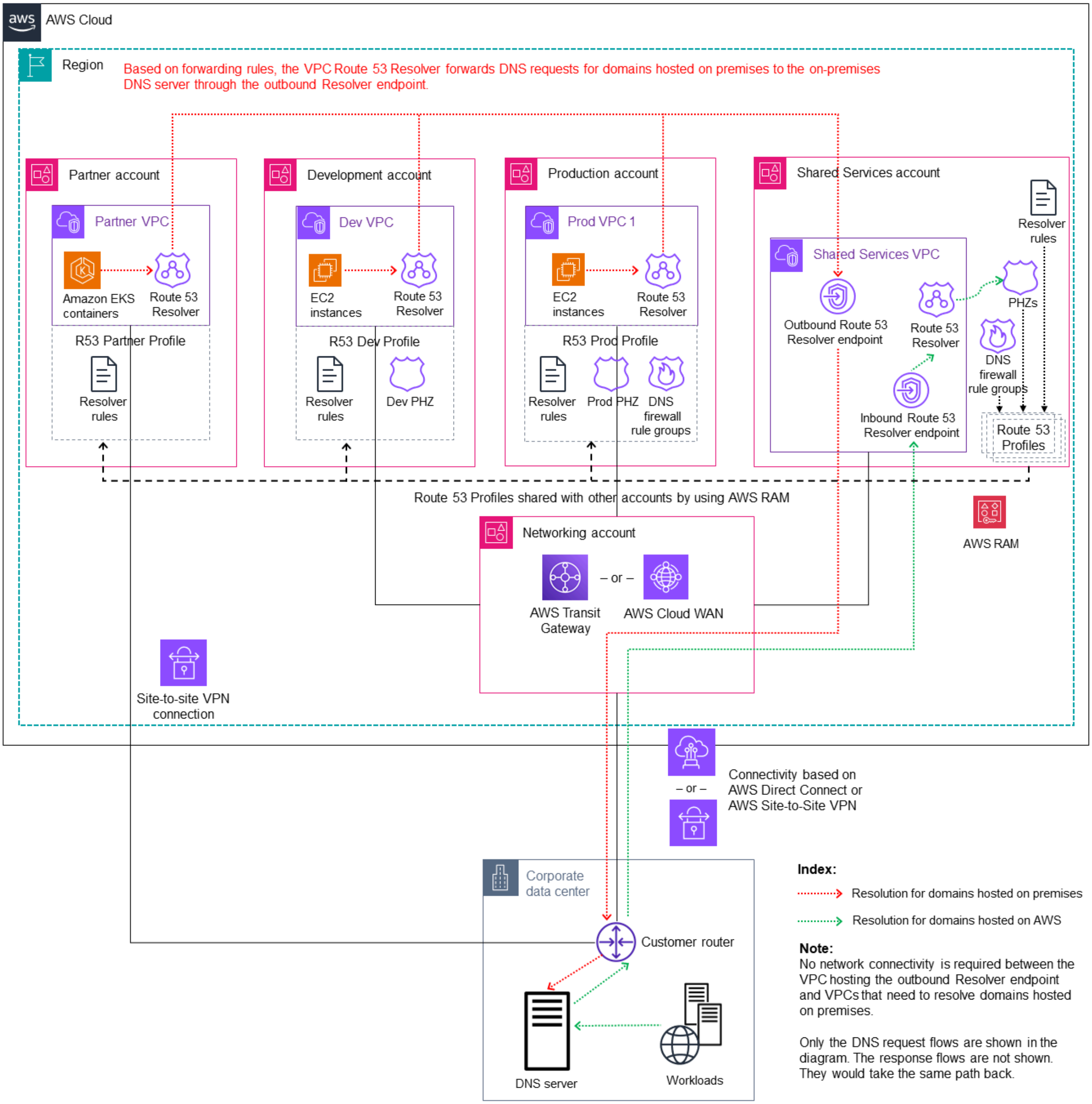


This setup works well when you work with DNS infrastructure on a limited scale. However, it can become challenging to manage as your environment grows. The operational overhead of managing how private hosted zone and Resolver rules are shared and associated with VPCs individually increases significantly with scale. Additionally, service quotas such as the 300 VPC association limit per private hosted zone can become constraining factors in large-scale deployments. Enhanced setup addresses these challenges.

Enhanced setup

Route 53 Profiles offer a streamlined solution for managing DNS resolution in hybrid networks across multiple AWS accounts. Instead of managing private hosted zones and Resolver rules individually, you can group DNS configurations into a single container that can be easily shared and applied across multiple VPCs and accounts in a Region. This setup maintains the centralized Resolver endpoint architecture in a Shared Services VPC while significantly simplifying the management of DNS configurations.

The following diagram shows DNS resolution flows in an enhanced setup.



Route 53 Profiles let you package private hosted zone associations, Resolver forwarding rules, and DNS firewall rules into a single, shareable unit. You can create Profiles in the Shared Services account and share them with member accounts by using AWS RAM. When a profile is shared and applied to target VPCs, all necessary associations and configurations are automatically handled by

the service. This significantly reduces the operational overhead of DNS management and provides excellent scalability for growing environments.

Automation and scale

Use infrastructure as code (IaC) tools such as AWS CloudFormation or Terraform to automatically provision and manage Route 53 Resolver endpoints, rules, private hosted zones, and Profiles. Integrate DNS configuration with continuous integration and continuous delivery (CI/CD) pipelines for consistency, repeatability, and rapid updates.

Tools

AWS services

- AWS Resource Access Manager (AWS RAM) helps you securely share your resources across AWS accounts to reduce operational overhead and provide visibility and auditability.
- Amazon Route 53 Resolver responds recursively to DNS queries from AWS resources and is available by default in all VPCs. You can create Resolver endpoints and conditional forwarding rules to resolve DNS namespaces between your on-premises data center and your VPCs.
- Amazon Route 53 private hosted zone is a container that holds information about how you want Route 53 to respond to DNS queries for a domain and its subdomains.
- Amazon Route 53 Profiles allow you to apply and manage DNS-related Route 53 configurations across many VPCs and in different AWS accounts in a simplified manner.

Best practices

This section provides some of the best practices for optimizing Route 53 Resolver. These represent a subset of Route 53 best practices. For a comprehensive list, see Best practices for Amazon Route 53.

Avoid loop configurations with Resolver endpoints

- Design your DNS architecture to prevent recursive routing by carefully planning VPC associations. When a VPC hosts an inbound endpoint, avoid associating it with Resolver rules that could create circular references.
- Use AWS RAM strategically when you share DNS resources across accounts to maintain clean routing paths.

For more information, see Avoid loop configurations with Resolver endpoints in the Route 53 documentation.

Scale Resolver endpoints

- For environments that require a high number of queries per second (QPS), be aware that there is a limit of 10,000 QPS per ENI in an endpoint. More ENIs can be added to an endpoint to scale DNS QPS.
- Amazon CloudWatch provides `InboundQueryVolume` and `OutboundQueryVolume` metrics (see the CloudWatch documentation). We recommend that you set up monitoring rules that alert you if the threshold exceeds a certain value (for example, 80 percent of 10,000 QPS).
- Configure stateful security group rules for Resolver endpoints to prevent connection tracking limits from causing DNS query throttling during high-volume traffic. To learn more about how connection tracking works in security groups, see Amazon EC2 security group connection tracking in the Amazon EC2 documentation.

For more information, see Resolver endpoint scaling in the Route 53 documentation.

Provide high availability for Resolver endpoints

- Create inbound endpoints with IP addresses in at least two Availability Zones for redundancy.
- Provision additional network interfaces to ensure availability during maintenance or traffic surges.

For more information, see High availability for Resolver endpoints in the Route 53 documentation.

Epics

Deploy Route 53 Resolver endpoints

Task	Description	Skills required
Deploy an inbound endpoint.	Route 53 Resolver uses the inbound endpoint to receive DNS queries from on-premises DNS resolvers. For instructions, see Forwarding inbound	AWS administrator, Cloud administrator

Task	Description	Skills required
	DNS queries to your VPCs in the Route 53 documentation. Make a note of the inbound endpoint IP address.	
Deploy an outbound endpoint.	Route 53 Resolver uses the outbound endpoint to send DNS queries to on-premises DNS resolvers. For instructions, see Forwarding outbound DNS queries to your network in the Route 53 documentation. Make a note of the output endpoint ID.	AWS administrator, Cloud administrator

Configure and share Route 53 private hosted zones

Task	Description	Skills required
Create a private hosted zone for a domain that's hosted on AWS.	This zone holds the DNS records for resources in an AWS-hosted domain (for example, `prod.aws.example.com`) that should be resolved from the on-premises environment. For instructions, see Creating a private hosted zone in the Route 53 documentation. When you create a private hosted zone, you must associate a VPC with the hosted zone owned by the same account. Select the	AWS administrator, Cloud administrator

Task	Description	Skills required
	Shared Services VPC for this purpose.	
Basic setup: Associate the private hosted zone with VPCs in other accounts.	If you're using basic setup (see the Architecture section): To enable resources in the member account VPCs to resolve DNS records in this private hosted zone, you must associate your VPCs with the hosted zone. You must authorize the association and then make the associati on programmatically. For instructions, see Associating an Amazon VPC and a private hosted zone that you created with different AWS accounts in the Route 53 documenta tion.	AWS administrator, Cloud administrator

Task	Description	Skills required
Enhanced setup: Configure and share Route 53 Profiles.	If you're using enhanced setup (see the Architecture section): 1. Create a Route 53 Profile and associate the relevant private hosted zones with it. For instructions, see Creating Route 53 Profiles in the Route 53 documentation. 2. Use AWS RAM to share the profile with member accounts and then associate the shared profile with target VPCs. For instructions, see Sharing a Route 53 Profile and Associate a Route 53 Profile to VPCs in the Route 53 documentation. **Note** Depending on your organization's structure and DNS requirements, you might need to create and manage multiple Profiles for different	AWS administrator, Cloud administrator

Task	Description	Skills required
	accounts or workloads .	

Configure and share Route 53 Resolver forwarding rules

Task	Description	Skills required
Create a forwarding rule for a domain that's hosted on premises.	This rule will instruct Route 53 Resolver to forward any DNS queries for on-premises domains (such as `onprem.example.com`) to on-premises DNS resolvers. To create this rule, you need the IP addresses of the on-premises DNS resolvers and the outbound endpoint ID. For instructions, see [Creating forwarding rules](#) in the Route 53 documentation.	AWS administrator, Cloud administrator
Basic setup: Share and associate the forwarding rule with your VPCs in other accounts.	If you're using basic setup: For the forwarding rule to take effect, you must share and associate the rule with your VPCs in other accounts. Route 53 Resolver then takes the rule into considera tion when it resolves a domain. For instructions, see [Sharing Resolver rules with other AWS accounts and using shared rules](#) and	AWS administrator, Cloud administrator

Task	Description	Skills required
	Associating forwarding rules with a VPC in the Route 53 documentation.	

Task	Description	Skills required
Enhanced setup: Configure and share Route 53 Profiles.	If you're using enhanced setup: 1. If you've already created a Route 53 Profile in the previous steps, you can use the same Profile. If not, create a Route 53 Profile and associate relevant Resolver forwarding rules with it. For instructions, see [Creating Route 53 Profiles](#) in the Route 53 documentation. 2. Use AWS RAM to share the profile with member accounts, and then associate the shared profile with target VPCs. For instructions, see [Sharing a Route 53 Profile](#) and [Associate a Route 53 Profile to VPCs](#) in the Route 53 documentation. **Note** Depending on your organization's structure and DNS requirements, you might need to create and manage multiple	AWS administrator, Cloud administrator

Task	Description	Skills required
	Profiles for different accounts or workloads .	

Configure on-premises DNS resolvers for AWS integration

Task	Description	Skills required
Configure conditional forwarding in the on-premis es DNS resolvers.	For DNS queries to be sent to AWS from the on-premises environment for resolution, you must configure condition al forwarding in the on-premises DNS resolvers to point to the inbound endpoint IP address. This instructs the DNS resolvers to forward all DNS queries for the AWS-hosted domain (for example, for `prod.aws.example.com`) to the inbound endpoint IP address for resolution by Route 53 Resolver.	Network administrator

Verify end-to-end DNS resolution in a hybrid environment

Task	Description	Skills required
Test DNS resolution from AWS to the on-premises environment.	From an instance in a VPC that has the forwarding rule associated with it, perform a DNS query for an on-premises	Network administrator

Task	Description	Skills required
	hosted domain (for example, for `db.onprem.example.com`).	
Test DNS resolution from the on-premises environment to AWS.	From an on-premises server, perform DNS resolution for an AWS-hosted domain (for example, for `ec2.prod.aws.example.com`).	Network administrator

Related resources

- Hybrid Cloud DNS Options for Amazon VPC (AWS whitepaper)
- Working with private hosted zones (Route 53 documentation)
- Getting started with Route 53 Resolver (Route 53 documentation)
- Simplify DNS management in a multi-account environment with Route 53 Resolver (AWS blog post)
- Unify DNS management using Amazon Route 53 Profiles with multiple VPCs and AWS accounts (AWS blog post)
- Migrating your multi-account DNS environment to Amazon Route 53 Profiles (AWS blog post)
- Using Amazon Route 53 Profiles for scalable multi-account AWS environments (AWS blog post)

Verify that ELB load balancers require TLS termination

Priyanka Chaudhary, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, IP addresses, and AWS Lambda functions. The load balancers use listeners to define the ports and protocols that the load balancer uses to accept traffic from users. Application Load Balancers make routing decisions at the application layer and use the HTTP/HTTPS protocols. Classic Load Balancers make routing decisions at either the transport layer, by using TCP or Secure Sockets Layer (SSL) protocols, or at the application layer, by using HTTP/HTTPS.

This pattern provides a security control that examines multiple event types for Application Load Balancers and Classic Load Balancers. When the function is invoked, AWS Lambda inspects the event and ensures that the load balancer is compliant.

The function initiates an Amazon CloudWatch Events event on the following API calls: CreateLoadBalancer, CreateLoadBalancerListeners, DeleteLoadBalancerListeners, CreateLoadBalancerPolicy, SetLoadBalancerPoliciesOfListener, CreateListener, DeleteListener, and ModifyListener. When the event detects one of these APIs, it calls AWS Lambda, which runs a Python script. The Python script evaluates to see if the listener contains an SSL certificate, and if the policy that is applied is using Transport Layer Security (TLS). If the SSL policy is determined to be anything other than TLS, the function sends an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information.

Prerequisites and limitations

Prerequisites

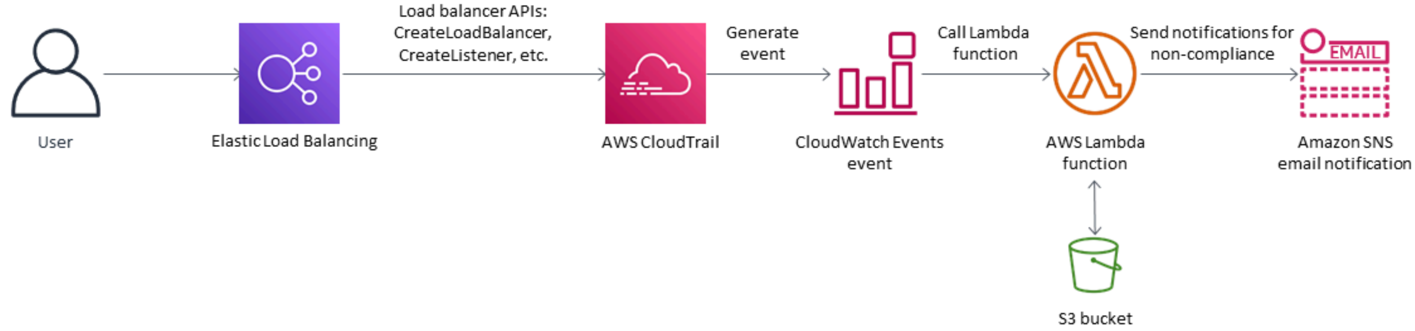
- An active AWS account

Limitations

- This security control does not check for existing load balancers, unless an update is made to the load balancer listeners.
- This security control is regional. You must deploy it in each AWS Region you want to monitor.

Architecture

Target architecture



Automation and scale

- If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes the following attachments:

- `ELBRequirestlstermination.zip` – The Lambda code for the security control.
- `ELBRequirestlstermination.yml` – The CloudFormation template that sets up the event and Lambda function.

Epics

Set up the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	On the [Amazon S3 console](https://console.aws.amazon.com/s3/), choose or create an S3 bucket to host the Lambda code .zip file. This S3 bucket must be in the same AWS Region as the load balancer that you want to evaluate. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.	Cloud architect
Upload the Lambda code.	Upload the Lambda code (`ELBRequirestlstermination.zip` file) that's provided in the *Attachments* section to the S3 bucket.	Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Launch the AWS CloudFormation template.	Open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/) in the same AWS Region as your S3 bucket and deploy the attached template `ELBRequirestlstermination.yml` . For more information about deploying AWS CloudFormation templates, see [Creating a stack on the AWS CloudFormation console](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html) in the CloudFormation documentation.	Cloud architect
Complete the parameters in the template.	When you launch the template, you'll be prompted for the following information: • **S3 bucket:** Specify the bucket that you created or selected in the first epic. This is where you uploaded the attached Lambda code (`ELBRequirestlstermination.zip` file). • **S3 key:** Specify the location of the Lambda .zip file in your S3 bucket (for example, `ELBRequirestlstermination.zip` or `controls/ELBRequirestlsterm`	Cloud architect

Task	Description	Skills required
	`ination.zip`). Do not include leading slashes. • **Notification email:** Provide an active email address where you want to receive Amazon SNS notifications. • **Lambda logging level:** Specify the logging level and frequency for the Lambda function. Use **Info** to log detailed informational messages on progress, **Error** for error events that would still allow the deployment to continue, and **Warning** for potentially harmful situations.	

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the CloudFormation template deploys successfully, it sends a subscription email to the email address you provided. You must confirm this email subscription to start receiving violation notifications.	Cloud architect

Related resources

- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- What is AWS Lambda? (AWS Lambda documentation)
- What is a Classic Load Balancer? (ELB documentation)
- What is an Application Load Balancer? (ELB documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

View AWS Network Firewall logs and metrics by using Splunk

Ivo Pinto, Amazon Web Services

Summary

Many organizations use Splunk Enterprise as a centralized aggregation and visualization tool for logs and metrics from different sources. This pattern helps you configure Splunk to fetch AWS Network Firewall logs and metrics from Amazon CloudWatch Logs by using the Splunk Add-On for AWS.

To achieve this, you create a read-only AWS Identity and Access Management (IAM) role. Splunk Add-On for AWS uses this role to access CloudWatch. You configure the Splunk Add-On for AWS to fetch metrics and logs from CloudWatch. Finally, you create visualizations in Splunk from the retrieved log data and metrics.

Prerequisites and limitations

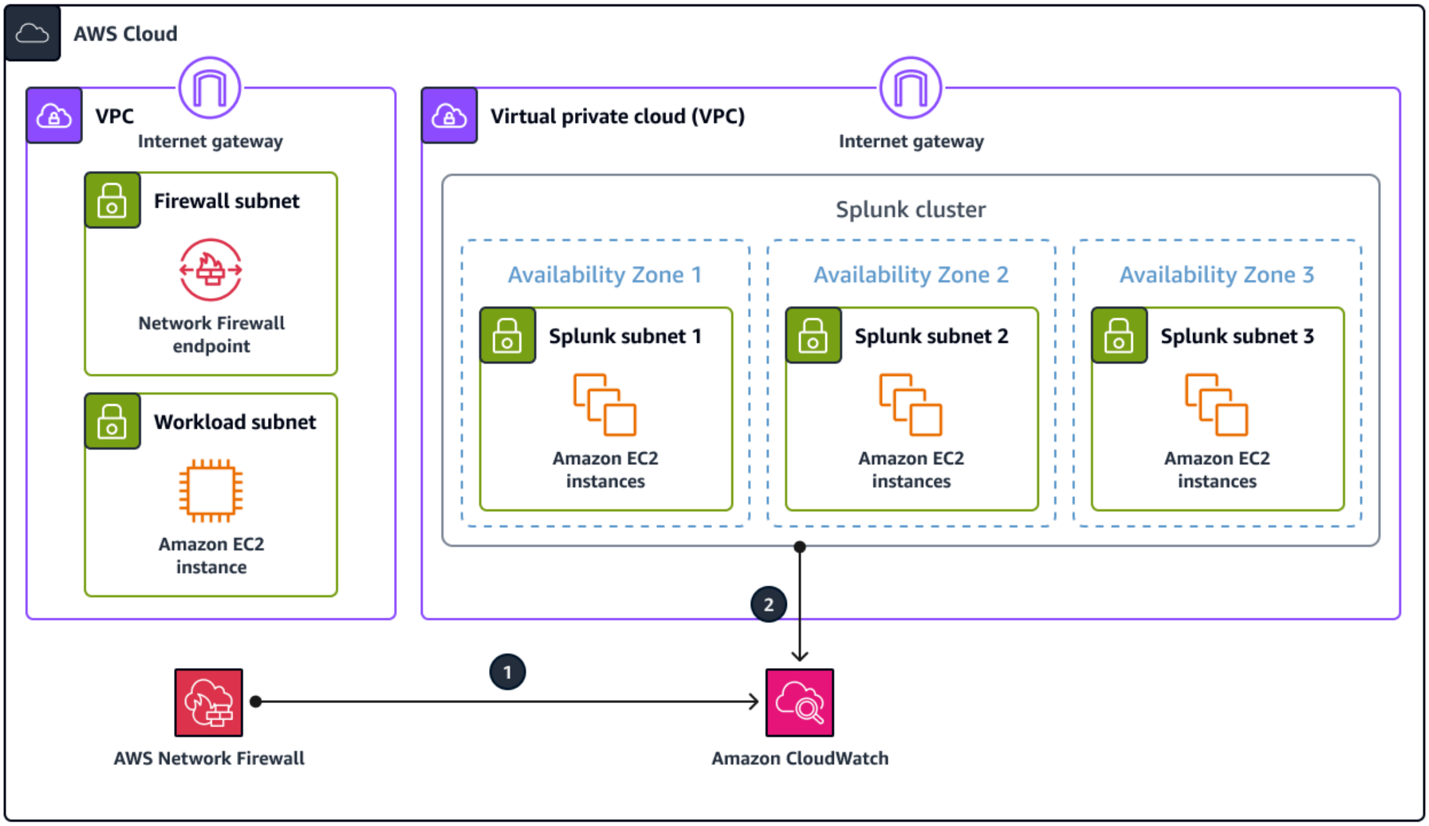
Prerequisites

- A Splunk account
- A Splunk Enterprise instance, version 8.2.2 or later
- An active AWS account
- Network Firewall, set up and configured to send logs to CloudWatch Logs

Limitations

- Splunk Enterprise must be deployed as a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances in the AWS Cloud.
- Collecting data by using an automatically discovered IAM role for Amazon EC2 is not supported in the AWS China Regions.

Architecture



The diagram illustrates the following:

1. Network Firewall publishes logs to CloudWatch Logs.
2. Splunk Enterprise retrieves metrics and logs from CloudWatch.

To populate example metrics and logs in this architecture, a workload generates traffic that passes through the Network Firewall endpoint to go to the internet. This is achieved by the use of [route tables](). Although this pattern uses a single Amazon EC2 instance as the workload, this pattern can apply to any architecture as long as Network Firewall is configured to send logs to CloudWatch Logs.

This architecture also uses a Splunk Enterprise instance in another virtual private cloud (VPC). However, the Splunk instance can be in another location, such as in the same VPC as the workload, as long as it can reach the CloudWatch APIs.

Tools

AWS services

- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service for VPCs in the AWS Cloud.

Other tools

- Splunk helps you monitor, visualize, and analyze log data.

Epics

Create an IAM role

Task	Description	Skills required

Create the IAM policy.	Follow the instructions in Creating policies using the JSON editor to create the IAM policy that grants read-only access to the CloudWatch Logs data and CloudWatch metrics. Paste the following policy into the JSON editor.
	`{`
	`    "Statement": [`
	`        {`
	`            "Action": [`
	` "cloudwatch:List*",`
	` "cloudwatch:Get*",`
	` "network-firewall:`
	`List*",`
		AWS administrator

Task	Description	Skills required

	` "logs:Describe*",`
	
	` "logs:Get*",`
	
	` "logs:List*",`
	
	` "logs:StartQuery",`
	
	` "logs:StopQuery",`
	
	` "logs:TestMetricFi lter",`
	
	` "logs:FilterLogEve nts",`
	
	` "network-firewall: Describe*"`
	`            ],`
	`            "Effect": "Allow",`
	`            "Resource": "*"`
	`        }`
	`    ],`
	`    "Version": "2012-10-17"`
	`}`	

Create a new IAM role.	Follow the instructions in [Creating a role to delegate permissions to an AWS service](#) to create the IAM role that the Splunk Add-On for AWS uses to access CloudWatc h. For **Permissions policies**, choose the policy that you created previously.	AWS administrator

Task	Description	Skills required
Assign the IAM role to the EC2 instances in the Splunk cluster.	1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2. In the navigation pane, choose **Instances**. 3. Select the EC2 instances in the Splunk cluster. 4. Choose **Actions**, **Security**, and then **Modify IAM role**. 5. Select the IAM role that you created previously, and then choose **Save**.	AWS administrator

Install the Splunk Add-On for AWS

Task	Description	Skills required
Install the add-on.	1. In the Splunk dashboard, navigate to **Splunk Apps**. 2. Search for **Splunk Add-on for Amazon Web Services**. 3. Choose **Install**. 4. Provide your Splunk credentials.	Splunk administrator
Configure the AWS credentials.	1. In the Splunk dashboard, navigate to **Splunk Add-on for AWS**. 2. Choose **Configuration**. 3. In the **Autodiscovered IAM Role** column, select the	Splunk administrator

Task	Description	Skills required
	IAM role that you created previously. For more information, see [Find an IAM role within your Splunk platform instance](#) in the Splunk documentation.	

Configure Splunk access to CloudWatch

Task	Description	Skills required
Configure the retrieval of Network Firewall logs from CloudWatch Logs.	1. In the Splunk dashboard, navigate to **Splunk Add-on for AWS**. 2. Choose **Input**. 3. Choose **Create New Input**. 4. In the list, choose **Custom Data Type**, and then choose **CloudWatch Logs**. 5. Provide the **Name**, **AWS Account**, **AWS Region**, and **Log Group** for your Network Firewall logs. 6. Choose **Save**. By default, Splunk fetches the log data every 10 minutes. This is a configurable parameter under **Advanced Settings**. For more informati on, see [Configure a CloudWatch Logs input using](#)	Splunk administrator

Task	Description	Skills required
	Splunk Web in the Splunk documentation.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Configure the retrieval of Network Firewall metrics from CloudWatch.</td><td>
1. In the Splunk dashboard, navigate to Splunk Add-on for AWS.
2. Choose Input.
3. Choose Create New Input.
4. In the list, choose CloudWatch.
5. Provide the Name, AWS Account, and AWS Region for your Network Firewall metrics.
6. Next to Metric Configura tion, choose Edit in advanced mode.
7. (Optional) Delete all preconfigured namespace s.
8. Choose Add Namespace , and then name it AWS/ NetworkFirewall.
9. In Dimension Value, add the following.
<pre><code>[{"AvailabilityZon
e":[".*"],"Engine"
:[".*"],"FirewallN
ame":[".*"]}]</code></pre>
10. For Metrics, choose All.
11. For Metric Statistics, choose Sum.
12. Choose OK.
13. Choose Save.
</td><td>Splunk administrator</td></tr>
</table>

Task	Description	Skills required
	By default, Splunk fetches the metric data every 5 minutes. This is a configurable parameter under **Advanced Settings**. For more information, see [Configure a CloudWatch input using Splunk Web](#) in the Splunk documentation.	

Create Splunk visualizations by using queries

Task	Description	Skills required
View the top source IP addresses.	1. In the Splunk dashboard, navigate to **Search & Reporting**. 2. In the **enter search here** box, enter the following. `sourcetype="aws:cloudwatchlogs" | top event.src_ip` This query displays a table of the source IP addresses with the most traffic, in descending order. 3. For a graphical representation, choose **Visualization**.	Splunk administrator
View packet statistics.	1. In the Splunk dashboard, navigate to **Search & Reporting**.	Splunk administrator

Task	Description	Skills required
	2. In the **enter search here** box, enter the following. `sourcetype="aws:cloudwatch"| timechart sum(Sum) by metric_name` This query displays a table of the metrics `DroppedPackets` , `PassedPackets` , and `ReceivedPackets` per minute. 3. For a graphical representation, choose **Visualization**.	
View the most-used source ports.	1. In the Splunk dashboard, navigate to **Search & Reporting**. 2. In the **enter search here** box, enter the following. `sourcetype="aws:cloudwatchlogs" | top event.dest_port` This query displays a table of the source ports with the most traffic, in descending order. 3. For a graphical representation, choose **Visualization**.	Splunk administrator

Related resources

AWS documentation

- Creating a role to delegate permissions to an AWS service (IAM documentation)
- Creating IAM policies (IAM documentation)
- Logging and monitoring in AWS Network Firewall (Network Firewall documentation)
- Route table configurations for AWS Network Firewall (Network Firewall documentation)

AWS blog posts

- AWS Network Firewall deployment models

AWS Marketplace

- Splunk Enterprise Amazon Machine Image (AMI)

More patterns

- Access a bastion host by using Session Manager and Amazon EC2 Instance Connect
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer
- Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer
- Centralize DNS resolution by using AWS Managed Microsoft AD and on-premises Microsoft Active Directory
- Check for single-host network entries in security group ingress rules for IPv4 and IPv6
- Create a portal for micro-frontends by using AWS Amplify, Angular, and Module Federation
- Deploy an Amazon API Gateway API on an internal website using private endpoints and an Application Load Balancer
- Deploy detective attribute-based access controls for public subnets by using AWS Config
- Deploy preventative attribute-based access controls for public subnets
- Enable encrypted connections for PostgreSQL DB instances in Amazon RDS
- Extend VRFs to AWS by using AWS Transit Gateway Connect
- Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud
- Migrate NGINX Ingress Controllers when enabling Amazon EKS Auto Mode
- Preserve routable IP space in multi-account VPC designs for non-workload subnets
- Prevent internet access at the account level by using a service control policy
- Send alerts from AWS Network Firewall to a Slack channel
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront
- Set up disaster recovery for Oracle JD Edwards EnterpriseOne with AWS Elastic Disaster Recovery
- Use BMC Discovery queries to extract migration data for migration planning
- Use Network Firewall to capture the DNS domain names from the Server Name Indication for outbound traffic

Content delivery

Topics

- Send AWS WAF logs to Splunk by using AWS Firewall Manager and Amazon Data Firehose
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront
- More patterns

Send AWS WAF logs to Splunk by using AWS Firewall Manager and Amazon Data Firehose

Michael Friedenthal, Aman Kaur Gandhi, and JJ Johnson, Amazon Web Services

Summary

Historically, there were two ways to move data into Splunk: a push or a pull architecture. A *pull architecture* offers delivery data guarantees through retries, but it requires dedicated resources in Splunk that poll data. Pull architectures usually are not real time because of the polling. A *push architecture* in typically has lower latency, is more scalable, and reduces operational complexity and costs. However, it doesn't guarantee delivery and typically requires agents.

Splunk integration with Amazon Data Firehose delivers real-time streaming data to Splunk through an HTTP event collector (HEC). This integration provides the advantages of both push and pull architectures—it guarantees data delivery through retries, is near real-time, and is low latency and low complexity. The HEC quickly and efficiently sends data over HTTP or HTTPS directly to Splunk. HECs are token-based, which eliminates the need to hardcode credentials in an application or in supporting files.

In an AWS Firewall Manager policy, you can configure logging for all of the AWS WAF web ACL traffic in all of your accounts, and you can then use a Firehose delivery stream to send that log data to Splunk for monitoring, visualization, and analysis. This solution provides the following benefits:

- Central management and logging for AWS WAF web ACL traffic in all of your accounts
- Splunk integration with a single AWS account
- Scalability
- Near real-time delivery of log data
- Cost optimization through the use of a serverless solution, so you don't have to pay for unused resources.

Prerequisites and limitations

Prerequisites

- An active AWS account that is part of an organization in AWS Organizations.
- You must have the following permissions to enable logging with Firehose:

  - `iam:CreateServiceLinkedRole`
  - `firehose:ListDeliveryStreams`
  - `wafv2:PutLoggingConfiguration`
- AWS WAF and its web ACLs must be configured. For instructions, see [Getting started with AWS WAF](#).
- AWS Firewall Manager must be setup. For instructions, see [AWS Firewall Manager prerequisites](#).
- The Firewall Manager security policies for AWS WAF must be configured. For instructions, see [Getting started with AWS Firewall ManagerAWS WAF policies](#).
- Splunk must be setup with a public HTTP endpoint that can be reached by Firehose.

Limitations

- The AWS accounts must be managed in a single organization in AWS Organizations.
- The web ACL must be in the same Region as the delivery stream. If you are capturing logs for Amazon CloudFront, create the Firehose delivery stream in the US East (N. Virginia) Region, `us-east-1`.
- The Splunk add-on for Firehose is available for paid Splunk Cloud deployments, distributed Splunk Enterprise deployments, and single-instance Splunk Enterprise deployments. This add-on is not supported for free trial Splunk Cloud deployments.

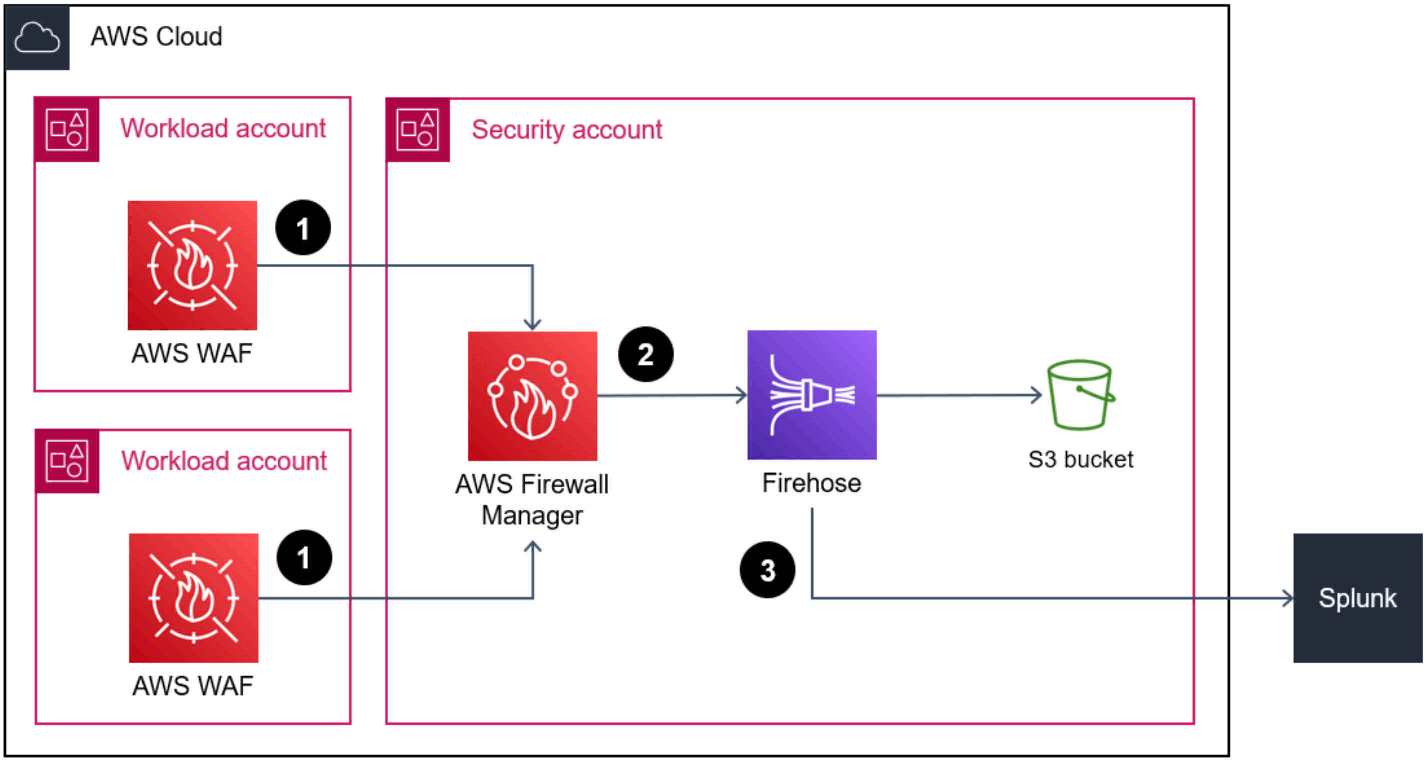
Architecture

Target technology stack

- Firewall Manager
- Firehose
- Amazon Simple Storage Service (Amazon S3)
- AWS WAF
- Splunk

Target architecture

The following image shows how you can use Firewall Manager to centrally log all AWS WAF data and send it to Splunk through Firehose.



1. The AWS WAF web ACLs send firewall log data to Firewall Manager.
2. Firewall Manager sends the log data to Firehose.
3. The Firehose delivery stream forwards the log data to Splunk and to an S3 bucket. The S3 bucket acts as a backup in the event of an error with the Firehose delivery stream.

Automation and scale

This solution is designed to scale and accommodate all AWS WAF web ALCs within the organization. You can configure all web ACLs to use the same Firehose instance. However, if you want to set up and use multiple Firehose instances, you can.

Tools

AWS services

- [AWS Firewall Manager] is a security management service that helps you to centrally configure and manage firewall rules across your accounts and applications in AWS Organizations.
- [Amazon Data Firehose] helps you deliver real-time [streaming data] to other AWS services, custom HTTP endpoints, and HTTP endpoints owned by supported third-party service providers, such as Splunk.

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS WAF is a web application firewall that helps you monitor HTTP and HTTPS requests that are forwarded to your protected web application resources.

Other tools

- Splunk helps you monitor, visualize, and analyze log data.

Epics

Configure Splunk

Task	Description	Skills required
Install the Splunk App for AWS.	1. Sign in to your Splunk heavy forwarder. The default URL is `http://<IP address>:8000` . 2. In the left navigation, next to **Apps**, choose the gear button. 3. Choose **Browse more apps**. 4. Search for **AWS**. 5. For **Splunk App for AWS**, choose **Install**. 6. Enter your Splunk.com sign-in credentials, accept the terms and conditions, and then choose **Login and Install**. 7. Choose **Done**.	Security administrator, Splunk administrator
Install the add-on for AWS WAF.	Repeat the previous instructions to install the **AWS Web**	Security administrator, Splunk administrator

Task	Description	Skills required
	Application Firewall Add-on for Splunk.	

Task	Description	Skills required
Install and configure the Splunk add-on for Firehose.	1. Install and configure the Splunk add-on for Firehose. As part of the installation and configura tion, if necessary for your Splunk platform, you set up an HTTP Event Collector and prepare the infrastructure to send the log data to your indexers. See the instructions that correspond to your Splunk deployment: • Splunk Cloud deploymen t (Splunk documenta tion) • Distributed Splunk Enterprise deployment (Splunk documentation) • Single-instance Splunk Enterprise deployment (Splunk documentation) **Important** Stop this procedure after you have installed and configured the Splunk add-on. Do not proceed with the instructi ons for configuri ng Firehose to send	Security administrator, Splunk administrator

Task	Description	Skills required
	data to the Splunk platform. 2. Make note of the HTTP event collector token and the HTTP endpoint. You need this value later, when you configure the delivery stream.	

Create the Firehose delivery stream

Task	Description	Skills required
Grant Firehose access to a Splunk destination.	Configure the access policy that permits Firehose to access a Splunk destinati on and back up the log data to an S3 bucket. For more information, see [Grant Firehose access to a Splunk destination]().	Security administrator
Create a Firehose delivery stream.	In the same account where you manage the web ACLs for AWS WAF, create a delivery stream in Firehose. You are required to have an IAM role when creating a delivery stream. Firehose assumes that IAM role and gains access to the specified S3 bucket. For instructions, see [Creating a delivery stream](). Note the following:	Security administrator

Task	Description	Skills required
	• The delivery stream name must start with `aws-waf-logs-` . • For the source, choose **Direct PUT**. • For **S3 backup mode**, choose **Backup all events**, and then choose an existing bucket or create a new one. • For the destination, follow the instructions in [Choose Splunk for your destination]() in the Firehose documentation. For information about the values for Splunk endpoints and endpoint types, see [Configure Amazon Data Firehose]() in the Splunk documentation. Repeat this process for each token that you configured in the HTTP event collector.	
Test the delivery stream.	Test the delivery stream to validate that it is properly configured. For instructions, see [Test using Splunk as the destination]() in the Firehose documentation.	Security administrator

Configure Firewall Manager to log data

Task	Description	Skills required
Configure the Firewall Manager policies.	The Firewall Manager policies must be configured to enable logging and to forward logs to the correct Firehose delivery stream. For more information and instructions, see Configuring logging for an AWS WAF policy.	Security administrator

Related resources

AWS resources

- Logging web ACL traffic (AWS WAF documentation)
- Configuring logging for an AWS WAF policy (AWS WAF documentation)
- Tutorial: Sending VPC Flow Logs to Splunk Using Amazon Data Firehose (Firehose documentation)
- How do I push VPC flow logs to Splunk using Amazon Data Firehose? (AWS Knowledge Center)
- Power data ingestion into Splunk using Amazon Data Firehose (AWS blog post)

Splunk documentation

- Splunk Add-on for Amazon Data Firehose

Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront

Angel Emmanuel Hernandez Cebrian, Amazon Web Services

Summary

When you serve static content that is hosted on Amazon Web Services (AWS), the recommended approach is to use an Amazon Simple Storage Service (S3) bucket as the origin and use Amazon CloudFront to distribute the content. This solution has two primary benefits: the convenience of caching static content at edge locations, and the ability to define web access control lists (web ACLs) for the CloudFront distribution, which helps you secure requests to the content with minimal configuration and administrative overhead.

However, there is a common architectural limitation to the standard, recommended approach. In some environments, you want virtual firewall appliances deployed in a virtual private cloud (VPC) to inspect all content, including static content. The standard approach doesn’t route traffic through the VPC for inspection. This pattern provides an alternative architectural solution. You still use a CloudFront distribution to serve static content in an S3 bucket, but the traffic is routed through the VPC by using an Application Load Balancer. An AWS Lambda function then retrieves and returns the content from the S3 bucket.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Static website content hosted in an S3 bucket.

Limitations

- The resources in this pattern must be in a single AWS Region, but they can be provisioned in different AWS accounts.
- Limits apply to the maximum request and response size that the Lambda function can receive and send, respectively. For more information, see *Limits* in Lambda functions as targets (Elastic Load Balancing documentation).
- It's important to find a good balance between performance, scalability, security, and cost-effectiveness when using this approach. Despite the high scalability of Lambda, if the number

of concurrent Lambda invocations exceeds the maximum quota, some requests are throttled. For more information, see Lambda quotas (Lambda documentation). You also need to consider pricing when using Lambda. To minimize Lambda invocations, make sure that you properly define the cache for the CloudFront distribution. For more information, see Optimizing caching and availability (CloudFront documentation).

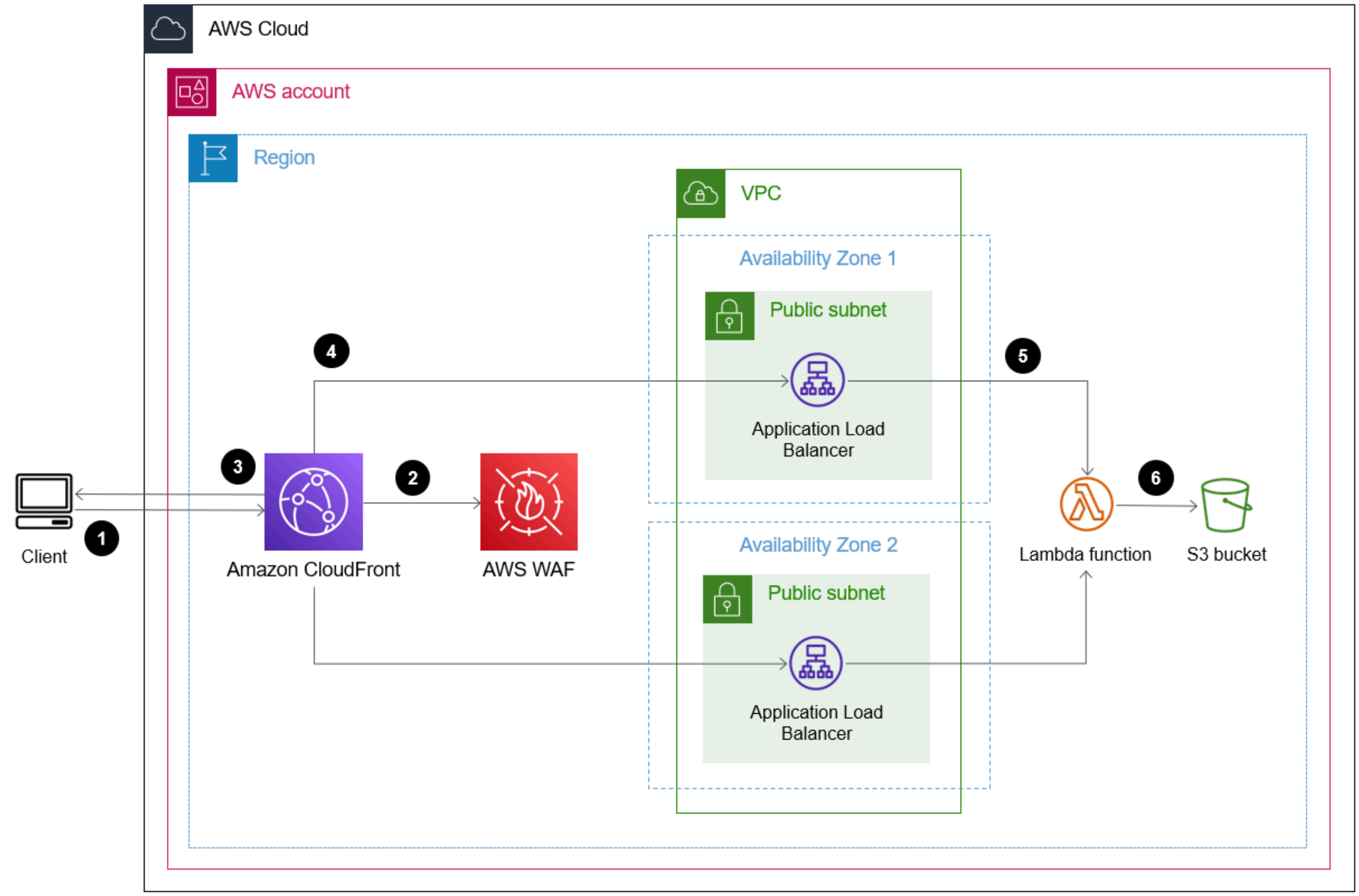
Architecture

Target technology stack

- CloudFront
- Amazon Virtual Private Cloud (Amazon VPC)
- Application Load Balancer
- Lambda
- Amazon S3

Target architecture

The following image shows the suggested architecture when you need to use CloudFront to serve static content from an S3 bucket through a VPC.



1. The client requests the URL of CloudFront distribution to get a particular website file in the S3 bucket.
2. CloudFront sends the request to AWS WAF. AWS WAF filters the request by using the web ACLs applied to the CloudFront distribution. If the request is determined to be valid, the flow continues. If the request is determined to be invalid, the client receives a 403 error.
3. CloudFront checks its internal cache. If there is a valid key matching the incoming request, the associated value is sent back to the client as a response. If not, the flow continues.
4. CloudFront forwards the request to the URL of the specified Application Load Balancer.
5. The Application Load Balancer has a listener associated with a target group based on a Lambda function. The Application Load Balancer invokes the Lambda function.
6. The Lambda function connects to the S3 bucket, perform a `GetObject` operation on it, and returns the content as a response.

Automation and scale

To automate the deployment of static content using this approach, create CI/CD pipelines to update the Amazon S3 buckets that host websites.

The Lambda function scales automatically to handle the concurrent requests, within the quotas and limitations of the service. For more information, see Lambda function scaling and Lambda quotas (Lambda documentation). For the other AWS services and features, such as CloudFront and the Application Load Balancer, AWS scales these automatically.

Tools

- Amazon CloudFront speeds up distribution of your web content by delivering it through a worldwide network of data centers, which lowers latency and improves performance.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. In this pattern, you use an Application Load Balancer provisioned through Elastic Load Balancing to direct traffic to the Lambda function.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Use CloudFront to serve static content from Amazon S3 through a VPC

Task	Description	Skills required
Create a VPC.	Create a VPC for hosting the resources deployed in this pattern, such as the Application Load Balancer and the Lambda function. For instructions, see Create a	Cloud architect

Task	Description	Skills required
	VPC (Amazon VPC documenta tion).	
Create an AWS WAF web ACL.	Create an AWS WAF web ACL. Later in this pattern, you apply this web ACL to the CloudFront distribut ion. For instructions, see Creating a web ACL (AWS WAF documentation).	Cloud architect
Create the Lambda function.	Create the Lambda function that serves the static content hosted in the S3 bucket as a website. Use the code provided in the Additional information section of this pattern. Customize the code to identify your target S3 bucket.	General AWS

Upload the Lambda function.	Enter the following command to upload the Lambda function code to a .zip file archive in Lambda.
	`aws lambda update-function-code \`
	`--function-name  \`
	`--zip-file fileb://lambda-alb-s3-website.zip`
		General AWS

Task	Description	Skills required
Create an Application Load Balancer.	Create an internet-facing Application Load Balancer that points to the Lambda function. For instructions, see [Create a target group for the Lambda function](#) (Elastic Load Balancing documentation). For a high-availability configuration, create the Application Load Balancer and attach it to private subnets in different Availability Zones.	Cloud architect

Task	Description	Skills required

Create a CloudFront distribution.	Create a CloudFront distribution that points to the Application Load Balancer you created.
	
	1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v3/home.
	2. Choose **Create Distribution**.
	3. On the first page of the **Create Distribution Wizard**, in the **Web** section, choose **Get Started**.
	4. Specify settings for your distribution. For more information, see [Values that you specify when you create or update a distribution](). Note the following:
	  a. Set the Application Load Balancer as the origin.
	  b. In **Distribution settings**, choose existing web ACLs that you want to apply through AWS WAF. For more information, see [AWS WAF web ACL]().
	5. Save your changes.
		Cloud architect

Task	Description	Skills required
	6. After CloudFront creates your distribution, the value of the **Status** column for your distribution changes from **InProgress** to **Deployed**. If you chose to enable the distribution, it will be ready to process requests after the status switches to **Deployed**.	

Related resources

AWS documentation

- Optimizing caching and availability (CloudFront documentation)
- Lambda functions as targets (Elastic Load Balancing documentation)
- Lambda quotas (Lambda documentation)

AWS service websites

- Application Load Balancer
- Lambda
- CloudFront
- Amazon S3
- AWS WAF
- Amazon VPC

Additional information

Code

The following example Lambda function is written in Node.js. This Lambda function acts as a web server that performs a `GetObject` operation to an S3 bucket that contains the website resources.

```
/**

 * This is an AWS Lambda function created for demonstration purposes.

 * It retrieves static assets from a defined Amazon S3 bucket.

 * To make the content available through a URL, use an Application Load Balancer with a
 Lambda integration.
 *
 * Set the S3_BUCKET environment variable in the Lambda function definition.
 */

var AWS = require('aws-sdk');

exports.handler = function(event, context, callback) {

    var bucket = process.env.S3_BUCKET;
    var key = event.path.replace('/', '');

    if (key == '') {
        key = 'index.html';
    }

    // Fetch from S3
    var s3 = new AWS.S3();
    return s3.getObject({Bucket: bucket, Key: key},
       function(err, data) {

            if (err) {
                return err;
            }

            var isBase64Encoded = false;
            var encoding = 'utf8';

            if (data.ContentType.indexOf('image/') > -1) {
                isBase64Encoded = true;
                encoding = 'base64'
            }

            var resp = {
                statusCode: 200,
                headers: {
```

```
                'Content-Type': data.ContentType,
            },
            body: new Buffer(data.Body).toString(encoding),
            isBase64Encoded: isBase64Encoded
        };

        callback(null, resp);
    }
  );
};
```

More patterns

- Check an Amazon CloudFront distribution for access logging, HTTPS, and TLS version
- Deploy a gRPC-based application on an Amazon EKS cluster and access it with an Application Load Balancer
- Deploy preventative attribute-based access controls for public subnets
- Deploy resources in an AWS Wavelength Zone by using Terraform
- Deploy the Security Automations for AWS WAF solution by using Terraform
- Set up a serverless cell router for a cell-based architecture
- Use Amazon Q Developer as a coding assistant to increase your productivity
- View AWS Network Firewall logs and metrics by using Splunk

Databases & storage

Topics

- Databases
- Storage & backup

Databases

Topics

- Access on-premises Microsoft SQL Server tables from Microsoft SQL Server on Amazon EC2 using linked servers
- Add HA to Oracle PeopleSoft on Amazon RDS Custom by using a read replica
- Assess query performance for migrating SQL Server databases to MongoDB Atlas on AWS
- Automate blue/green deployments of Amazon Aurora global databases by using IaC principles
- Automate the replication of Amazon RDS instances across AWS accounts
- Automate database tasks in SQL Server Express edition running on Amazon EC2 by using AWS Lambda and Task Scheduler
- Automate cross-Region failover and failback by using DR Orchestrator Framework
- Automatically back up SAP HANA databases using Systems Manager and EventBridge
- Automatically generate a PynamoDB model and CRUD functions for Amazon DynamoDB by using a Python application
- Block public access to Amazon RDS by using Cloud Custodian
- Configure cross-account access to Amazon DynamoDB
- Configure read-only routing in an Always On availability group in SQL Server on AWS
- Connect by using an SSH tunnel in pgAdmin
- Convert JSON Oracle queries into PostgreSQL database SQL
- Copy Amazon DynamoDB tables across accounts using AWS Backup
- Copy Amazon DynamoDB tables across accounts using a custom implementation
- Create detailed cost and usage reports for Amazon RDS and Amazon Aurora
- Deploy SQL Server failover cluster instances on Amazon EC2 and Amazon FSx by using Terraform
- Emulate Oracle PL/SQL associative arrays in Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL
- Enable encrypted connections for PostgreSQL DB instances in Amazon RDS
- Encrypt an existing Amazon RDS for PostgreSQL DB instance
- Enforce automatic tagging of Amazon RDS databases at launch
- Estimate the cost of a DynamoDB table for on-demand capacity
- Estimate storage costs for an Amazon DynamoDB table

- Estimate the Amazon RDS engine size for an Oracle database by using AWR reports
- Export Amazon RDS for SQL Server tables to an S3 bucket by using AWS DMS
- Handle anonymous blocks in Dynamic SQL statements in Aurora PostgreSQL
- Handle overloaded Oracle functions in Aurora PostgreSQL-Compatible
- Help enforce DynamoDB tagging
- Implement cross-Region disaster recovery with AWS DMS and Amazon Aurora
- Migrate Oracle functions and procedures that have more than 100 arguments to PostgreSQL
- Migrate SAP HANA to AWS using SAP HSR with the same hostname
- Migrate Microsoft SQL Server Always On availability group using AWS Application Migration Service
- Migrate SQL Server to AWS using distributed availability groups
- Migrate from Oracle 8i or 9i to Amazon RDS for Oracle using SharePlex and AWS DMS
- Monitor Amazon Aurora for instances without encryption
- Monitor Oracle GoldenGate logs by using Amazon CloudWatch
- Replatform Oracle Database Enterprise Edition to Standard Edition 2 on Amazon RDS for Oracle
- Replicate mainframe databases to AWS by using Precisely Connect
- Schedule jobs for Amazon RDS for PostgreSQL and Aurora PostgreSQL by using Lambda and Secrets Manager
- Send notifications for an Amazon RDS for SQL Server database instance by using an on-premises SMTP server and Database Mail
- Set up disaster recovery for SAP on IBM Db2 on AWS
- Set up a CI/CD pipeline for database migration by using Terraform
- Set up an HA/DR architecture for Oracle E-Business Suite on Amazon RDS Custom with an active standby database
- Set up data replication between Amazon RDS for MySQL and MySQL on Amazon EC2 using GTID
- Transition roles for an Oracle PeopleSoft application on Amazon RDS Custom for Oracle
- Unload data from an Amazon Redshift cluster across accounts to Amazon S3
- Database migration patterns by workload
- More patterns

Access on-premises Microsoft SQL Server tables from Microsoft SQL Server on Amazon EC2 using linked servers

Tirumala Dasari and Eduardo Valentim, Amazon Web Services

Summary

This pattern describes how to access on-premises Microsoft SQL Server database tables running on Microsoft Windows, from Microsoft SQL Server databases running or hosted on Amazon Elastic Compute Cloud (Amazon EC2) Windows or Linux instances by using linked servers.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Amazon EC2 with Microsoft SQL Server running on Amazon Linux AMI (Amazon Machine Image)
- AWS Direct Connect between the on-premises Microsoft SQL Server (Windows) server and the Windows or Linux EC2 instance

Product versions

- SQL Server 2016 or later

Architecture

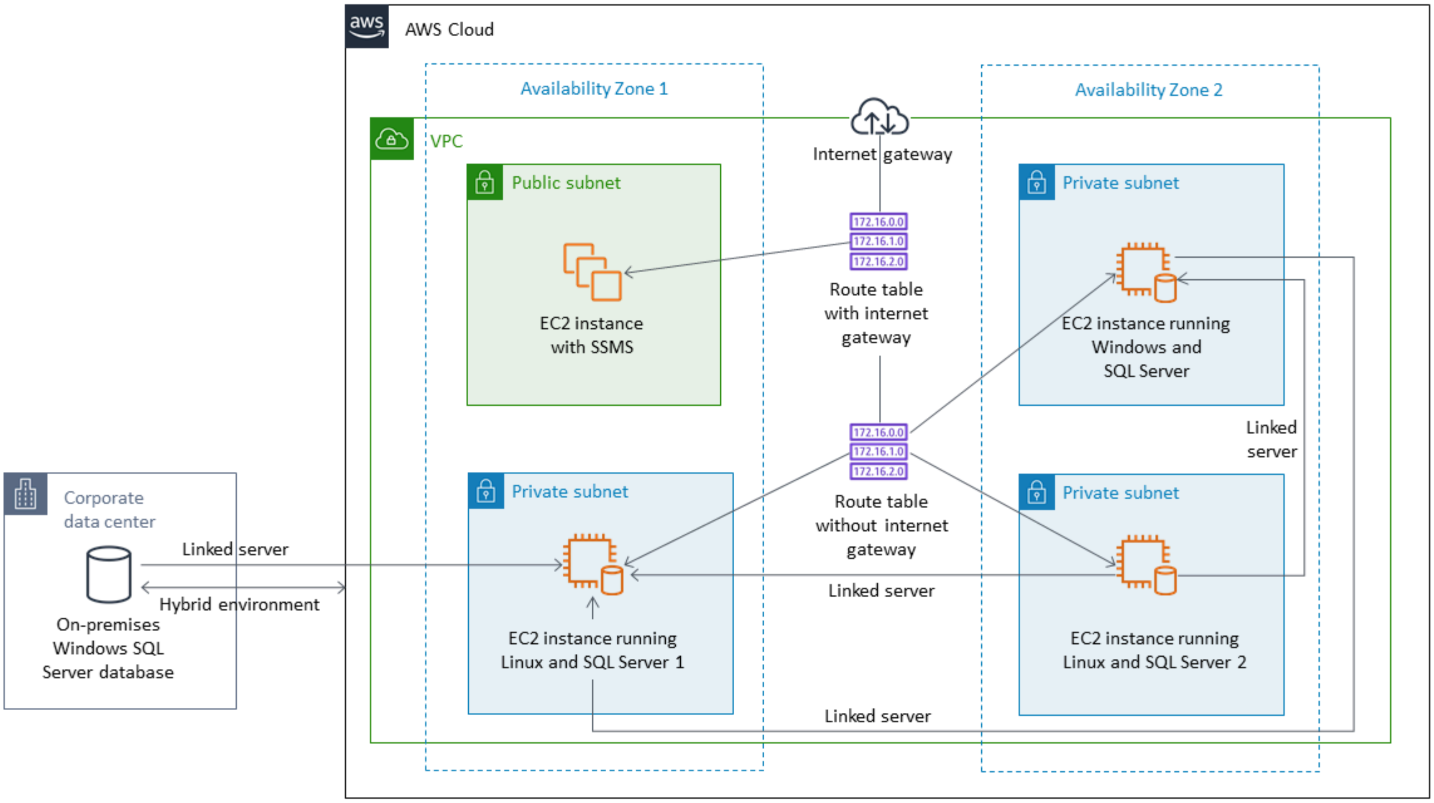
Source technology stack

- On-premises Microsoft SQL Server database running on Windows
- Amazon EC2 with Microsoft SQL Server running on Windows AMI or Linux AMI

Target technology stack

- Amazon EC2 with Microsoft SQL Server running on Amazon Linux AMI
- Amazon EC2 with Microsoft SQL Server running on Windows AMI

Source and target database architecture



Tools

- [Microsoft SQL Server Management Studio (SSMS)] is an integrated environment for managing a SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

Epics

Change authentication mode to Windows for SQL Server in Windows SQL Server

Task	Description	Skills required
Connect to Windows SQL Server through SSMS.		DBA
Change the authentication mode to Windows in SQL Server from the context		DBA

Task	Description	Skills required
(right-click) menu for the Windows SQL Server instance.		

Restart the Windows MSSQL service

Task	Description	Skills required
Restart the SQL service.	1. In SSMS Object Explorer, choose the SQL Server instance. 2. Open the context (right-cl ick) menu. 3. Choose **Restart**.	DBA

Create new login and choose databases to access in Windows SQL Server

Task	Description	Skills required
In the Security tab, open the context (right-click) menu for Login and select a new login.		DBA
In the General tab, choose SQL Server authentication, enter a user name, enter the password, and then confirm the password and clear the option for changing the password at the next login.		DBA
In the Server Roles tab, choose Public.		DBA

Task	Description	Skills required
In the User Mapping tab, choose the database and schema you want to access, and then highlight the database to select database roles.	Select public and db_datareader to access data from the database tables.	DBA
Choose OK to create a user.		DBA

Add Windows SQL Server IP to Linux SQL Server host file

Task	Description	Skills required
Connect to the Linux SQL Server box through the terminal window.		DBA
Open the /etc/hosts file and add the IP address of the Windows machine with SQL Server.		DBA
Save the hosts file.		DBA

Create linked server on Linux SQL Server

Task	Description	Skills required
Create a linked server by using the stored procedures master.sys.sp_addlinkedserver and master.dbo.sp_addlinkedsrvlogin.	For more information about using these stored procedures, see the *Additional information* section.	DBA, Developer

Verify the created linked server and databases in SSMS

Task	Description	Skills required
In Linux SQL Server in SSMS, go to Linked Servers and refresh.		DBA
Expand the created linked servers and catalogs in the left pane.	You'll see the selected SQL Server databases with tables and views.	DBA

Verify that you can access Windows SQL Server database tables

Task	Description	Skills required
In the SSMS query window, run the query: "select top 3 * from [sqllin].dms_sample_win.dbo.mlb_data".	Note that the FROM clause uses a four-part syntax: computer.database.schema.table (e.g., SELECT name "SQL2 databases" FROM [sqllin].master.sys.databases). In our example, we created an alias for SQL2 in the hosts file, so you don't need to enter the actual NetBIOS name between the square brackets. If you do use the actual NetBIOS names, note that AWS defaults to NetBIOS names like Win-xxxx, and SQL Server requires square brackets for names with dashes.	DBA, Developer

Related resources

- [Release notes for SQL Server on Linux]

Additional information

Using stored procedures to create linked servers

SSMS doesn't support the creation of linked servers for Linux SQL Server, so you have to use these stored procedures to create them:

```
EXEC master.sys.sp_addlinkedserver @server= N'SQLLIN' , @srvproduct= N'SQL Server'
EXEC master.dbo.sp_addlinkedsrvlogin
 @rmtsrvname=N'SQLLIN',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassword='Te
```

Note 1: Enter the sign-in credentials that you created earlier in Windows SQL Server in the stored procedure `master.dbo.sp_addlinkedsrvlogin`.

Note 2: `@server` name `SQLLIN` and host file entry name `172.12.12.4 SQLLIN` should be the same.

You can use this process to create linked servers for the following scenarios:

- Linux SQL Server to Windows SQL Server through a linked server (as specified in this pattern)
- Windows SQL Server to Linux SQL Server through a linked server
- Linux SQL Server to another Linux SQL Server through a linked server

Add HA to Oracle PeopleSoft on Amazon RDS Custom by using a read replica

sampath kathirvel, Amazon Web Services

Summary

To run the Oracle PeopleSoft enterprise resource planning (ERP) solution on Amazon Web Services (AWS), you can use Amazon Relational Database Service (Amazon RDS) or Amazon RDS Custom for Oracle, which supports legacy, custom, and packaged applications that require access to the underlying operating system and database environment. For key factors to consider when planning a migration, see Oracle database migration strategies in AWS Prescriptive Guidance.

As of this writing, RDS Custom for Oracle doesn't support the Multi-AZ option, which is available for Amazon RDS for Oracle as an HA solution using storage replication. Instead, this pattern achieves HA by using a standby database that creates and maintains a physical copy of the primary database. The pattern focuses on the steps to run a PeopleSoft application database on Amazon RDS Custom with HA by using Oracle Data Guard to set up a read replica.

This pattern also changes the read replica to read-only mode. Having your read replica in read-only mode provides additional benefits:

- Offloading read-only workloads from the primary database
- Enabling automatic repair of corrupted blocks by retrieving healthy blocks from the standby database using the Oracle Active Data Guard feature
- Using the Far Sync capability to keep the remote standby database in sync without the performance overhead associated with long-distance redo log transmission.

Using a replica in read-only mode requires the Oracle Active Data Guard option, which comes at an extra cost because it is a separately licensed feature of Oracle Database Enterprise Edition.

Prerequisites and limitations

Prerequisites

- An existing PeopleSoft application on Amazon RDS Custom. If you don't have an application, see the pattern Migrate Oracle PeopleSoft to Amazon RDS Custom.

- A single PeopleSoft application tier. However, you can adapt this pattern to work with multiple application tiers.
- Amazon RDS Custom configured with at least 8 GB of swap space.
- An Oracle Active Data Guard database license for converting the read replica into read-only mode and using it for offloading reporting tasks to the standby. For more information, see the Oracle Technology Commercial Price List.

Limitations

- General limitations and unsupported configurations for RDS Custom for Oracle
- Limitations associated with Amazon RDS Custom for Oracle read replicas

Product versions

- For Oracle Database versions supported by Amazon RDS Custom, see RDS Custom for Oracle.
- For Oracle Database instance classes supported by Amazon RDS Custom, see DB instance class support for RDS Custom for Oracle.

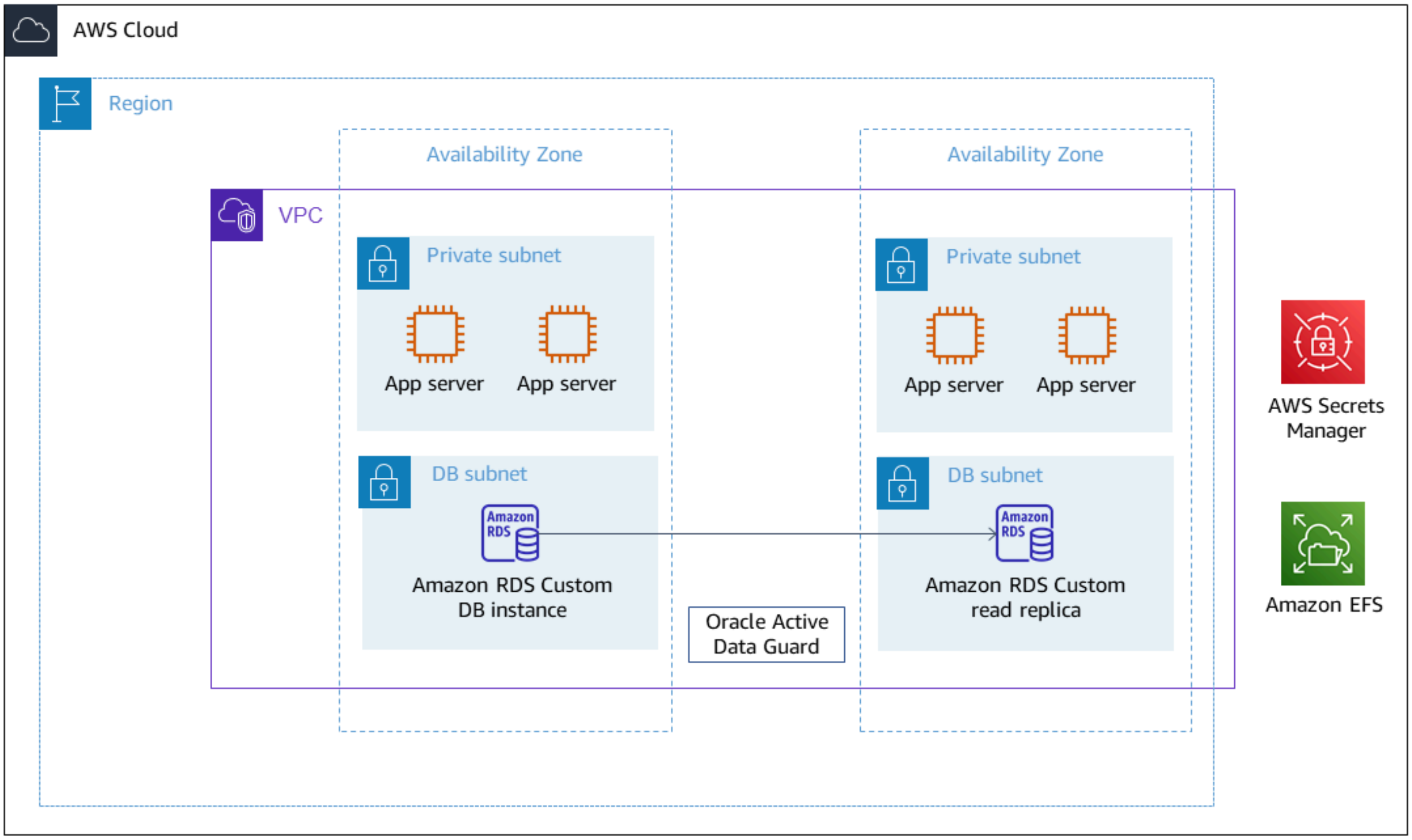
Architecture

Target technology stack

- Amazon RDS Custom for Oracle
- AWS Secrets Manager
- Oracle Active Data Guard
- Oracle PeopleSoft application

Target architecture

The following diagram shows an Amazon RDS Custom DB instance and an Amazon RDS Custom read replica. The read replica uses Oracle Active Data Guard to replicate to another Availability Zone. You can also use the read replica to offload read traffic on the primary database and for reporting purposes.



For a representative architecture using Oracle PeopleSoft on AWS, see Set up a highly available PeopleSoft architecture on AWS.

Tools

AWS services

- Amazon RDS Custom for Oracle is a managed database service for legacy, custom, and packaged applications that require access to the underlying operating system and database environment.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. In this pattern, you retrieve the database user passwords from Secrets Manager for `RDS_DATAGUARD` with the secret name `do-not-delete-rds-custom-+<<RDS Resource ID>>+-dg`.

Other Tools

- Oracle Data Guard helps you create, maintain, manage, and monitor standby databases.

Best practices

To work toward a zero data loss (RPO=0) objective, use the `MaxAvailability` Data Guard protection mode, with the redo transport `SYNC+NOAFFIRM` setting for better performance. For more information about selecting the database protection mode, see the *Additional information* section.

Epics

Create the read replica

Task	Description	Skills required
Create the read replica.	To create a read replica of the Amazon RDS Custom DB instance, follow the instructions in the [Amazon RDS documentation](#) and use the Amazon RDS Custom DB instance that you created (see the *Prerequisites* section) as the source database. By default, the Amazon RDS Custom read replica is created as a physical standby and is in the mounted state. This is intentional to ensure compliance with the Oracle Active Data Guard license. This pattern includes code for setting up a multitenant container database (CDB) or a non-CDB instance.	DBA

Change Oracle Data Guard protection mode to MaxAvailability

Task	Description	Skills required
Access the Data Guard broker configuration on the primary database.	In this example, the Amazon RDS Custom read replica is `RDS_CUSTOM_ORCL_D` for the Non-CDB instance and `RDS_CUSTOM_RDSCDB_B` for the CDB instance. The databases for Non-CDB are `orcl_a` (primary) and `orcl_d` (standby). The database names for CDB are `rdscdb_a` (primary) and `rdscdb_b` (standby). You can connect to the RDS Custom read replica directly or through the primary database. You can find the net service name for your database in the `tnsnames.ora` file located in the `$ORACLE_HOME/network/admin` directory. RDS Custom for Oracle automatically populates these entries for your primary database and your read replicas. The password for the `RDS_DATAGUARD` user is stored in AWS Secrets Manager, with secret name do-not-delete-rds-custom-+<<RDS	DBA

Task	Description	Skills required
	`Resource ID>>+-dg`. For more information on how to connect to an RDS Custom instance using the SSH (Secure Shell) key retrieved from Secrets Manager, see [Connecting to your RDS Custom DB instance using SSH](). To access the Oracle Data Guard broker configuration through the Data Guard command line (`dgmgrl`), use the following code. **Non-CDB**	

```
$ dgmgrl RDS_DATAGUARD@RDS_CUSTOM_ORCL_D
DGMGRL for Linux:
 Release 19.0.0.0.0 -
 Production on Fri Sep
 30 22:44:49 2022
Version 19.10.0.0.0
Copyright (c) 1982,
 2019, Oracle and/or its
 affiliates. All rights
 reserved.
Welcome to DGMGRL, type
 "help" for information.
Password:
Connected to "ORCL_D"
Connected as SYSDG.
DGMGRL>
DGMGRL> show database
 orcl_d
```

Task	Description	Skills required
	``` Database - orcl_d Role: PHYSICAL STANDBY Intended State: APPLY- ON Transport Lag: 0 seconds (computed 0 seconds ago) Apply Lag: 0 seconds (computed 0 seconds ago) Average Apply Rate: 11.00 KByte/s Instance(s): ORCL SUCCESS DGMGRL> ``` **CDB** ``` -bash-4.2$ dgmgrl C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_B DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Jan 11 20:24:11 2023 Version 19.16.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Welcome to DGMGRL, type "help" for informati on. Password: Connected to "RDSCDB_B " Connected as SYSDG. DGMGRL> DGMGRL> show database rdscdb_b ```	

Task	Description	Skills required

	<pre>Database - rdscdb_b
	  Role:
	 PHYSICAL STANDBY
	  Intended State:
	 APPLY-ON
	  Transport Lag:
	 0 seconds (computed 1
	 second ago)
	  Apply Lag:
	 0 seconds (computed 1
	 second ago)
	  Average Apply Rate:
	 2.00 KByte/s
	  Real Time Query:
	 OFF
	  Instance(s):
	    RDSCDB
	Database Status:
	SUCCESS
	DGMGRL></pre>	

Task	Description	Skills required

Change the log transport setting by connecting to DGMGRL from the primary node.	Change the log transport mode to `FastSync`, corresponding to the redo transport setting `SYNC+NOAFFIRM`. To ensure that you have valid settings after the role switch, change it for both the primary database and the standby database.
	
	Non-CDB
	
	`DGMGRL>`
	`DGMGRL> edit database orcl_d set property logxptmode=fastsync;`
	`Property "logxptmode" updated`
	`DGMGRL> show database orcl_d LogXptMode;`
	`LogXptMode = 'fastsync'`
	`DGMGRL> edit database orcl_a set property logxptmode=fastsync;`
	`Property "logxptmode" updated`
	`DGMGRL> show database orcl_a logxptmode;`
	`LogXptMode = 'fastsync'`
	`DGMGRL>`
	
	CDB
	
	`DGMGRL> edit database rdscdb_b set property logxptmode=fastsync;DGMGRL> edit database`
		DBA

Task	Description	Skills required

	``
	rdscdb_b set property
	logxptmode=fastsync;
	Property "logxptmode"
	updated
	DGMGRL> show database
	rdscdb_b LogXptMode;
	LogXptMode =
	'fastsync'
	DGMGRL> edit database
	rdscdb_a set property
	logxptmode=fastsync;
	Property "logxptmode"
	updated
	DGMGRL> show database
	rdscdb_a logxptmode;
	LogXptMode =
	'fastsync'
	DGMGRL>
	``	

Task	Description	Skills required

Change the protection mode to MaxAvailability.	Change the protection mode to `MaxAvailability` by connecting to DGMGRL from the primary node.
	
	Non-CDB
	
	`DGMGRL> edit configuration set protection mode as maxavailability;`
	`Succeeded.`
	`DGMGRL> show configuration;`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_a - Primary database`
	`orcl_d - Physical standby database`
	`Fast-Start Failover: Disabled`
	`Configuration Status:`
	`SUCCESS (status updated 38 seconds ago)`
	`DGMGRL>`
	
	CDB
	
	`DGMGRL> show configuration`
	`Configuration - rds_dg`
	`  Protection Mode: MaxAvailability`
	`  Members:`
	`  rdscdb_a - Primary database`
		DBA

Task	Description	Skills required
	``` rdscdb_b - Physical standby database Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 57 seconds ago) DGMGRL> ```	

Change the replica status from mount to read-only and enable redo apply

Task	Description	Skills required
Stop redo apply for the standby database.	The read replica is created in `MOUNT` mode by default. To open it in read-only mode, you first need to turn off redo apply by connecting to `DGMGRL` from the primary or standby node. **Non-CDB** ``` DGMGRL> show database orcl_dDGMGRL> show database orcl_d Database - orcl_d Role: PHYSICAL STANDBY Intended State: APPLY- ON Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago) Average Apply Rate: 11.00 KByte/s Real Time Query: OFF ```	DBA

Task	Description	Skills required
| | ```
Instance(s):
ORCL
Database Status:
SUCCESS
DGMGRL> edit database
 orcl_d set state=app
ly-off;
Succeeded.
DGMGRL> show database
 orcl_d
Database - orcl_d
Role: PHYSICAL STANDBY
Intended State: APPLY-
OFF
Transport Lag: 0
 seconds (computed 1
 second ago)
Apply Lag: 42 seconds
 (computed 1 second ago)
Average Apply Rate:
 (unknown)
Real Time Query: OFF
Instance(s):
ORCL
Database Status:
SUCCESS
DGMGRL>
``` <br> **CDB** <br> ```
DGMGRL> show configura
tionDGMGRL> show
 configuration
Configuration - rds_dg
  Protection Mode:
 MaxAvailability
  Members:
  rdscdb_a - Primary
 database
    rdscdb_b - Physical
 standby database
``` | |

Task	Description	Skills required

	<pre>Fast-Start Failover:
	 Disabled
	Configuration Status:
	SUCCESS (status
	 updated 57 seconds ago)
	DGMGRL> show database
	 rdscdb_b;
	Database - rdscdb_b
	  Role:
	 PHYSICAL STANDBY
	  Intended State:
	 APPLY-ON
	  Transport Lag:
	 0 seconds (computed 1
	 second ago)
	  Apply Lag:
	 0 seconds (computed 1
	 second ago)
	  Average Apply Rate:
	 2.00 KByte/s
	  Real Time Query:
	 OFF
	  Instance(s):
	    RDSCDB
	Database Status:
	SUCCESS
	DGMGRL> edit database
	 rdscdb_b set state=app
	ly-off;
	Succeeded.
	DGMGRL> show database
	 rdscdb_b;
	Database - rdscdb_b
	  Role:
	 PHYSICAL STANDBY
	  Intended State:
	 APPLY-OFF
	  Transport Lag:
	 0 seconds (computed 1
	 second ago)</pre>	

Task	Description	Skills required

	``
	  Apply Lag:
	 0 seconds (computed 1
	 second ago)
	  Average Apply Rate:
	 (unknown)
	  Real Time Query:
	 OFF
	  Instance(s):
	    RDSCDB
	Database Status:
	SUCCESS
	``	

Task	Description	Skills required

Open the read replica instance in read-only mode.	Connect to the standby database by using the TNS entry, and open it in read-only mode by connecting to it from the primary or standby node.
	
	Non-CDB
	
	`$ sqlplus RDS_DATAGUARD@RDS_CUSTOM_ORCL_D as sysdg`
	`-bash-4.2$ sqlplus RDS_DATAGUARD@RDS_CUSTOM_ORCL_D as sysdg`
	`SQL*Plus: Release 19.0.0.0.0 - Production on Fri Sep 30 23:00:14 2022`
	`Version 19.10.0.0.0`
	`Copyright (c) 1982, 2020, Oracle. All rights reserved.`
	`Enter password:`
	`Last Successful login time: Fri Sep 30 2022 22:48:27 +00:00`
	`Connected to:`
	`Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production`
	`Version 19.10.0.0.0`
	`SQL> select open_mode from v$database;`
	`OPEN_MODE`
	`--------------------`
	`MOUNTED`
	`SQL> alter database open read only;`
		DBA

Task	Description	Skills required
	``` Database altered. SQL> select open_mode from v$database; OPEN_MODE -------------------- READ ONLY SQL> ``` **CDB** ``` -bash-4.2$ sqlplus C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_B as sysdg SQL*Plus: Release 19.0.0.0.0 - Productio n on Wed Jan 11 21:14:07 2023 Version 19.16.0.0.0 Copyright (c) 1982, 2022, Oracle. All rights reserved. Enter password: Last Successful login time: Wed Jan 11 2023 21:12:05 +00:00 Connected to: Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production Version 19.16.0.0.0 SQL> select name,open _mode from v$database; NAME OPEN_MODE --------- --------- ----------- RDSCDB MOUNTED SQL> alter database open read only; Database altered. ```	

Task	Description	Skills required
	``` SQL> select name,open _mode from v$database; NAME OPEN_MODE --------- --------- ----------- RDSCDB READ ONLY SQL> ```	

Task	Description	Skills required

Activate redo apply on the read replica instance.	Activate redo apply on the read replica instance by using DGMGRL from the primary or standby node.
	
	Non-CDB
	
	`$ dgmgrl RDS_DATAGUARD@RDS_CUSTOM_ORCL_D`
	`DGMGRL for Linux: Release 19.0.0.0.0 - Production on Fri Sep 30 23:02:16 2022`
	`Version 19.10.0.0.0`
	`Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.`
	`Welcome to DGMGRL, type "help" for information.`
	`Password:`
	`Connected to "ORCL_D"`
	`Connected as SYSDG.`
	`DGMGRL>`
	`edit database orcl_d set state=apply-on;`
	`DGMGRL> edit database orcl_d set state=apply-on;`
	`Succeeded.`
	`DGMGRL> show database orcl_d`
	`Database - orcl_d`
	`Role: PHYSICAL STANDBY`
	`Intended State: APPLY-ON`
		DBA

Task	Description	Skills required

	<pre>Transport Lag: 0
	 seconds (computed 0
	 seconds ago)
	Apply Lag: 0 seconds
	 (computed 0 seconds
	 ago)
	Average Apply Rate:
	 496.00 KByte/s
	Real Time Query: ON
	Instance(s):
	ORCL
	Database Status:
	SUCCESS
	DGMGRL></pre> **CDB** <pre>-bash-4.2$ dgmgrl
	 C##RDS_DATAGUARD@R
	DS_CUSTOM_RDSCDB_B
	-bash-4.2$ dgmgrl
	 C##RDS_DATAGUARD@R
	DS_CUSTOM_RDSCDB_B
	DGMGRL for Linux:
	 Release 19.0.0.0.0 -
	 Production on Wed Jan
	 11 21:21:11 2023
	Version 19.16.0.0.0
	Copyright (c) 1982,
	 2019, Oracle and/or
	 its affiliates. All
	 rights reserved.
	Welcome to DGMGRL, type
	 "help" for informati
	on.
	Password:
	Connected to "RDSCDB_B
	"
	Connected as SYSDG.</pre>	

Task	Description	Skills required

	<pre>DGMGRL> edit database
	 rdscdb_b set state=app
	ly-on;
	Succeeded.
	DGMGRL> show database
	 rdscdb_b
	Database - rdscdb_b
	  Role:
	 PHYSICAL STANDBY
	  Intended State:
	 APPLY-ON
	  Transport Lag:
	 0 seconds (computed 0
	 seconds ago)
	  Apply Lag:
	 0 seconds (computed 0
	 seconds ago)
	  Average Apply Rate:
	 35.00 KByte/s
	  Real Time Query: ON
	  Instance(s):
	    RDSCDB
	Database Status:
	SUCCESS
	DGMGRL> show database
	 rdscdb_b
	Database - rdscdb_b
	  Role:
	 PHYSICAL STANDBY
	  Intended State:
	 APPLY-ON
	  Transport Lag:
	 0 seconds (computed 1
	 second ago)
	  Apply Lag:
	 0 seconds (computed 1
	 second ago)
	  Average Apply Rate:
	 16.00 KByte/s
	  Real Time Query: ON
	  Instance(s):
	    RDSCDB</pre>	

Task	Description	Skills required
	Database Status: SUCCESS DGMGRL>	

Related resources

- Configuring Amazon RDS as an Oracle PeopleSoft Database (AWS whitepaper)
- Oracle Data Guard Broker guide (Oracle reference documentation)
- Data Guard Concepts and Administration (Oracle reference documentation)

Additional information

Select your database protection mode

Oracle Data Guard provides three protection modes to configure your Data Guard environment based on your availability, protection, and performance requirements. The following table summarizes these three modes.

Protection mode	Redo transport setting	Description
MAXIMUM PERFORMANCE	`ASYNC`	For transactions happening on the primary database, redo data is asynchronously transmitted and written to the standby database redo log. Therefore, the performance impact is minimal. `MaxPerformance` can't provide RPO=0 because of asynchronous log shipping.
MAXIMUM PROTECTION	`SYNC+AFFIRM`	For transactions on the primary database, redo data is synchronously transmitted

		and written to the standby database redo log on disk before the transaction is acknowledged. If the standby database becomes unavailab le, the primary database shuts itself down to ensure transactions are protected.
MAXIMUM AVAILABILITY	`SYNC+AFFIRM`	This is similar to `MaxProtec tion` mode, except when no acknowledgement is received from the standby database. In that case, it operates as if it were in `MaxPerformance` mode to preserve the primary database availability until it's able to write its redo stream to a synchronized standby database again.

	SYNC+NOAFFIRM	For transactions on the primary database, redo is synchronously transmitted to the standby database, and the primary waits only for acknowledgement that the redo has been received on the standby, not that it has been written to the standby disk. This mode, which is also known as `FastSync`, can provide a performance benefit at the expense of potential exposure to data loss in a special case of multiple simultaneous failures.

Read replicas in RDS Custom for Oracle are created with maximum performance protection mode, which is also the default protection mode for Oracle Data Guard. The maximum performance mode provides the lowest performance impact on the primary database, which can help you meet the recovery point objective (RPO) requirement measured in seconds.

To work to achieve a zero data loss (RPO=0) objective, you can customize the Oracle Data Guard protection mode to `MaxAvailability` with the `SYNC+NOAFFIRM` setting for redo transport for better performance. Because commits on the primary database are acknowledged only after the corresponding redo vectors are successfully transmitted to the standby database, the network latency between the primary instance and replica can be crucial for commit-sensitive workloads. We recommend performing load testing for your workload to assess the performance impact when the read replica is customized to run in `MaxAvailability` mode.

Deploying the read replica in the same Availability Zone as the primary database provides lower network latency compared with deploying the read replica in a different Availability Zone. However, deploying the primary and read replicas in the same Availability Zone might not meet your HA requirements because, in the unlikely event of Availability Zone unavailability, both the primary instance and read replica instance are impacted.

Assess query performance for migrating SQL Server databases to MongoDB Atlas on AWS

Battulga Purevragchaa, Amazon Web Services

Krishnakumar Sathyanarayana, PeerIslands US Inc

Babu Srinivasan, MongoDB

Summary

This pattern provides guidance for loading MongoDB with near real-world data and assessing MongoDB query performance that is as close to the production scenario as possible. The assessment provides input to help you plan your migration to MongoDB from a relational database. The pattern uses PeerIslands Test Data Generator and Performance Analyzer to test query performance.

This pattern is particularly useful for Microsoft SQL Server migration to MongoDB, because performing schema transformations and loading data from current SQL Server instances to MongoDB can be very complex. Instead, you can load near real-world data into MongoDB, understand MongoDB performance, and fine-tune the schema design before you start the actual migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Familiarity with MongoDB Atlas
- Target MongoDB schema
- Typical query patterns

Limitations

- Data load times and performance will be limited by the MongoDB cluster instance size. We recommend that you choose instances that are recommended for production use to understand real-world performance.

- PeerIslands Test Data Generator and Performance Analyzer currently supports only online data loads and queries. Offline batch processing (for example, loading data into MongoDB by using Spark connectors) isn't yet supported.
- PeerIslands Test Data Generator and Performance Analyzer supports field relations within a collection. It doesn't support relationships across collections.

Product editions

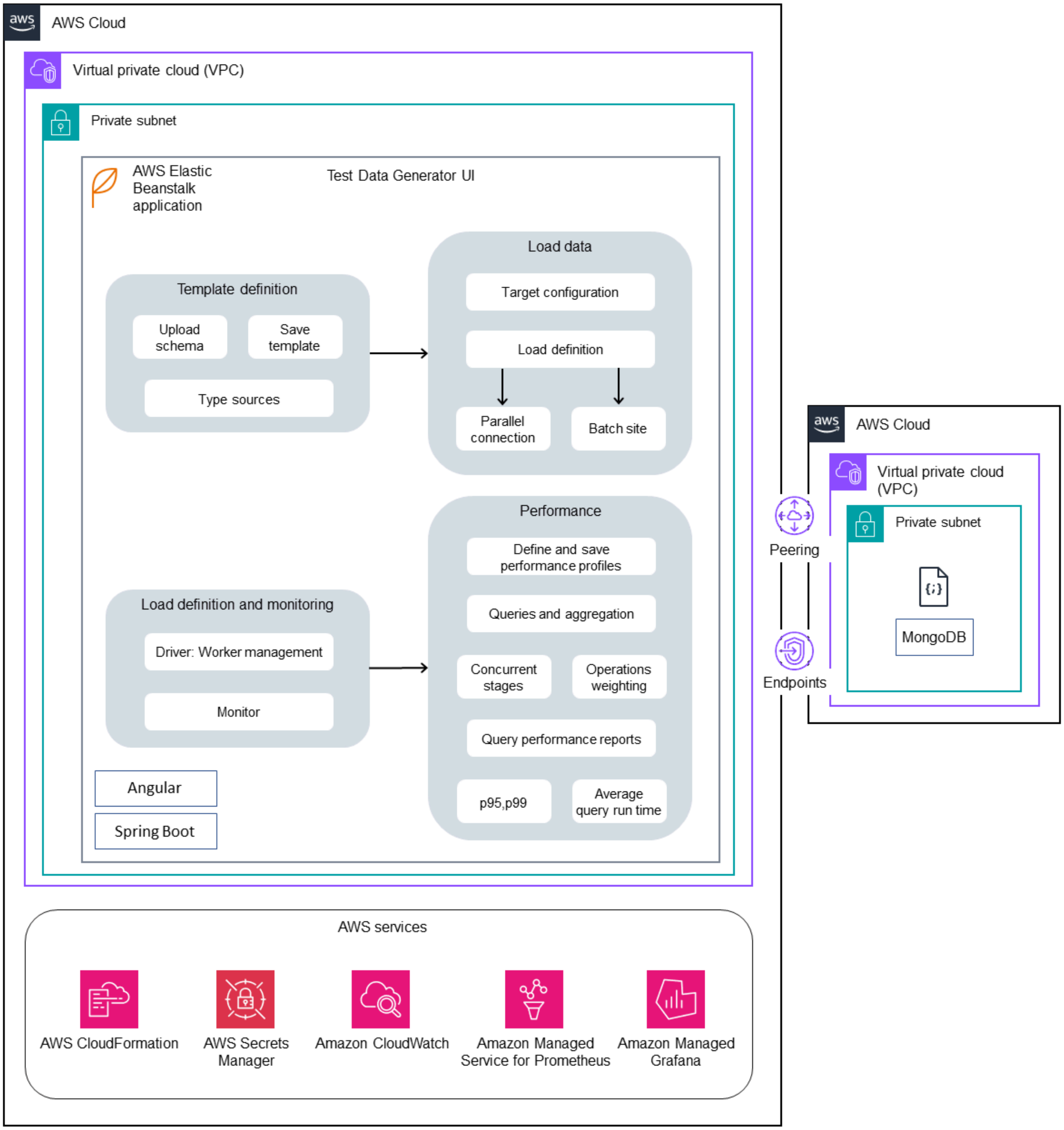
- This pattern supports both MongoDB Atlas and MongoDB Enterprise Advanced.

Architecture

Target technology stack

- MongoDB Atlas or MongoDB Enterprise Advanced

Architecture



PeerIslands Test Data Generator and Performance Analyzer is built by using Java and Angular, and stores its generated data on Amazon Elastic Block Store (Amazon EBS). The tool consists of two workflows: test data generation and performance testing.

- In test data generation, you create a template, which is the JSON representation of the data model that has to be generated. After you create the template, you can generate the data in a target collection, as defined by the load generation configuration.
- In performance testing, you create a profile. A profile is a multi-stage testing scenario where you can configure create, read, update, and delete (CRUD) operations, aggregation pipelines, the weightage for each operation, and the duration of each stage. After you create the profile, you can run performance testing on the target database based on the configuration.

PeerIslands Test Data Generator and Performance Analyzer stores its data on Amazon EBS, so you can connect Amazon EBS to MongoDB by using any MongoDB-supported connection mechanism, including peering, allow lists, and private endpoints. By default, the tool doesn't include operational components; however, it can be configured with Amazon Managed Service for Prometheus, Amazon Managed Grafana, Amazon CloudWatch, and AWS Secrets Manager if required.

Tools

- PeerIslands Test Data Generator and Performance Analyzer includes two components. The Test Data Generator component helps you generate highly customer-specific, real-world data based on your MongoDB schema. The tool is fully UI-driven with a rich data library and can be used to quickly generate billions of records on MongoDB. The tool also provides capabilities to implement relationships between fields in the MongoDB schema. The Performance Analyzer component helps you generate highly customer-specific queries and aggregations, and perform realistic performance testing on MongoDB. You can use the Performance Analyzer to test MongoDB performance with rich load profiles and parameterized queries for your specific use case.

Best practices

See the following resources:

- MongoDB Schema Design Best Practices (MongoDB Developer website)
- Best Practices of Deploying MongoDB Atlas on AWS (MongoDB website)
- Connecting Applications Securely to a MongoDB Atlas Data Plane with AWS PrivateLink (AWS blog post)
- Best Practices Guide for MongoDB Performance (MongoDB website)

Epics

Understand your source data

Task	Description	Skills required
Understand the database footprint of the current SQL Server source.	Understand your current SQL Server footprint. This can be achieved by running queries against the `INFORMATION` schema of the database. Determine the number of tables and size of each table. Analyze the index associate d with each table. For more information about SQL analysis, see the blog post [SQL2Mongo: Data Migration Journey](#) on the PeerIslands website.	DBA
Understand the source schema.	Determine the table schema and the business represent ation of the data (for example, zip codes, names, and currency). Use your existing entity relationship (ER) diagram or generate the ER diagram from the existing database. For more information, see the blog post [SQL2Mongo: Data Migration Journey](#) on the PeerIslands website.	DBA
Understand query patterns.	Document the top 10 SQL queries you use. You can use the **performance_schema**	DBA

Task	Description	Skills required
	.events_statements _summary_by_digest tables that are available in the database to understand the top queries. For more information, see the blog post [SQL2Mongo: Data Migration Journey]() on the PeerIslands website.	
Understand SLA commitmen ts.	Document the target service-level agreements (SLAs) for database operations. Typical measures include query latency and queries per second. The measures and their targets are typically available in non-funct ional requirements (NFR) documents.	DBA

Define the MongoDB schema

Task	Description	Skills required
Define the target schema.	Define various options for the target MongoDB schema. For more information, see [Schemas]() in the MongoDB Atlas documentation. Consider the best practices and design patterns based on the table relations. See [Data Model Examples and Patterns]()	MongoDB engineer

Task	Description	Skills required
	in the MongoDB documentation for details.	
Define target query patterns.	Define MongoDB queries and aggregation pipelines. These queries are the equivalent of the top queries you captured for your SQL Server workload. To understand how to construct MongoDB aggregation pipelines, see the MongoDB documentation.	MongoDB engineer
Define the MongoDB instance type.	Determine the size of the instance that you plan to use for testing. For guidance, see the MongoDB documentation.	MongoDB engineer

Prepare the target database

Task	Description	Skills required
Set up the MongoDB Atlas cluster.	To set up a MongoDB cluster on AWS, follow the instructions in the MongoDB documentation.	MongoDB engineer
Create users in the target database.	Configure the MongoDB Atlas cluster for access and network security by following the instructions in the MongoDB documentation.	MongoDB engineer

Task	Description	Skills required
Create appropriate roles in AWS and configure role-based access control for Atlas.	If required, set up additiona l users by following the instructions in the MongoDB documentation. Configure authentication and authoriza tion through AWS roles.	MongoDB engineer
Set up Compass for MongoDB Atlas access.	Set up the MongoDB Compass GUI utility for ease of navigation and access.	MongoDB engineer

Set up the base load by using Test Data Generator

Task	Description	Skills required
Install Test Data Generator.	Install PeerIsland Test Data Generator in your environme nt.	MongoDB engineer
Configure Test Data Generator to generate the appropriate data.	Create a template by using the data library to generate specific data for each field in the MongoDB schema. For more information, see the MongoDB Data Generator & Perf. Analyzer video.	MongoDB engineer
Horizontally scale Test Data Generator to generate the required load.	Use the template you created to start the load generation against the target collection by configuring the required parallelism. Determine the time frames and scale to generate the necessary data.	MongoDB engineer

Task	Description	Skills required
Validate the load in MongoDB Atlas.	Check the data loaded into MongoDB Atlas.	MongoDB engineer
Generate required indexes on MongoDB.	Define indexes as required, based on query patterns. For best practices, see the [MongoDB documentation]().	MongoDB engineer

Conduct performance testing

Task	Description	Skills required
Set up load profiles in Performance Analyzer.	Create a performance testing profile in Performance Analyzer by configuring specific queries and their corresponding weightage, duration of the test run, and stages. For more informati on, see the [MongoDB Data Generator & Perf. Analyzer]() video.	MongoDB engineer
Run performance testing.	Use the performance testing profile you created to start the test against the target collection by configuri ng the required paralleli sm. Horizontally scale the performance test tool to run queries against MongoDB Atlas.	MongoDB engineer

Task	Description	Skills required
Record test results.	Record P95, P99 latency for the queries.	MongoDB engineer
Tune your schema and query patterns.	Modify indexes and query patterns to address any performance issues.	MongoDB engineer

Close the project

Task	Description	Skills required
Shut down temporary AWS resources.	Delete all temporary resources that you used for Test Data Generator and Performance Analyzer.	AWS administrator
Update performance test results.	Understand MongoDB query performance and compare it against your SLAs. If necessary, fine-tune the MongoDB schema and rerun the process.	MongoDB engineer
Conclude the project.	Close out the project and provide feedback.	MongoDB engineer

Related resources

- GitHub repository: S3toAtlas
- Schema: MongoDB Schema design
- Aggregation pipelines : MongoDB aggregation pipelines
- MongoDB Atlas sizing : Sizing tier selection
- Video: MongoDB Data Generator & Perf. Analyzer

- References: MongoDB documentation
- Tutorials: MongoDB developer guide, MongoDB Jumpstart
- AWS Marketplace: MongoDB Atlas on AWS Marketplace
- AWS Partner Solutions: MongoDB Atlas on AWS Reference Deployment

Additional resources:

- SQL analysis
- MongoDB Developer Community forums
- MongoDB Performance Tuning Questions
- Operational Analytics with Atlas and Redshift
- Application modernization with MongoDB Atlas and AWS Elastic Beanstalk

Automate blue/green deployments of Amazon Aurora global databases by using IaC principles

Ishwar Chauthaiwale, ANKIT JAIN, and Ramu Jagini, Amazon Web Services

Summary

Managing database updates, migrations, or scaling efforts can be challenging for organizations that run critical workloads on [Amazon Aurora global databases](). Ensuring that these operations are carried out seamlessly, with zero downtime, is essential to maintaining service availability and avoiding disruptions for your users.

A blue/green deployment strategy offers a solution to this challenge by allowing you to run two identical environments concurrently: blue (the current environment) and green (the new environment). A blue/green strategy enables you to implement changes, perform testing, and switch traffic between environments with minimal risk and downtime.

This pattern helps you automate the blue/green deployment process for Aurora global databases by using infrastructure as code (IaC) principles. It uses [AWS CloudFormation](), [AWS Lambda](), and [Amazon Route 53]() to simplify blue/green deployments. To improve reliability, it uses global transaction identifiers (GTIDs) for replication. GTID-based replication provides better data consistency and failover capabilities between environments compared with binary log (binlog) replication.

> **Note**
>
> This pattern assumes that you're using an Aurora MySQL-Compatible Edition global database cluster. If you're using Aurora PostgreSQL-Compatible instead, please use the PostgreSQL equivalents of the MySQL commands.

By following the steps in this pattern, you can:

- Provision a green Aurora global database: Using CloudFormation templates, you create a green environment that mirrors your existing blue environment.
- Set up GTID-based replication: You configure GTID replication to keep the blue and green environments synchronized.
- Seamlessly switch traffic: You use Route 53 and Lambda to automatically switch the traffic from the blue to the green environment after full synchronization.

- Finalize the deployment: You validate that the green environment is fully operational as the primary database, and then stop replication and clean up any temporary resources.

The approach in this pattern provides these benefits:

- Reduces downtime during critical database updates or migrations: Automation ensures a smooth transition between environments with minimal service disruption.
- Enables rapid rollbacks: If an issue arises after traffic is switched to the green environment, you can quickly revert to the blue environment and maintain service continuity.
- Enhances testing and verification: The green environment can be fully tested without affecting the live blue environment, which reduces the likelihood of errors in production.
- Ensures data consistency: GTID-based replication keeps your blue and green environments in sync, which prevents data loss or inconsistencies during migration.
- Maintains business continuity: Automating your blue/green deployments helps avoid long outages and financial losses by keeping your services available during updates or migrations.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A source Aurora MySQL-Compatible global database cluster (blue environment). Global databases provide a multi-Region configuration for high availability and disaster recovery. For instructions for setting up a global database cluster, see the Aurora documentation.
- GTID-based replication enabled on the source cluster.

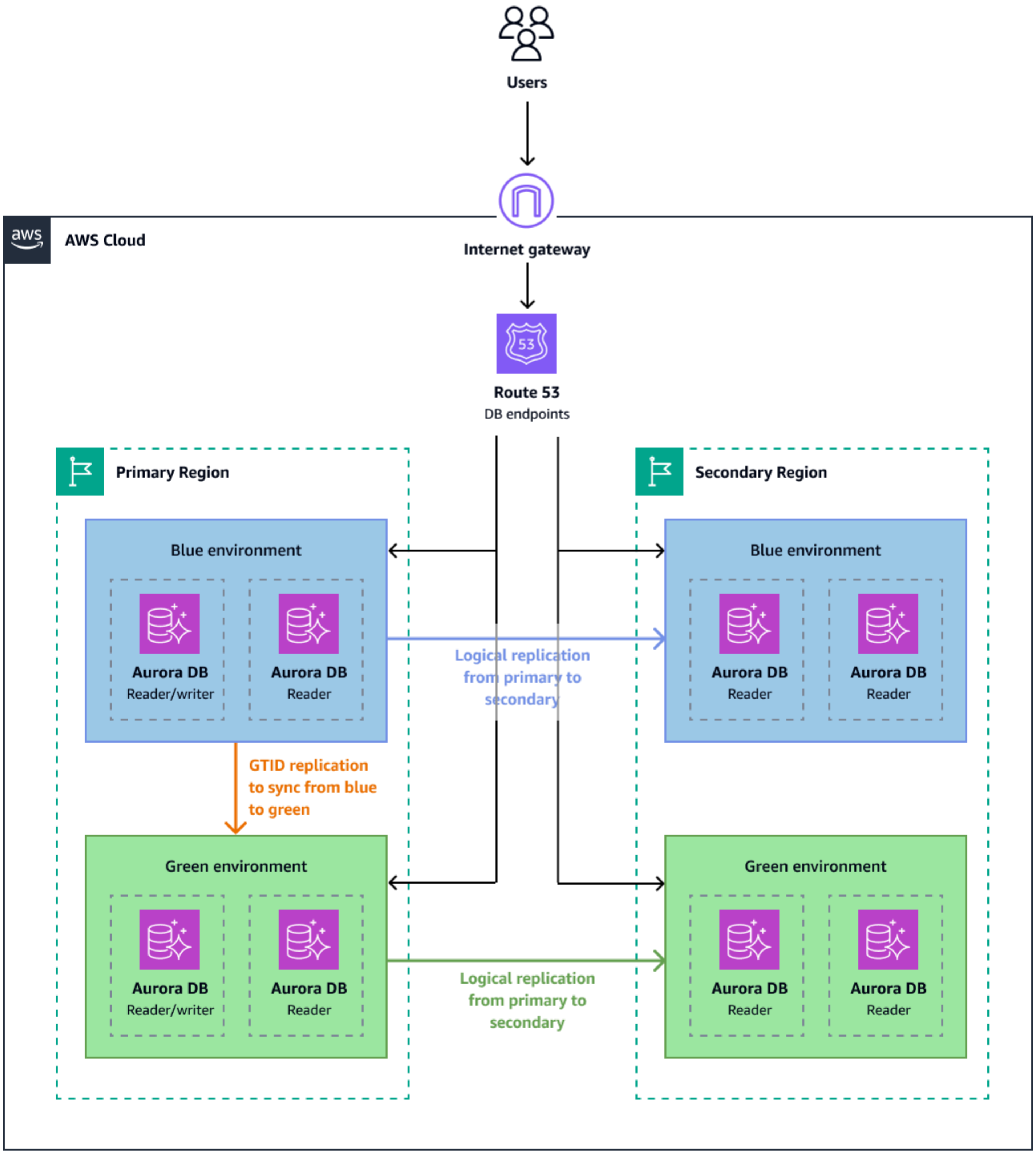
Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Product versions

- Aurora MySQL-Compatible 8.0 or later

Architecture



The diagram illustrates the following:

- Global database setup: An Aurora global database cluster is strategically deployed across two AWS Regions. This configuration enables geographic distribution and Regional redundancy for enhanced disaster recovery capabilities.
- Primary to secondary Region replication: The logical replication mechanism ensures seamless data synchronization from the primary Region to the secondary Region. This replication maintains data consistency with minimal latency across geographical distances.
- GTID-based replication between clusters: GTID-based replication maintains transactional consistency and ordered data flow between the blue primary cluster and the green primary cluster, and ensures reliable data synchronization.
- Blue primary to secondary replication: Logical replication establishes a robust data pipeline between the blue primary cluster and its secondary cluster. This replication enables continuous data synchronization and high availability.
- Route 53 DNS configuration: Route 53 hosted zone records manage the DNS resolution for all blue and green cluster database endpoints. This configuration provides seamless endpoint mapping and enables efficient traffic routing during failover scenarios.

Tools

AWS services

- Amazon Aurora is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL.
- AWS CloudFormation helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run on AWS. You create a template that describes all the AWS resources that you want, and CloudFormation takes care of provisioning and configuring those resources for you.
- AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon Route 53 is a highly available and scalable DNS web service.

Best practices

We recommend that you thoroughly review AWS documentation to deepen your understanding of the blue/green deployment strategy, GTID-based replication, and weighted routing policies in Route 53. This knowledge is crucial for effectively implementing and managing your database migrations, ensuring data consistency, and optimizing traffic routing. By gaining a comprehensive understanding of these AWS features and best practices, you'll be better equipped to handle future updates, minimize downtime, and maintain a resilient and secure database environment.

For guidelines for using the AWS services for this pattern, see the following AWS documentation:

- Best practices with Amazon Aurora MySQL
- AWS CloudFormation best practices
- Best practices for working with AWS Lambda functions
- Best practices for Amazon Route 53

Epics

Create the green environment

Task	Description	Skills required
Create a snapshot backup from the blue cluster.	In a blue/green deploymen t, the green environment represents a new, identical version of your current (blue) database environment. You use the green environme nt to safely test updates, validate changes, and ensure stability before switching production traffic. It acts as a staging ground for implement ing database changes with minimal disruption to the live environment. To create a green environme nt, you first create a snapshot of the primary (blue) cluster in your Aurora MySQL-Com	DBA

Task	Description	Skills required
	patible global database. This snapshot serves as the foundation for creating the green environment. To create a snapshot: 1. Sign in to the AWS Management Console and open the [Amazon Relational Database Service (Amazon RDS) console](https://console.aws.amazon.com/rds/). 2. Select your primary (blue) cluster. 3. Choose **Actions**, **Take snapshot**. 4. Provide a name for the snapshot, such as `blue-green-demo` , and start the backup process. Alternatively, you can use the AWS Command Line Interface (AWS CLI) to create the snapshot: `aws rds create-db-cluster-snapshot --db-cluster-snapshot-identifier blue-green-demo --db-cluster-identifier ex-global-cluster --region eu-west-1`	

Task	Description	Skills required
	Make sure that the snapshot completes successfully before proceeding to the next step.	
Generate the CloudFormation template for your global database and its resources.	The CloudFormation IaC generator helps you generate CloudFormation templates from existing AWS resources. Use this feature to create a CloudFormation template for your existing Aurora MySQL-Compatible global database and its associated resources. This template configures subnet groups, security groups, parameter groups, and other settings. 1. Follow the [instructions in the CloudFormation documentation] to navigate to the tool and connect it to your AWS environment. 2. Select your Aurora global database and associated resources to generate the template.	DBA

Task	Description	Skills required
Modify the CloudFormation template for the green environment.	Customize the CloudFormation template to reflect the settings for the green environment. This includes updating resource names and identifiers to ensure that the green environment operates independently of the blue cluster. 1. Update the `DBClusterIdentifier` and `DBInstanceIdentifier` properties to represent the green environment. 2. Modify other resource names (for example, subnet groups and security groups) to avoid conflicts with the existing blue environment. 3. Enable GTID-based replication in the template by configuring the correct parameters, as described in the [Aurora documentation](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/). 4. Change the `SnapshotIdentifier` property to specify the AWS Region, your account ID, and the name of the snapshot from the previous step:	DBA

Task	Description	Skills required
	`SnapshotIdentifier: arn:aws:rds:<region>:<account-id>:snapshot:<snapshot-name>` **Note** If you use the `SnapshotIdentifier` property to restore a DB cluster, avoid specifying properties such as `GlobalClusterIdentifier` , `MasterUsername` , or `MasterUserPassword` .	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy the CloudFormation stack to create resources for the green environment.</td><td>In this step, you deploy the customized CloudForm ation template to create the resources for the green environment.

To deploy the CloudForm ation stack:

1. Open the AWS CloudForm ation console.
2. In the upper right, choose Create stack, With new resources (standard).
3. Upload your modified CloudFormation template or specify the template URL. Choose Next.
4. Enter a stack name such as <code>GreenClusterStack</code> , and provide any necessary parameters (for example, <code>GreenClusterIdenti fier</code>). Choose Next.
5. Configure additional stack options as needed, and check the box to acknowledge that CloudFormation might create AWS Identity and Access Management (IAM) resources. Choose Next.
6. Review the stack details.
7. Choose Submit.</td><td>DBA</td></tr>
</table>

Task	Description	Skills required
	CloudFormation initiates the process of creating the green environment resources. This process might take several minutes to complete.	

Task	Description	Skills required
Validate the CloudFormation stack and resources.	When the CloudFormation stack deployment is complete, you'll need to verify that the green environment has been created successfully: 1. In the **Outputs** section of the CloudFormation stack, check the endpoints of the database cluster and database instance to verify correct setup. 2. Open the [Amazon RDS console](#) and confirm that the new Aurora database cluster (green environment) is available. 3. Make sure that all associated resources, such as subnets and security groups, have been created and linked to the green environment. After verification, your green environment is ready for further setup, including replication from the blue environment.	DBA

Configure GTID-based replication

Task	Description	Skills required
Verify GTID settings on the blue cluster.	GTIDs provide a highly reliable method for replicating data between your blue and green environments. [GTID-based replication](#) offers a resilient, simplified approach by assigning a unique identifier to every transaction in the blue environment. This method ensures that data synchronization between environments is seamless, consistent, and easier to manage than traditional binlog replication. Before you configure replication, you need to ensure that GTID-based replication is properly enabled on the blue cluster. This step guarantees that each transaction in the blue environment is uniquely tracked and can be replicated in the green environment. To confirm that GTID is enabled: 1. On the [Amazon RDS console](#), review the parameter group assigned to the blue cluster.	DBA

Task	Description	Skills required
	2. Verify that the following parameters are set: • `gtid-mode = ON` • `enforce_gtid_consistency = ON` These settings enable GTID tracking for all future transactions in the blue environment. After you confirm these settings, you can start setting up replication.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a replication user.</td><td>

To replicate data from the blue environment to the green environment, you need to create a dedicated replication user on the blue cluster. This user will be responsible for managing the replication process.

To set up the replication user:

1. Connect to the blue cluster by using a MySQL client.
2. Run the following commands to create the replication user:

```
CREATE USER 'repl_user'@'%' IDENTIFIED BY 'repl_password';
GRANT REPLICATION SLAVE ON . TO 'repl_user'@'%';
FLUSH PRIVILEGES;
```

This user now has the necessary permissions to replicate data between the two environments.

</td><td>DBA</td></tr>
</table>

Task	Description	Skills required
Configure GTID-based replication on the green cluster.	The next step is to configure the green cluster for GTID-based replication. This setup ensures that the green environment will continuously mirror all transactions that happen in the blue environment. To configure the green cluster: 1. Connect to the green cluster by using a MySQL client. 2. Run the following command to configure replication: `CHANGE MASTER TO MASTER_HOST='blue-cluster-endpoint', MASTER_USER='repl_user', MASTER_PASSWORD='repl_password', MASTER_AUTO_POSITION=1;` where: • Replace `blue-cluster-endpoint` with the endpoint of your blue cluster. • The `MASTER_AUTO_POSITION=1` setting instructs MySQL	DBA

Task	Description	Skills required
	to use GTID-based replication. It automatically positions the green cluster to replicate the blue cluster's transactions without having to track logs and positions manually.	
Start replication on the green cluster.	You can now start the replication process. On the green cluster, run the command: `START SLAVE;` This enables the green environment to start synchronizing data, and receiving and applying transactions from the blue environment.	DBA

Task	Description	Skills required
Verify the replication process.	To verify that the green environment is accurately replicating the data from the blue cluster: 1. Run the following command on the green cluster to check the replication status: `SHOW SLAVE STATUS\G;` 2. Review the output to verify the following: • `Slave_IO_Running = Yes` • `Slave_SQL_Running = Yes` • The `Retrieved_Gtid_Set` and `Executed_Gtid_Set` values are up-to-date and synchronized with the blue cluster. • There are no replication errors in the `Last_Error` field. If all indicators are correct, GTID-based replication is functioning smoothly, and the green environment is fully synchronized with the blue environment.	DBA

Switch traffic from blue to green cluster

Task	Description	Skills required
Configure Route 53 weighted routing policies.	After you verify data consistency between the blue and green environme nts, you can switch traffic from the blue cluster to the green cluster. This transitio n should be smooth and should minimize downtime and ensure the integrity of your application's database. To address these requireme nts, you can use Route 53 for DNS routing and Lambda to automate traffic switching. Additionally, a well-defined rollback plan ensures that you can revert to the blue cluster in case of any issues. The first step is to configure weighted routing in Route 53. Weighted routing allows you to control the distribut ion of traffic between the blue and green clusters, and gradually shift traffic from one environment to the other. To configure weighted routing: 1. Open the [Route 53 console](https://console.aws.amazon.com/route53/) and choose your hosted zone.	AWS DevOps

Task	Description	Skills required
	2. Create two DNS records (CNAMEs) for the database: one record for the blue cluster and one record for the green cluster. 3. Assign initial weights: • Set a low initial weight (such as 5 percent) for the green cluster to send a small portion of traffic for testing. • Set a higher weight (such as 95 percent) for the blue cluster, so it retains the majority of traffic. This configuration allows you to perform a gradual transition that reduces risk and supports real-time testing before you switch over fully. For more information about weighted routing policies, see the [Route 53 documentation]().	

Task	Description	Skills required

Deploy a Lambda function to monitor replication lag.	To ensure that the green environment is fully synchronized with the blue environment, deploy a Lambda function that monitors replication lag between the clusters. This function can check the replication status, specifically the **Seconds_Behind_Master** metric, to determine whether the green cluster is ready to handle all traffic.
	
	Here's a sample Lambda function you can use:
	
	`import boto3`
	
	`def check_replication_lag(event, context):`
	`    client = boto3.client('rds')`
	`    response = client.describe_db_instances(DBInstanceIdentifier='green-cluster-instance')`
	`    replication_status = response['DBInstances'][0]['ReadReplicaDBInstanceIdentifiers']`
	`    if replication_status:`
	`        lag = replication_status[0]['ReplicationLag']`
	`        return lag`
		AWS DevOps

Task	Description	Skills required
	```return -1``` This function checks the replication lag and returns the value. If the lag is zero, the green cluster is fully in sync with the blue cluster.	

Task	Description	Skills required

Automate DNS weight adjustment by using Lambda.	When the replication lag reaches zero, it's time to switch all traffic to the green cluster. You can automate this transition by using another Lambda function that adjusts the DNS weights in Route 53 to direct 100 percent of traffic to the green cluster.
	
	Here's an example of a Lambda function that automates the traffic switch:
	
	`import boto3`
	
	`def switch_traffic(eve nt, context):`
	`    route53 = boto3.cli ent('route53')`
	`    lag = check_rep lication_lag(event, context)`
	`    if lag == 0:`
	`        response = route53.change_res ource_record_sets(`
	`            HostedZon eId='YOUR_HOSTED_Z ONE_ID',`
	`            ChangeBat ch={`
	`                'Changes': [`
	`                    {`
	`                        'Action': 'UPSERT',`
		AWS DevOps

Task	Description	Skills required

	'ResourceRecordSet': {
	'Name': 'db.examp le.com',
	'Type': 'CNAME',
	'SetIdentifier': 'GreenCluster',
	'Weight': 100,
	'TTL': 60,
	'ResourceRecords': [{'Value': 'green-cl uster-endpoint'}]
	}
	},
	{
	'Action': 'UPSERT',
	'ResourceRecordSet': {
	'Name': 'db.examp le.com',
	'Type': 'CNAME',
	'SetIdentifier': 'BlueCluster',
	'Weight': 0,
	'TTL': 60,
	'ResourceRecords':	

Task	Description	Skills required

	``
	 [{'Value': 'blue-clu
	ster-endpoint'}]
	
	 }
	                    }
	                ]
	            }
	        )
	        return response
	``
	
	This function checks replication lag and updates the Route 53 DNS weights when the lag is zero to fully switch traffic to the green cluster.
	
	Note
	
	During the cutover process, If the blue cluster experiences heavy write traffic, consider temporarily pausing write operations during the cutover. This ensures that replication catches up, and prevents data inconsistencies between the blue and green clusters.	

Task	Description	Skills required
Verify the traffic switch.	After the Lambda function adjusts the DNS weights, you should verify that all traffic is directed to the green cluster and that the switch was successful. To verify: 1. Monitor the Route 53 DNS records to confirm that traffic is being directed to the green cluster. For more information, see the [Route 53 documentation](https://docs.aws.amazon.com/Route53/). 2. Check application performance by confirming that users are being served from the green environment. 3. Verify database connections to confirm that the green cluster is handling all database requests. 4. Monitor Amazon CloudWatch metrics for any signs of latency, replication lag, or performance degradation. For more information, see the [Aurora documentation](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/).	AWS DevOps

Task	Description	Skills required
	If everything is performing as expected, the traffic switch is complete.	

Task	Description	Skills required
If you encounter any issues, roll back changes.	Having a rollback plan is critical in case any issues arise after the traffic switch. Here's how to quickly revert to the blue cluster if necessary: 1. Revert DNS weights in Route 53: Use the same Lambda function or manually adjust the Route 53 DNS weights to direct 100 percent of traffic back to the blue cluster. 2. Monitor application performance: Immediately monitor application logs, CloudWatch metrics, and database performance to confirm that the switch back to the blue environme nt has resolved the issues. 3. Identify and resolve Issues: Investigate and address any problems with the green cluster before you attempt another traffic switch. By implementing this rollback plan, you can ensure minimal disruption to your users in the event of any unexpected issues.	AWS DevOps

Validate and stop GTID-based replication

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Stop GTID-based replication on the green cluster.</td><td>After you switch traffic from the blue environment to the green environment, you should validate the success of the transition and ensure that the green cluster is functioning as expected. Additionally, the GTID-based replication between the blue and green clusters must be stopped, because the green environment now serves as the primary database. Completing these tasks ensures that your environment is secure, streamlined, and optimized for ongoing operations.

To stop replication:

1. Use a MySQL client to connect to the green cluster.
2. Run the following SQL command to stop the replication process on the green cluster:

<code>STOP SLAVE;</code>

3. (Optional) If desired, you can reset the replication configuration to clear</td><td>DBA</td></tr>
</table>

Task	Description	Skills required
	any residual replication settings: `RESET SLAVE ALL;` When you stop the replication, the green cluster becomes fully independent and operates as the primary database environment for your workloads.	

Task	Description	Skills required
Clean up resources.	Cleaning up any temporary or unused resources that were created during the migration from the blue to the green cluster ensures that your environment remains optimized, secure, and cost-effective. The cleanup includes adjusting security settings, taking final backups, and decommiss ioning unnecessary resources. To clean up resources: 1. Update security groups: Configure the security groups that are associated with both the blue and the green clusters to reflect the new primary environme nt (green). Restrict access to the blue environment if it is no longer needed, and verify that the green cluster's security settings follow best practices. 2. Make a final backup of the green cluster: After the migration is complete, take a final snapshot of the green cluster to serve as a backup. You can use this snapshot to restore the	AWS DevOps

Task	Description	Skills required

	environment in the future if necessary.
	`aws rds create-db-snapshot --db-instance-identifier green-cluster-instance --db-snapshot-identifier green-cluster-final-snapshot`
	3. Review and remove temporary resources: Review any temporary resources that were created during the migration, such as temporary security groups, snapshots, or other configurations. Delete resources that are no longer needed to prevent unnecessary costs. For example, delete the blue cluster if it is no longer required:
	`aws rds delete-db-cluster --db-cluster-identifier blue-cluster-identifier --skip-final-snapshot`
	
	Cleaning up resources helps maintain a secure and streamlined environment, reduces costs, and ensures	

Task	Description	Skills required
	that only necessary infrastru cture remains.	

Related resources

AWS CloudFormation:

- AWS CloudFormation User Guide
- AWS CloudFormation best practices
- Generate templates from existing resources with IaC Generator
- Import entire applications into AWS CloudFormation (AWS blog post)

Amazon Aurora:

- Amazon Aurora User Guide
- Managing an Amazon Aurora DB cluster

Blue/green deployment strategy:

- Overview of Amazon Aurora Blue/Green Deployments

GTID-based replication:

- Using GTID-based replication (Amazon RDS documentation)

AWS Lambda:

- AWS Lambda Developer Guide
- Best practices for working with AWS Lambda functions

Amazon Route 53:

- Amazon Route 53 Developer Guide
- Weighted routing

MySQL client tools:

- PyMYSQL

Automate the replication of Amazon RDS instances across AWS accounts

Parag Nagwekar and Arun Chandapillai, Amazon Web Services

Summary

This pattern shows you how to automate the process of replicating, tracking, and rolling back your Amazon Relational Database Service (Amazon RDS) DB instances across different AWS accounts by using AWS Step Functions and AWS Lambda. You can use this automation to perform large-scale replication of RDS DB instances without any performance impact or operational overhead—regardless of the size of your organization. You can also use this pattern to help your organization comply with mandatory data governance strategies or compliance requirements that call for your data to be replicated and redundant across different AWS accounts and AWS Regions. Cross-account replication of Amazon RDS data at scale is an inefficient and error-prone manual process that can be costly and time-consuming, but the automation in this pattern can help you achieve cross-account replication safely, effectively, and efficiently.

Prerequisites and limitations

Prerequisites

- Two AWS accounts
- An RDS DB instance, up and running in the source AWS account
- A subnet group for the RDS DB instance in the destination AWS account
- An AWS Key Management Service (AWS KMS) key created in the source AWS account and shared with the destination account (For more information about policy details, see the Additional information section of this pattern.)
- An AWS KMS key in the destination AWS account to encrypt the database in the destination account

Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Product versions

- Python 3.9 (using AWS Lambda)
- PostgreSQL 11.3, 13.x, and 14.x

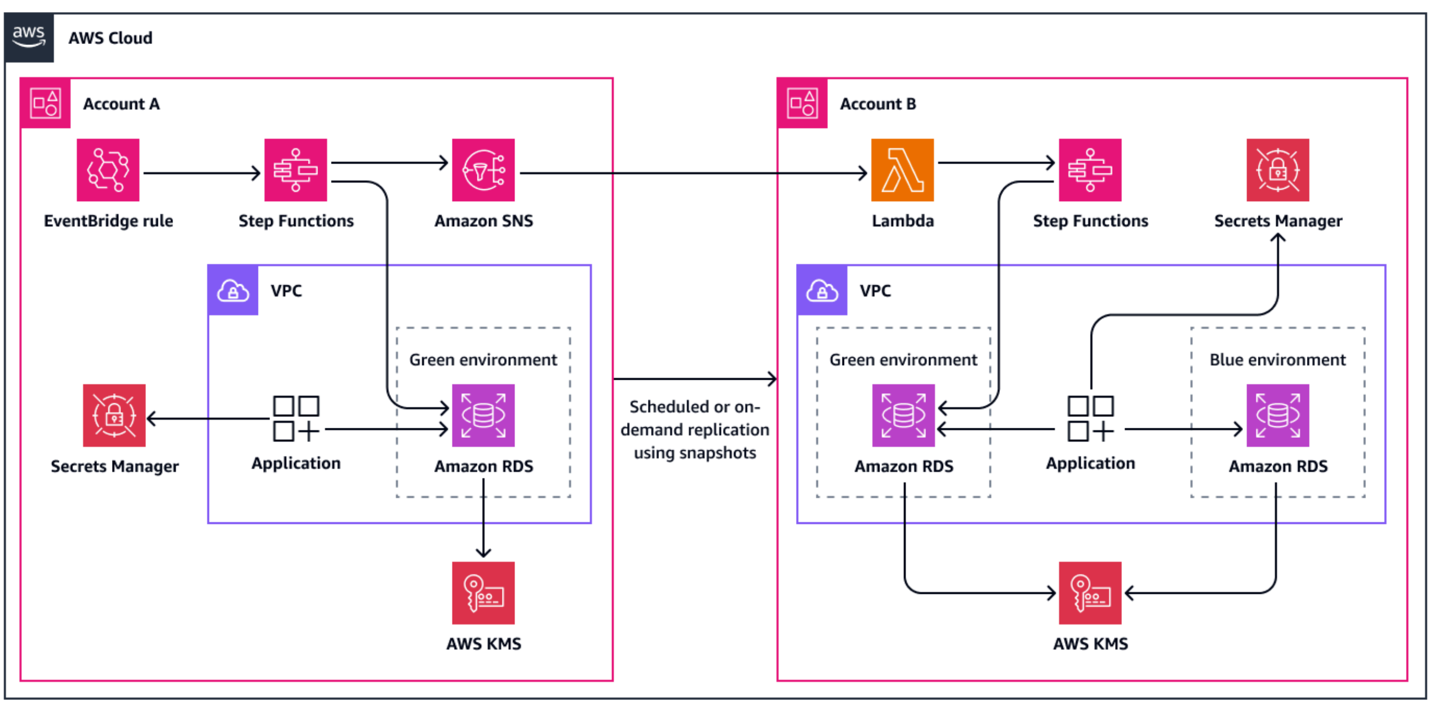
Architecture

Technology stack

- Amazon Relational Database Service (Amazon RDS)
- Amazon Simple Notification Service (Amazon SNS)
- AWS Key Management Service (AWS KMS)
- AWS Lambda
- AWS Secrets Manager
- AWS Step Functions

Target architecture

The following diagram shows an architecture for using Step Functions to orchestrate scheduled, on-demand replication of RDS DB instances from a source account (account A) to a destination account (account B).



In the source account (account A in the diagram), the Step Functions state machine performs the following:

1. Creates a snapshot from the RDS DB instance in account A.
2. Copies and encrypts the snapshot with an AWS KMS key from account A. To ensure encryption in transit, the snapshot is encrypted whether or not the DB instance is encrypted.
3. Shares the DB snapshot with account B by giving account B access to the snapshot.
4. Pushes a notification to the SNS topic, and then the SNS topic invokes the Lambda function in account B.

In the destination account (account B in the diagram), the Lambda function runs the Step Functions state machine to orchestrate the following:

1. Copies the shared snapshot from account A to account B, while using the AWS KMS key from account A to decrypt the data first and then encrypt the data by using the AWS KMS key in account B.
2. Reads the secret from Secrets Manager to capture the name of the current DB instance.
3. Restores the DB instance from the snapshot with a new name and default AWS KMS key for Amazon RDS.
4. Reads the endpoint of the new database and updates the secret in Secrets Manager with the new database endpoint, and then tags the previous DB instance so that it can be deleted later.
5. Keeps the latest N instances of the databases and deletes all the other instances.

Tools

AWS services

- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.

- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- AWS Step Functions is a serverless orchestration service that helps you combine Lambda functions and other AWS services to build business-critical applications.

Code repository

The code for this pattern is available in the GitHub Crossaccount RDS Replication repository.

Epics

Automate the replication of RDS DB instances across AWS accounts with a single click

Task	Description	Skills required
Deploy the CloudFormation stack in the source account.	1. Sign in to the AWS Management Console for the source account (account A) and open the CloudFormation console. 2. In the navigation pane, choose **Stacks**. 3. Choose **Create stack**, and then choose **With existing resources (import resources)**. 4. On the **Identify resources** page, choose **Next**.	Cloud administrator, Cloud architect

Task	Description	Skills required
	5. On the **Specify template** page, select **Upload a template**. 6. Choose **Choose file**, select the `Cloudformation-SourceAccountRDS.yaml` file from the GitHub [Crossaccount RDS Replication](https://github.com) repository, and then choose **Next**. 7. For **Stack name**, enter a name for your stack. 8. In the **Parameters** section, specify the parameters that are defined in the stack template: • For **DestinationAccountNumber**, enter the account number for your destination RDS DB instance. • For **KeyName**, enter your AWS KMS key. • For **ScheduleExpression**, enter a cron expression (the default is **12:00 am daily**). • For **SourceDBIdentifier**, enter the name of the source database. • For **SourceDBSnapshotName**, enter the	

Task	Description	Skills required
	name of the snapshot or accept the default. 9. Choose **Next**. 10. On the **Configure stack options** page, leave the default values, and then choose **Next**. 11. Review your stack configuration, and then choose **Submit**. 12. Choose the **Resources** tab for your stack, and then note the Amazon Resource Name (ARN) of the SNS topic.	

Task	Description	Skills required
Deploy the CloudFormation stack in the destination account.	1. Sign in to the AWS Management Console for the destination account (account B) and open the [CloudFormation console](https://console.aws.amazon.com/cloudformation/). 2. In the navigation pane, choose **Stacks**. 3. Choose **Create stack**, and then choose **With existing resources (import resources)**. 4. On the **Identify resources** page, choose **Next**. 5. On the **Specify template** page, select **Upload a template**. 6. Choose **file**, select the `Cloudformation-DestinationAccountRDS.yaml` file from the GitHub [Crossaccount RDS Replication](https://github.com/) repository, and then choose **Next**. 7. For **Stack name**, enter a name for your stack. 8. In the **Parameters** section, specify the parameters that are defined in the stack template: • For **DatabaseName**, enter a name for your database.	Cloud architect, DevOps engineer, Cloud administrator

Task	Description	Skills required
	• For **Engine**, enter the database engine type that matches the source database. • For **DBInstanceClass**, enter the preferred database instance type or accept the default. • For **Subnetgroups**, enter the existing VPC subnet group. For instructions about creating a subnet group, see [Step 2: Create a DB subnet group] in the Amazon RDS documenta tion. • For **SecretName**, enter the path and secret name, or accept the default. • For **SGID**, enter the security group ID of your destination cluster. • For **KMSKey**, enter the ARN of the KMS key in your destination account. • For **NoOfOlderInstances** , enter the number of old copies of the RDS DB instances that you want to keep for the rollback. 9. Choose **Next**.	

Task	Description	Skills required
	10. On the **Configure stack options** page, leave the default values, and then choose **Next**. 11. Review your stack configuration, and then choose **Submit**. 12. Choose the **Resources** tab for your stack, and then note the Physical ID and ARN of `InvokeStepFunction` .	
Verify the creation of the RDS DB instance in the destination account.	1. Sign in to the AWS Management Console and open the [Amazon RDS console](https://console.aws.amazon.com/rds/). 2. In the navigation pane, choose **Databases**, and then verify that the new RDS DB instance appears under the new cluster.	Cloud administrator, Cloud architect, DevOps engineer

Task	Description	Skills required

Subscribe the Lambda function to the SNS topic.	You must run the following AWS Command Line Interface (AWS CLI) commands to subscribe the Lambda function in the destination account (account B) to the SNS topic in the source account (account A).
	
	In account A, run following command:
	
	`aws sns add-permission \ --label lambda-access --aws-account-id <DestinationAccount> \ --topic-arn <Arn of SNSTopic > \ --action-name Subscribe ListSubscriptionsByTopic`
	
	In account B, run following command:
	
	`aws lambda add-permission \ --function-name <Name of InvokeStepFunction > \ --source-arn <Arn of SNSTopic > \ --statement-id function-with-sns \ --action lambda:InvokeFunction \`
		Cloud administrator, Cloud architect, DBA

Task	Description	Skills required

	`--principal sns.amazonaws.com`
	
	In account B, run following command:
	
	`aws sns subscribe \ --protocol "lambda" \ --topic-arn <Arn of SNSTopic> \ --notification-endpoint <Arn of InvokeStepFunction>`	

Task	Description	Skills required
Sync the RDS DB instance from the source account with the destination account.	Initiate the on-demand database replication by starting the Step Functions state machine in the source account. 1. Open the [Step Functions console](https://console.aws.amazon.com/states/). 2. In the navigation pane, choose **State machines**. 3. Choose your state machine. 4. On the **Executions** tab, select your function, and then choose **Start execution** to start the workflow. **Note** A scheduler is in place to help you run the replication automatically on schedule, but the scheduler is turned off by default. You can find the name of the Amazon CloudWatch rule for the scheduler in the **Resources** tab of the CloudFormation stack in the destination account. For instructions on how to	Cloud architect, DevOps engineer, Cloud administrator

Task	Description	Skills required
	modify the CloudWatch Events rule, see Deleting or Disabling a CloudWatch Events Rule in the CloudWatch documentation.	
Roll back your database to any of the previous copies when needed.	1. Open the Secrets Manager console. 2. From the list of secrets, choose the secret that you created by using the CloudFormation template earlier. Your application uses the secret to access the database in the destination cluster. 3. To update the secret value from the details page, in the **Secret value** section, choose **Retrieve secret** value, and then choose **Edit**. 4. Enter the details of the database endpoint.	Cloud administrator, DBA, DevOps engineer

Related resources

- Cross-Region read replicas (Amazon RDS documentation)
- Blue/Green Deployments (Amazon RDS documentation)

Additional information

You can use the following example policy to share your AWS KMS key across AWS accounts.

```
{
    "Version": "2012-10-17",
    "Id": "cross-account-rds-kms-key",
    "Statement": [
        {
            "Sid": "Enable user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<SourceAccount>:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow administration of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<DestinationAccount>:root"
            },
            "Action": [
                "kms:Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms:List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms:Delete*",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
```

```
                    "arn:aws:iam::<DestinationAccount>:root",
                    "arn:aws:iam::<SourceAccount>:root"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey",
                "kms:CreateGrant"
            ],
            "Resource": "*"
        }
    ]
}
```

Automate database tasks in SQL Server Express edition running on Amazon EC2 by using AWS Lambda and Task Scheduler

Subhani Shaik, Amazon Web Services

Summary

This pattern demonstrates how to schedule and manage database tasks in SQL Server Express edition, which is the free version of SQL Server. However, SQL Server Express edition lacks the SQL Server Agent service that typically handles automated database operations. This pattern explains how you can use Task Scheduler and Lambda as an alternative for automating database tasks in SQL Server Express edition running on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

Task Scheduler is a built-in Windows system utility that facilitates the automatic execution of routine tasks. It provides a mechanism to schedule and manage automated operations, eliminating the need for manual intervention in recurring processes. AWS Lambda is a serverless computing service that automatically runs code in response to events, without requiring you to manage the underlying infrastructure.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC) created with Amazon Virtual Private Cloud (Amazon VPC)
- An Amazon EC2 instance with Windows Server
- Amazon Elastic Block Store (Amazon EBS) volumes that are attached to an Amazon EC2 instance with Windows Server
- SQL Server Express Edition binaries

Limitations

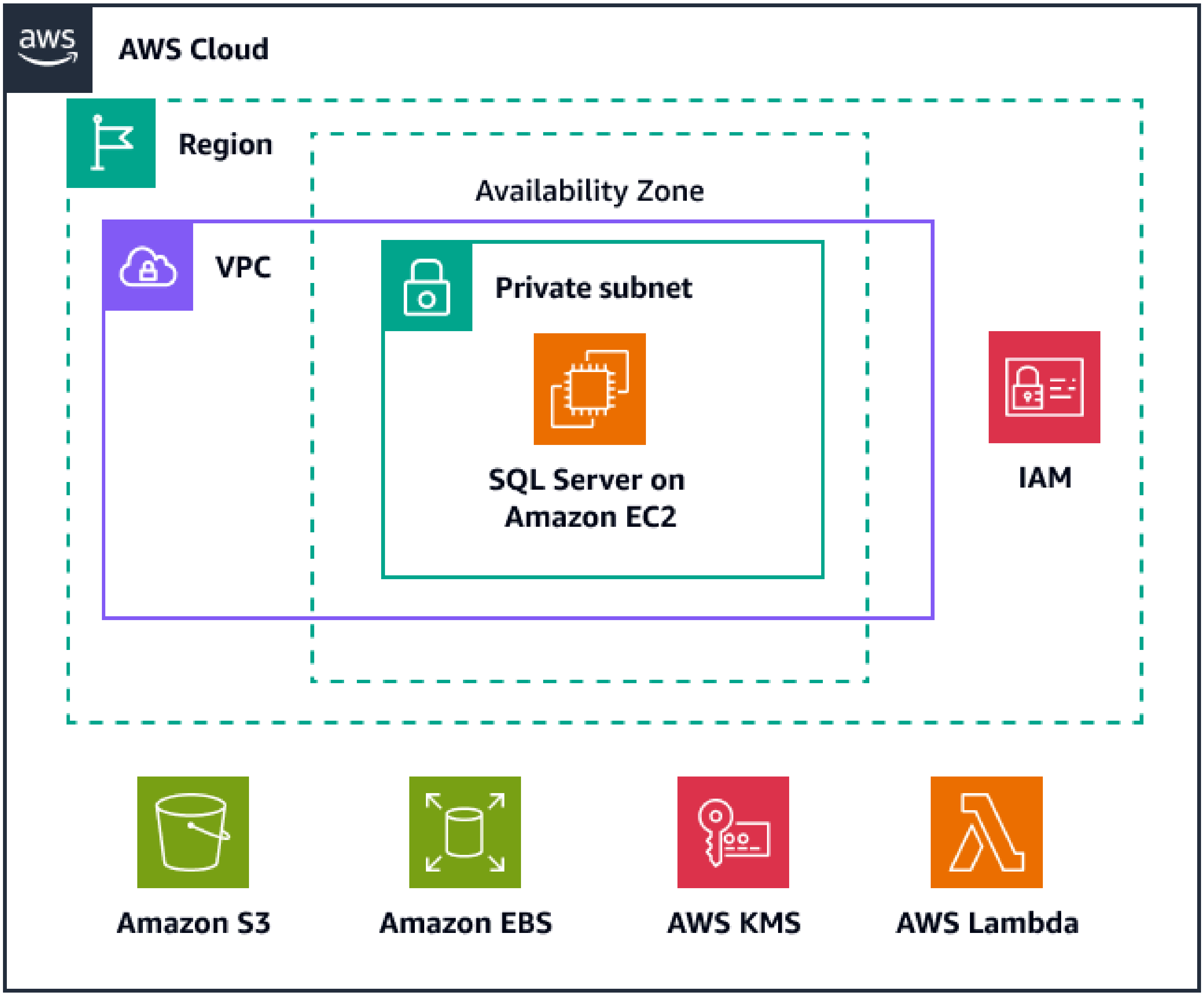
- For information about feature limitations of SQL Server Express edition, see the Microsoft website.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- SQL Server 2016 or later with SQL Server Express edition

Architecture

The following diagram shows an Amazon EC2 instance running with SQL Server Express edition installed. The instance is accessible through Remote Desktop Protocol (RDP) client or from AWS Systems Manager Session Manager. AWS Key Management Service (AWS KMS) handles the data encryption for the Amazon EBS volumes to ensure data-at-rest security. The infrastructure also includes AWS Identity and Access Management (IAM), which provides access control and manages permissions for the execution of Lambda functions. Amazon Simple Storage Service (Amazon S3) stores Lambda functions.



Tools

AWS services

- [Amazon Elastic Block Store (Amazon EBS)](#) provides block-level storage volumes for use with Amazon EC2 instances.
- [Amazon Elastic Compute Cloud (Amazon EC2)](#) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Systems Manager Session Manager is a fully managed AWS Systems Manager tool. With Session Manager, you can manage your Amazon EC2 instances, edge devices, on-premises servers, and virtual machines (VMs).
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- Microsoft SQL Server Management Studio (SSMS) is a tool for managing SQL Server, including accessing, configuring, and administering SQL Server components.
- Python is a general-purpose computer programming language. You can use it to build applications, automate tasks, and develop services on the AWS Cloud.
- Task Scheduler is a Microsoft tool that you can use to schedule routine tasks on your computer automatically.

Best practices

- Best practices for Amazon EC2
- Best practices for deploying Microsoft SQL Server on Amazon EC2
- Best practices for working with AWS Lambda functions
- Security best practices in IAM

Epics

Create an Amazon EC2 instance and install SQL Server Express edition

Task	Description	Skills required
Deploy an Amazon EC2 instance.	To create an Amazon EC2 instance, open the Amazon EC2 console at https://console.aws.amazon.com/ec2/ and select an [Amazon Machine Image (AMI)]() from the list of instances available for Windows Server. For more information, see [Launch an Amazon EC2 instance]() in the AWS documentation.	DBA, AWS DevOps
Install SQL Server Express edition.	To install SQL Server Express edition, complete the following steps: 1. To [connect to your Amazon EC2 instance](), choose an option: • **Option A** – Use Remote Desktop Protocol (RDP). For instructions, see [Connect to your Windows instance using an RDP client]() in the AWS documentation. • **Option B** – Use the Amazon EC2 console and AWS Systems Manager	DBA, AWS DevOps

Task	Description	Skills required
	Session Manager. For instructions, see Connect to your Amazon EC2 instance using Session Manager in the AWS documentation. 2. To download the required SQL Server Express edition, go to SQL Server Downloads on the Microsoft website. 3. To install SQL Server Express edition, follow the instructions in Plan a SQL Server installation on the Microsoft website.	

Create automated database maintenance tasks

Task	Description	Skills required
Identify routine tasks.	Identify the routine tasks that you want to automate. For example, the following tasks are eligible for automation: • Database backups (Full, differential, and transaction log) • Index maintenance and reorganization • Statistics updates • Application-specific operations	DBA

Task	Description	Skills required
	• Data cleanup or archiving	
Prepare SQL scripts.	To prepare SQL scripts, do the following: 1. Create SQL queries for each maintenance task. Following is an example [T-SQL](https://learn.microsoft.com/en-us/sql/t-sql/language-reference) query to perform a specific database backup: `Backup Database <Database_Name> To Disk='C:\Backups\Database_Name.bak'` 2. Save the script file as `<File Name>.sql` . Then, save the scripts to an accessible location on the server's local drive on the Amazon EC2 instance or a network file share.	DBA

Task	Description	Skills required
Configure access permissions.	To configure access permissio ns, do the following: 1. Set appropriate file system permissions. For instructi ons, see Configure file system permissions for Database Engine access on the Microsoft website. 2. Check that the SQL Server service account has the necessary access. For instructions, see Configure Windows service accounts and permissions on the Microsoft website. 3. Verify network connectiv ity for remote shares. For more information, see Accessing data using file shares in the AWS documentation.	DBA

Automate tasks with Task Scheduler

Task	Description	Skills required
Create batch files.	• To create a batch file, use a text editor to type the following command. Replace the parameters `username` and `password` with your own values. Then	AWS DevOps, DBA

Task	Description	Skills required

	save the file as <Name>.bat .
	`sqlcmd -S servername -U username -P password -i <T-SQL query path.sql>`
	• To create a batch file for SQL tasks, use a text editor and type the following commands. Replace the parameters ServerName , DatabaseName , username, and password with your own values. Then save the file as <Name>.bat .
	`@echo off sqlcmd -S [ServerName] -d [DatabaseName] -U username -P password -i "PathToSQLScript\Script.sql" -o "PathToOutput\Output.txt"`	

Task	Description	Skills required
Create tasks in Task Scheduler .	To create a task in Task Scheduler, use the following steps: 1. To open Task Scheduler , type **taskschd.msc** in Windows search. 2. Choose the **Action** menu, and then select **Create Basic Task**. 3. For **Name**, provide a name for the task, and then choose **Next**. 4. For **Trigger**, select the option for when you want the task to start and then choose **Next**. 5. Provide the **Start** and **Recur** information for the task, and then choose **Next**. 6. For the **Action** section, select **Start a program** and then choose **Next**. 7. For **Program/script**, specify the path for the batch file that you created in the previous task and then choose **Next**. 8. Choose **Finish**.	DBA

Task	Description	Skills required
	To run the task manually, right-click the newly created task and then select **Run**.	
View task status.	To view the status of a task in Task Scheduler, use the following steps: 1. In Task Scheduler, go to **Task Scheduler Library**, which displays all tasks. 2. To see the status of the task that you created earlier, select the task and then go to the **History** tab.	DBA, AWS DevOps

Automate tasks with AWS Lambda

Task	Description	Skills required
Implement the solution.	To implement this pattern's solution, use the following steps: 1. Create a Lambda function. For instructions, see [Create your first Lambda function] in the AWS documentation. 2. Schedule the Lambda function. For instructi ons, see [Invoke a Lambda function on a schedule] in the AWS documentation. 3. Run T-SQL queries. For more information, see	AWS DevOps, DevOps engineer

Task	Description	Skills required
	Tutorial: Using a Lambda function to access an Amazon RDS database in the AWS documentation. The tutorial explains how to connect Amazon RDS databases from Lambda functions to run SQL queries	

Troubleshooting

Issue	Solution
Lambda issues	For help with errors and issues that you might encounter when using AWS Lambda, see Troubleshooting issues in Lambda in the AWS documentation.

Related resources

- Amazon EC2 instance types
- AWS Lambda documentation
- AWS Lambda pricing
- Task Scheduler for developers (Microsoft website)

Automate cross-Region failover and failback by using DR Orchestrator Framework

Jitendra Kumar, Pavithra Balasubramanian, and Oliver Francis, Amazon Web Services

Summary

This pattern describes how to use DR Orchestrator Framework to orchestrate and automate the manual, error-prone steps to perform disaster recovery across Amazon Web Services (AWS) Regions. The pattern covers the following databases:

- Amazon Relational Database Service (Amazon RDS) for MySQL, Amazon RDS for PostgreSQL, or Amazon RDS for MariaDB
- Amazon Aurora MySQL-Compatible Edition or Amazon Aurora PostgreSQL-Compatible Edition (using a centralized file)
- Amazon ElastiCache (Redis OSS)

To demonstrate the functionality of DR Orchestrator Framework, you create two DB instances or clusters. The primary is in the AWS Region `us-east-1`, and the secondary is in `us-west-2`. To create these resources, you use the AWS CloudFormation templates in the `App-Stack` folder of the aws-cross-region-dr-databases GitHub repository.

Prerequisites and limitations

General prerequisites

- DR Orchestrator Framework deployed in both primary and secondary AWS Regions
- Two Amazon Simple Storage Service buckets
- A virtual private cloud (VPC) with two subnets and an AWS security group

Engine-specific prerequisites

- **Amazon Aurora** – At least one Aurora global database must be available in two AWS Regions. You can use `us-east-1` as the primary Region, and use `us-west-2` as the secondary Region.
- **Amazon ElastiCache (Redis OSS)** – An ElastiCache global datastore must be available in two AWS Regions. You can `use us-east-1` as the primary Region, and use `us-west-2` as the secondary Region.

Amazon RDS limitations

- DR Orchestrator Framework doesn't check the replication lag before doing a failover or failback. Replication lag must be checked manually.
- This solution has been tested using a primary database instance with one read replica. If you want to use more than one read replica, test the solution thoroughly before implementing it in a production environment.

Aurora limitations

- Feature availability and support vary across specific versions of each database engine and across AWS Regions. For more information on feature and Region availability for cross-Region replication, see Cross-Region read replicas.
- Aurora global databases have specific configuration requirements for supported Aurora DB instance classes and the maximum number of AWS Regions. For more information, see Configuration requirements of an Amazon Aurora global database.
- This solution has been tested using a primary database instance with one read replica. If you want to use more than one read replica, test the solution thoroughly before implementing it in a production environment.

ElastiCache limitations

- For information about Region availability for Global Datastore and ElastiCache configuration requirements, see Prerequisites and limitations in the ElastiCache documentation.

Amazon RDS product versions

Amazon RDS supports the following engine versions:

- **MySQL –** Amazon RDS supports DB instances running the following versions of MySQL: MySQL 8.0 and MySQL 5.7
- **PostgreSQL –** For information about supported versions of Amazon RDS for PostgreSQL, see Available PostgreSQL database versions.
- **MariaDB –** Amazon RDS supports DB instances running the following versions of MariaDB:
  - MariaDB 10.11
  - MariaDB 10.6

- MariaDB 10.5

Aurora product versions

- Amazon Aurora global database switchover requires Aurora MySQL-Compatible with MySQL 5.7 compatibility, version 2.09.1 and higher

  For more information, see Limitations of Amazon Aurora global databases.

ElastiCache (Redis OSS) product versions

Amazon ElastiCache (Redis OSS) supports the following Redis versions:

- Redis 7.1 (enhanced)
- Redis 7.0 (enhanced)
- Redis 6.2 (enhanced)
- Redis 6.0 (enhanced)
- Redis 5.0.6 (enhanced)

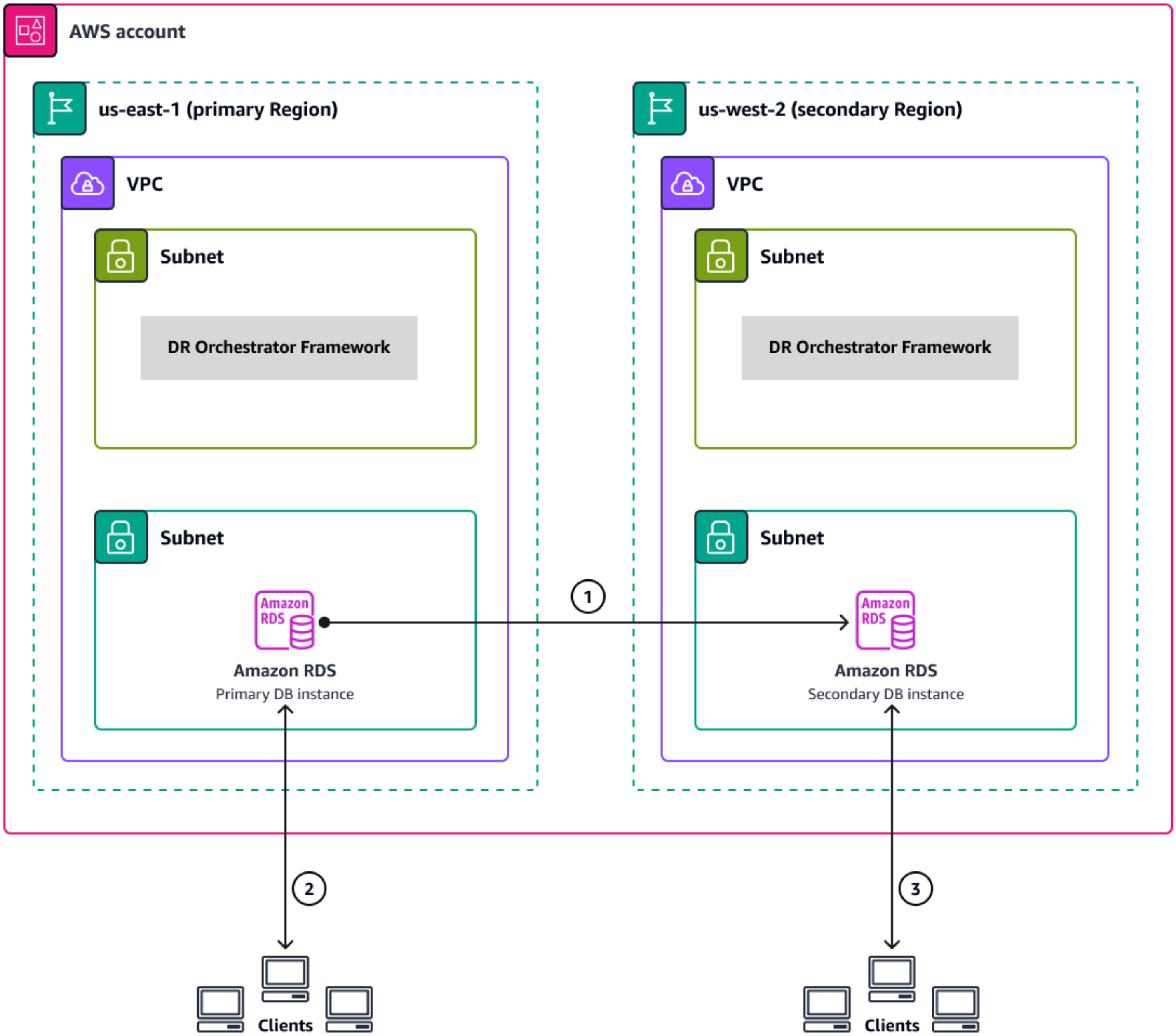
For more information, see Supported ElastiCache (Redis OSS) versions.

Architecture

Amazon RDS architecture

The Amazon RDS architecture includes the following resources:

- The primary Amazon RDS DB instance created in the primary Region (`us-east-1`) with read/write access for clients
- An Amazon RDS read replica created in the secondary Region (`us-west-2`) with read-only access for clients
- DR Orchestrator Framework deployed in both the primary and secondary Regions



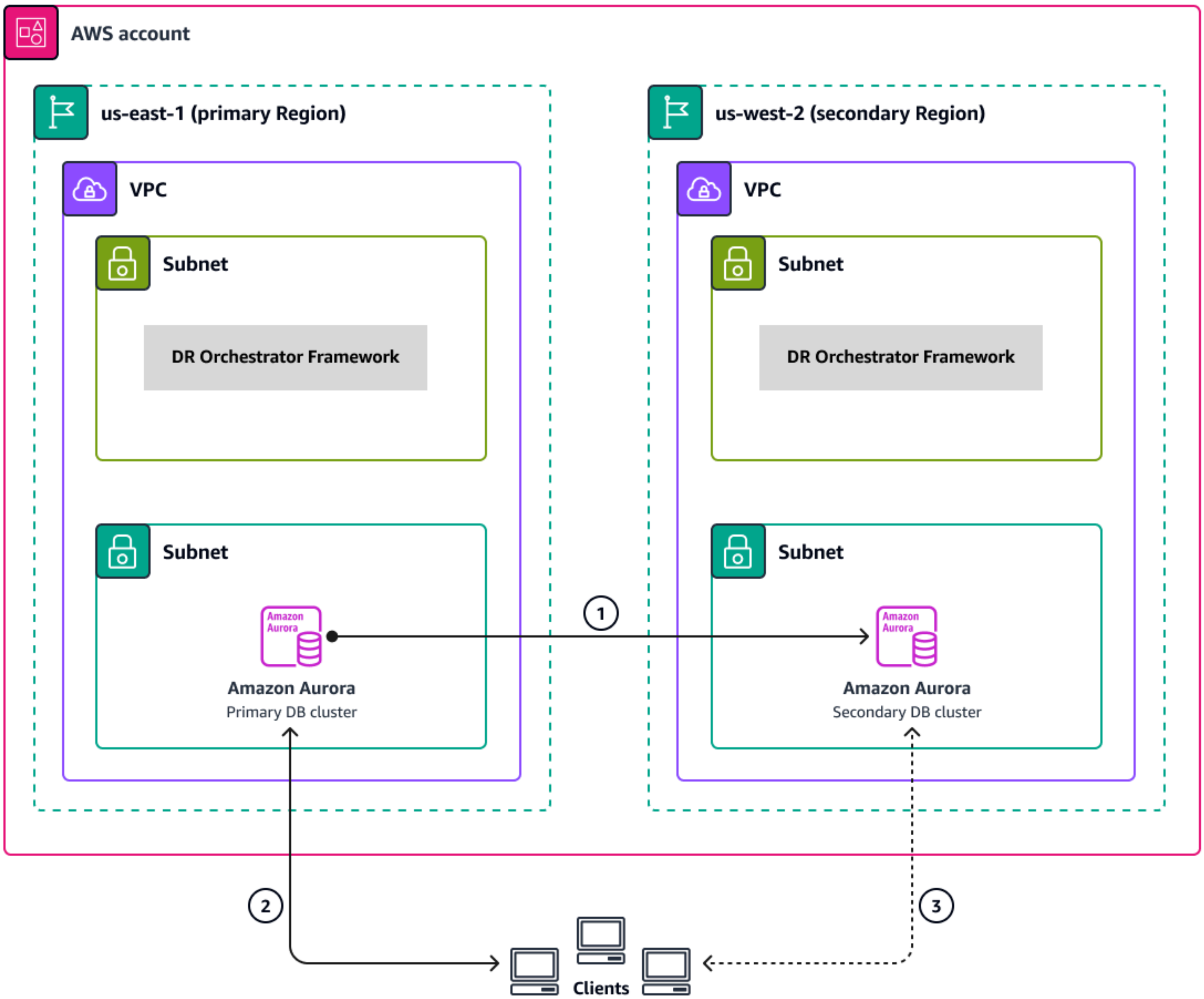
The diagram shows the following:

1. Asynchronous replication between the primary instance and the secondary instance
2. Read/write access for clients in the primary Region
3. Read-only access for clients in the secondary Region

Aurora architecture

The Amazon Aurora architecture includes the following resources:

- The primary Aurora DB cluster created in the primary Region (`us-east-1`) with an active-writer endpoint
- An Aurora DB cluster created in the secondary Region (`us-west-2`) with an inactive-writer endpoint
- DR Orchestrator Framework deployed in both the primary and secondary Regions



The diagram shows the following:

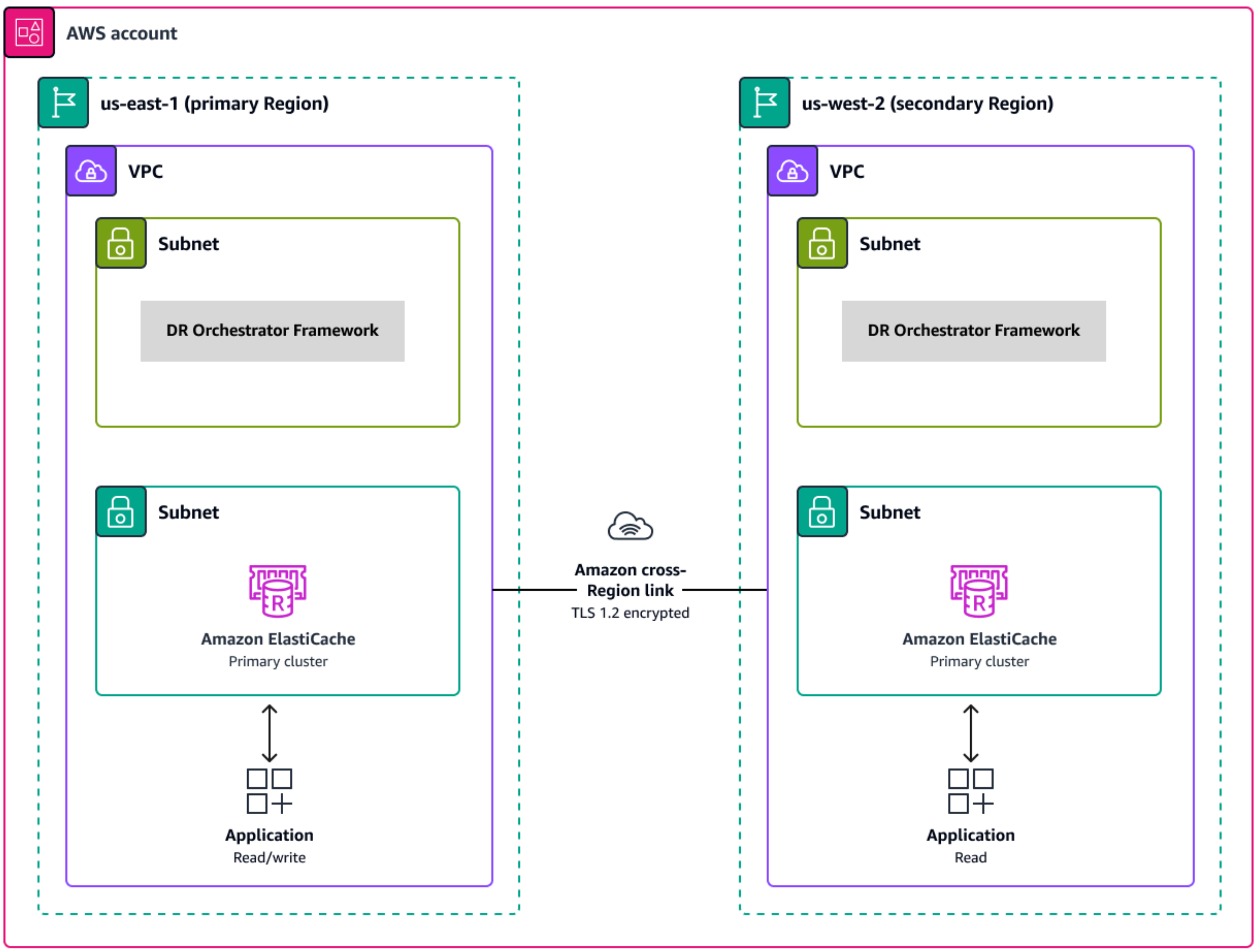
1. Asynchronous replication between the primary cluster and the secondary cluster
2. The primary DB cluster with an active-writer endpoint

3. The secondary DB cluster with an inactive-writer endpoint

ElastiCache (Redis OSS) architecture

The Amazon ElastiCache (Redis OSS) architecture includes the following resources:

- An ElastiCache (Redis OSS) global datastore created with two clusters:
  1. The primary cluster in the primary Region (`us-east-1`)
  2. The secondary cluster in the secondary Region (`us-west-2`)
- An Amazon cross-Region link with TLS 1.2 encryption between the two clusters
- DR Orchestrator Framework deployed in both primary and secondary Regions



Automation and scale

DR Orchestrator Framework is scalable and supports the failover or failback of more than one AWS database in parallel.

You can use the following payload code to fail over multiple AWS databases in your account. In this example, three AWS databases (two global databases such as Aurora MySQL-Compatible or Aurora PostgreSQL-Compatible, and one Amazon RDS for MySQL instance) fail over to the DR Region:

```
{
  "StatePayload": [
    {
      "layer": 1,
      "resources": [
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global
 databases (MySQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import dr-globaldb-cluster-mysql-global-
identifier",
            "DBClusterIdentifier": "!Import dr-globaldb-cluster-mysql-cluster-
identifier"
          }
        },
        {
          "resourceType": "PlannedFailoverAurora",
          "resourceName": "Switchover (planned failover) of Amazon Aurora global
 databases (PostgreSQL)",
          "parameters": {
            "GlobalClusterIdentifier": "!Import dr-globaldb-cluster-postgres-global-
identifier",
            "DBClusterIdentifier": "!Import dr-globaldb-cluster-postgres-cluster-
identifier"
          }
        },
        {
          "resourceType": "PromoteRDSReadReplica",
          "resourceName": "Promote RDS for MySQL Read Replica",
          "parameters": {
            "RDSInstanceIdentifier": "!Import rds-mysql-instance-identifier",
            "TargetClusterIdentifier": "!Import rds-mysql-instance-global-arn"
          }
```

```
        }
      ]
    }
  ]
}
```

Tools

AWS services

- Amazon Aurora is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL.
- Amazon ElastiCache helps you set up, manage, and scale distributed in-memory cache environments in the AWS Cloud. This pattern uses Amazon ElastiCache (Redis OSS).
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use. In this pattern, Lambda functions are used by AWS Step Functions to perform the steps.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud. This pattern supports Amazon RDS for MySQL, Amazon RDS for PostgreSQL, and Amazon RDS for MariaDB.
- AWS SDK for Python (Boto3) helps you integrate your Python application, library, or script with AWS services. In this pattern, Boto3 APIs are used to communicate with the database instances or global databases.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications. In this pattern, Step Functions state machines are used to orchestrate and run the cross-Region failover and failback of the database instances or global databases.

Code repository

The code for this pattern is available in the aws-cross-region-dr-databases repository on GitHub.

Epics

Install DR Orchestrator Framework

Task	Description	Skills required
Clone the GitHub repository.	To clone the repository, run the following command: `git clone https://github.com/aws-samples/aws-cross-region-dr-databases.git`	AWS DevOps, AWS administrator

Package Lambda functions code in a .zip file archive.	Create the archive files for Lambda functions to include the DR Orchestrator Framework dependencies:
	`cd <YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts`
	
	`bash scripts/deploy-orchestrator-sh.sh`
		AWS administrator

Create S3 buckets.	S3 buckets are needed to store DR Orchestrator Framework along with your latest configuration. Create two S3 buckets, one in the primary Region (`us-east-1`), and one in the secondary Region (`us-west-2`): • `dr-orchestrator-xxxxxx-us-east-1`	AWS administrator

Task	Description	Skills required
	• `dr-orchestrator-xxxxxx-us-west-2` Replace `xxxxxx` with a random value to make the bucket names unique.	
Create subnets and security groups.	In both the primary Region (`us-east-1`) and the secondary Region (`us-west-2`), create two subnets and one security group for Lambda function deployment in your VPC: • `subnet-XXXXXXX` • `subnet-YYYYYYY` • `sg-XXXXXXXXXXXX`	AWS administrator

Task	Description	Skills required
Update the DR Orchestrator parameter files.	In the `<YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts/cloudformation` folder, update the following DR Orchestrator parameter files: • `Orchestrator-Deployer-parameters-us-east-1.json` • `Orchestrator-Deployer-parameters-us-west-2.json` Use the following parameter values, replacing x and y with the names of your resources: `[ { "ParameterKey": "TemplateStoreS3BucketName", "ParameterValue": "dr-orchestrator-xxxxxx-us-east-1" }, { "ParameterKey": "TemplateVPCId", "ParameterValue": "vpc-xxxxxx" }, {`	AWS administrator

Task	Description	Skills required

	``
	        "ParameterKey":
	 "TemplateLambdaSub
	netID1",
	        "Paramete
	rValue": "subnet-x
	xxxxx"
	    },
	    {
	        "ParameterKey":
	 "TemplateLambdaSub
	netID2",
	        "Paramete
	rValue": "subnet-y
	yyyyy"
	    },
	    {
	        "ParameterKey":
	 "TemplateLambdaSec
	urityGroupID",
	        "Paramete
	rValue": "sg-xxxxx
	xxxxx"
	    }
	]
	``	

Task	Description	Skills required
Upload the DR Orchestrator Framework code to the S3 bucket.	The code will be safer in an S3 bucket than in the local directory. Upload the `DR-Orchestration-artifacts` directory, including all files and subfolders, to the S3 buckets. To upload the code, do the following: 1. Sign in to the AWS Management Console. 2. Navigate to the Amazon S3 console. 3. Select the `dr-orchestrator-xxxxxx-us-east-1 bucket` . 4. Choose **Upload**, and then choose **Add folder**. 5. Select the `DR-Orchestration-artifacts` folder. 6. Choose **Upload**. 7. Select the `dr-orchestrator-xxxxxx-us-west-2` bucket. 8. Repeat steps 4–7.	AWS administrator

Task	Description	Skills required

Deploy DR Orchestrator Framework in the primary Region.	To deploy DR Orchestrator Framework in the primary Region (`us-east-1`), run the following commands:
	`cd <YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts/cloudformation`
	
	`aws cloudformation deploy \`
	`--region us-east-1 \`
	`--stack-name dr-orchestrator \`
	`--template-file Orchestrator-Deployer.yaml \`
	`--parameter-overrides file://Orchestrator-Deployer-parameters-us-east-1.json \`
	`--capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \`
	`--disable-rollback`
		AWS administrator

Task	Description	Skills required

Deploy DR Orchestrator Framework in the secondary Region.	In the secondary Region (`us-west-2`), run the following commands:
	`cd <YOUR-LOCAL-GIT-FOLDER>/DR-Orchestration-artifacts/cloudformation`
	
	`aws cloudformation deploy \`
	`--region us-west-2 \`
	`--stack-name dr-orchestrator \`
	`--template-file Orchestrator-Deployer.yaml \`
	`--parameter-overrides file://Orchestrator-Deployer-parameters-us-west-2.json \`
	`--capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \`
	`--disable-rollback`
		AWS administrator

Task	Description	Skills required
Verify the deployment.	If the AWS CloudFormation command runs successfully, it returns the following output: `Successfully created/updated stack - dr-orchestrator` Alternatively, you can navigate to the AWS CloudFormation console and verify the status of the `dr-orchestrator` stack.	AWS administrator

Create the database instances or clusters

Task	Description	Skills required
Create the database subnets and security groups.	In your VPC, create two subnets and one security group for the DB instance or global database in both the primary (`us-east-1`) and the secondary (`us-west-2`) Regions: • `subnet-XXXXXX` • `subnet-XXXXXX` • `sg-XXXXXXXXXX`	AWS administrator
Update the parameter file for the primary DB instance or cluster.	In the `<YOUR LOCAL GIT FOLDER>/App-Stack` folder, update the parameter file for the primary Region.	AWS administrator

Task	Description	Skills required
	Amazon RDS In the `RDS-MySQL-parameter-us-east-1.json` file, update `SubnetIds` and `DBSecurityGroup` with the names of resources that you created: `{ "Parameters": { "SubnetIds": "subnet-xxxxxx,subnet-xxxxxx", "DBSecurityGroup": "sg-xxxxxxxxxx", "MySqlGlobalIdentifier":"rds-mysql-instance", "InitialDatabaseName": "mysqldb", "DBPortNumber": "3789", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2", "KMSKeyAliasName": "rds/rds-mysql-instance-KmsKeyId" } }`  **Amazon Aurora**  In the `Aurora-MySQL-parameter-us-east-1.json` file, update `SubnetIds` and `DBSecurit`	

Task	Description	Skills required
	`yGroup` with the names of resources that you created: `{ "Parameters": { "SubnetIds": "subnet1-xxxxxx,subnet2-xxxxxx", "DBSecurityGroup": "sg-xxxxxxxxxx", "GlobalClusterIdentifier":"dr-globaldb-cluster-mysql", "DBClusterName":"dbcluster-01", "SourceDBClusterName":"dbcluster-02", "DBPortNumber": "3787", "DBInstanceClass": "db.r5.large", "InitialDatabaseName": "sampledb", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2", "KMSKeyAliasName": "rds/dr-globaldb-cluster-mysql-KmsKeyId" } }` **Amazon ElastiCache (Redis OSS)** In the `ElastiCache-parameter-us-east-1.json` file, update	

Task	Description	Skills required
	`SubnetIds` and `DBSecurityGroup` with the names of resources that you created. `{ "Parameters": { "CacheNodeType": "cache.m5.large", "DBSecurityGroup": "sg-xxxxxxxxxx", "SubnetIds": "subnet-xxxxxx,subnet-xxxxxx", "EngineVersion": "5.0.6", "GlobalReplicationGroupIdSuffix": "demo-redis-global-datastore", "NumReplicas": "1", "NumShards": "1", "ReplicationGroupId": "demo-redis-cluster", "DBPortNumber": "3788", "TransitEncryption": "true", "KMSKeyAliasName": "elasticache/demo-redis-global-datastore-KmsKeyId", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2" } }`	

Task	Description	Skills required

Deploy your DB instance or cluster in the primary Region.	To deploy your instance or cluster in the primary Region (`us-east-1`), run the following commands based on your database engine.
	
	Amazon RDS
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \`
	`--region us-east-1 \`
	`--stack-name rds-mysql-app-stack \`
	`--template-file RDS-MySQL-Primary.yaml \`
	`--parameter-overrides file://RDS-MySQL-parameter-us-east-1.json \`
	`--capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \`
	`--disable-rollback`
	
	Amazon Aurora
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \`
	`--region us-east-1 \`
	`--stack-name aurora-mysql-app-stack \`
		AWS administrator

Task	Description	Skills required
	``` --template-file Aurora- MySQL-Primary.yaml \ --parameter-overrides file://Aurora-MySQ L-parameter-us-eas t-1.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback ``` **Amazon ElastiCache (Redis OSS)** ``` cd <YOUR-LOCAL-GIT-FO LDER>/App-Stack aws cloudformation deploy \ --region us-east-1 -- stack-name elasticac he-ds-app-stack \ --template-file ElastiCache-Primar y.yaml \ --parameter-overrides file://ElastiCache -parameter-us-east -1.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback ```	

Task	Description	Skills required
	Verify that the AWS CloudFormation resources deployed successfully.	

Task	Description	Skills required

Update the parameter file for the secondary DB instance or cluster.	In the `<YOUR LOCAL GIT FOLDER>/App-Stack` folder, update the parameter file for the secondary Region.
	
	Amazon RDS
	
	In the `RDS-MySQL-parameter-us-west-2.json` file, update `SubnetIDs` and `DBSecurityGroup` with the names of resources that you created. Update the `PrimaryRegionKMSKeyArn` with the value of `MySQLKmsKeyId` taken from the **Outputs** section of the AWS CloudFormation stack for the primary DB instance:
	
	`{`
	`  "Parameters": {`
	`    "SubnetIds":`
	` "subnet-aaaaaaaaa,`
	`subnet-bbbbbbbbb",`
	`    "DBSecurityGroup":`
	` "sg-cccccccccc",`
	`    "MySqlGlobalIdenti`
	`fier":"rds-mysql-i`
	`nstance",`
	`    "InitialDatabaseNa`
	`me": "mysqldb",`
	`    "DBPortNumber":`
	` "3789",`
	`    "PrimaryRegion":`
	` "us-east-1",`
		AWS administrator

Task	Description	Skills required

	``
	    "SecondaryRegion":
	  "us-west-2",
	    "KMSKeyAliasName":
	  "rds/rds-mysql-ins
	tance-KmsKeyId",
	    "PrimaryRegionKMSK
	eyArn":"arn:aws:km
	s:us-east-1:xxxxxx
	xxx:key/mrk-xxxxxx
	xxxxxxxxxxxxxxx"
	  }
	}
	``
	
	Amazon Aurora
	
	In the `Aurora-MySQL-parameter-us-west-2.json` file, update `SubnetIDs` and `DBSecurityGroup` with the names of resources you created. Update the `PrimaryRegionKMSKeyArn` with the value of `AuroraKmsKeyId` taken from the **Outputs** section of the AWS CloudFormation stack for the primary DB instance:
	
	``
	{
	  "Parameters": {
	    "SubnetIds":
	  "subnet1-aaaaaaaaa
	,subnet2-bbbbbbbbb",
	    "DBSecurityGroup":
	  "sg-cccccccccc",
	    "GlobalClusterIden
	tifier":"dr-globaldb-
	cluster-mysql",
	``	

Task	Description	Skills required

	``
	    "DBClusterName":"dbcluster-01",
	    "SourceDBClusterName":"dbcluster-02",
	    "DBPortNumber": "3787",
	    "DBInstanceClass": "db.r5.large",
	    "InitialDatabaseName": "sampledb",
	    "PrimaryRegion": "us-east-1",
	    "SecondaryRegion": "us-west-2",
	    "KMSKeyAliasName": "rds/dr-globaldb-cluster-mysql-KmsKeyId"
	  }
	}
	``
	Amazon ElastiCache (Redis OSS)
	In the `ElastiCache-parameter-us-west-2.json` file, update `SubnetIDs` and `DBSecurityGroup` with the names of resources that you created. Update the `PrimaryRegionKMSKeyArn` with the value of `ElastiCacheKmsKeyId` taken from the **Outputs** section of the AWS CloudFormation stack for the primary DB instance:	

Task	Description	Skills required
	`{ "Parameters": { "CacheNodeType": "cache.m5.large", "DBSecurityGroup": "sg-cccccccccc", "SubnetIds": "subnet-aaaaaaaaa, subnet-bbbbbbbbb", "EngineVersion": "5.0.6", "GlobalReplicationGroupIdSuffix": "demo-redis-global-datastore", "NumReplicas": "1", "NumShards": "1", "ReplicationGroupId": "demo-redis-cluster", "DBPortNumber": "3788", "TransitEncryption": "true", "KMSKeyAliasName": "elasticache/demo-redis-global-datastore-KmsKeyId", "PrimaryRegion": "us-east-1", "SecondaryRegion": "us-west-2" } }`	

Task	Description	Skills required

Deploy your DB instance or cluster in the secondary Region.	Run the following commands, based on your database engine.
	
	Amazon RDS
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \ --region us-west-2 \ --stack-name rds-mysql-app-stack \ --template-file RDS-MySQL-DR.yaml \ --parameter-overrides file://RDS-MySQL-parameter-us-west-2.json \ --capabilities CAPABILITY_AUTO_EXPAND CAPABILITY_NAMED_IAM CAPABILITY_IAM \ --disable-rollback`
	
	Amazon Aurora
	
	`cd <YOUR-LOCAL-GIT-FOLDER>/App-Stack`
	
	`aws cloudformation deploy \ --region us-west-2 \ --stack-name aurora-mysql-app-stack \ --template-file Aurora-MySQL-DR.yaml \`
		AWS administrator

Task	Description	Skills required
	``` --parameter-overrides file://Aurora-MySQ L-parameter-us-wes t-2.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback ``` **Amazon ElastiCache (Redis OSS)** ``` cd <YOUR-LOCAL-GIT-FO LDER>/App-Stack aws cloudformation deploy \ --region us-west-2 \ --stack-name elasticac he-ds-app-stack \ --template-file ElastiCache-DR.yaml \ --parameter-overrides file://ElastiCache -parameter-us-west -2.json \ --capabilities CAPABILITY_AUTO_EX PAND CAPABILIT Y_NAMED_IAM CAPABILIT Y_IAM \ --disable-rollback ``` Verify that the AWS CloudFormation resources deployed successfully.	

Related resources

- Disaster recovery strategy for databases on AWS (AWS Prescriptive Guidance strategy)
- Automate your DR solution for relational databases on AWS (AWS Prescriptive Guidance guide)
- Using Amazon Aurora global databases
- Replication across AWS Regions using global datastores
- Automate your DR solution for relational databases on AWS (AWS Prescriptive Guidance guide)

Automatically back up SAP HANA databases using Systems Manager and EventBridge

Ambarish Satarkar and Gaurav Rath, Amazon Web Services

Summary

This pattern describes how to automate SAP HANA database backups using AWS Systems Manager, Amazon EventBridge, Amazon Simple Storage Service (Amazon S3), and AWS Backint Agent for SAP HANA.

This pattern provides a shell script-based approach using the `BACKUP DATA` command and removes the need to maintain scripts and job configurations for each operating system (OS) instance across numerous systems.

Note: As of April 2023, AWS Backup announced support for SAP HANA databases on Amazon Elastic Compute Cloud (Amazon EC2). For more information, see [SAP HANA databases on Amazon EC2 instances backup](#).

Based on your organization’s needs, you can use the AWS Backup service to automatically back up your SAP HANA databases or you can use this pattern.

Prerequisites and limitations

Prerequisites

- An existing SAP HANA instance with a supported release in running state on a managed Amazon Elastic Compute Cloud (Amazon EC2) instance that is configured for Systems Manager
- Systems Manager Agent (SSM Agent) 2.3.274.0 or later installed
- An S3 bucket that doesn’t have public access enabled
- An `hdbuserstore` key named `SYSTEM`
- An AWS Identity and Access Management (IAM) role for the Automation runbook to run on schedule
- `AmazonSSMManagedInstanceCore` and `ssm:StartAutomationExecution` policies are attached to Systems Manager Automation service role.

Limitations

- AWS Backint Agent for SAP HANA doesn't support deduplication.
- AWS Backint Agent for SAP HANA doesn't support data compression.

Product versions

AWS Backint Agent is supported on the following operating systems:

- SUSE Linux Enterprise Server
- SUSE Linux Enterprise Server for SAP
- Red Hat Enterprise Linux for SAP

AWS Backint Agent supports the following databases:

- SAP HANA 1.0 SP12 (single node and multiple nodes)
- SAP HANA 2.0 and later (single node and multiple nodes)

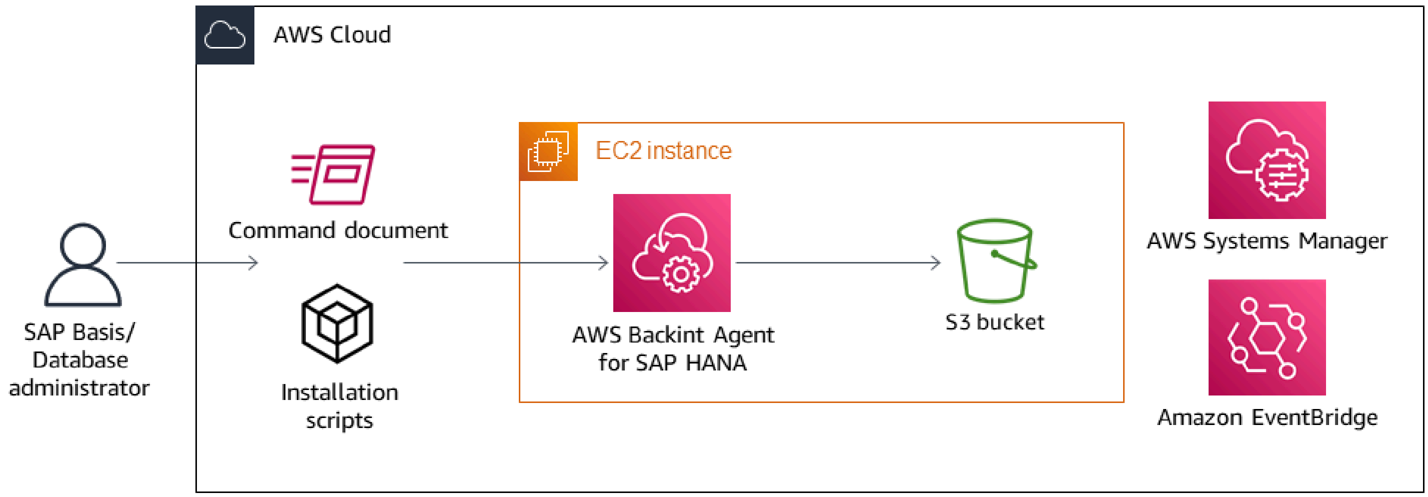
Architecture

Target technology stack

- AWS Backint Agent
- Amazon S3
- AWS Systems Manager
- Amazon EventBridge
- SAP HANA

Target architecture

The following diagram shows the installation scripts that install AWS Backint Agent, the S3 bucket, and Systems Manager and EventBridge, which use a Command document to schedule regular backups.



Automation and scale

- Multiple AWS Backint Agents can be installed by using a Systems Manager Automation runbook.
- Each run of the Systems Manager runbook can scale to *n* number of SAP HANA instances, based on target selection.
- EventBridge can automate SAP HANA backups.

Tools

- [AWS Backint Agent for SAP HANA](#) is a standalone application that integrates with your existing workflows to back up your SAP HANA database to an S3 bucket that you specify in the configuration file. AWS Backint Agent supports full, incremental, and differential backups of SAP HANA databases. It runs on an SAP HANA database server, where backups and catalogs are transferred from the SAP HANA database to the AWS Backint Agent.
- [Amazon EventBridge](#) is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your applications, software as a service (SaaS) applications, and AWS services to targets such as AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other accounts.
- [Amazon Simple Storage Service (Amazon S3)](#) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- [AWS Systems Manager](#) helps you to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources.

Code

The code for this pattern is available in the aws-backint-automated-backup GitHub repository.

Epics

Create an hdbuserstore key SYSTEM

Task	Description	Skills required

Create an hdbuserstore key.	1. Navigate to `/usr/sap/<SID>/HDB<Inst No>/exe`.
	2. Run the following command, with XX as the SAP HANA database instance number.
	`hdbuserstore -i set SYSTEM <hostname>:3XX13@SYSTEMDB SYSTEM`
	For example, for an SAP HANA host `saphanadb` with instance number `00`, run the following command.
	`hdbuserstore -i set SYSTEM saphanadb:30013@SYSTEMDB SYSTEM`
		AWS administrator, SAP HANA Administrator

Install AWS Backint Agent

Task	Description	Skills required
Install AWS Backint Agent.	Follow the instructions in [Install and configure AWS Backint Agent for SAP HANA]() in the AWS Backint Agent documentation.	AWS administrator, SAP HANA administrator

Create the Systems Manager Command document

Task	Description	Skills required
Create the Systems Manager Command document.	1. Sign in to the AWS Management Console and open the AWS Systems Manager Console. 2. Choose **Documents,** and choose **Owned by me**. 3. Confirm that you are in the same AWS Region as your SAP HANA database. 4. Choose **Create document**, **Command or session** to create your document. 5. Use a unique and descripti ve name, with no spaces (for example, **SAP HANA-Backup**). 6. Make sure that **Document type** is set to **Command document**.	AWS administrator, SAP HANA administrator

Task	Description	Skills required
	7. Under the **Content** header, there is some sample code. Make sure that you choose the **JSON** code type, and replace the code with the code from the `HDB_Backup_SSM_Document.json` file from the [GitHub repository](https://github.com). 8. Choose **Create document**. 9. Check your document in the **Owned by me** section.	

Schedule backups on a regular frequency

Task	Description	Skills required
Schedule regular backups using Amazon EventBridge.	1. Open the Amazon EventBridge console, choose **Rules**, and choose **Create rule**. 2. On the **Define rule detail** screen, enter a unique name and description for your rule, and use the default event bus. 3. Under **Rule type**, choose **Schedule**, and choose **Next**. 4. On then **Define schedule** screen, choose the appropriate schedule pattern and cron or rate	AWS administrator, SAP HANA administrator

Task	Description	Skills required
	expression based on the required frequency. 5. On the **Select targets** screen, for **Target type**, choose **AWS service**. Under **Select a target**, choose **Systems Manager Run Command**. 6. Choose the document that you created earlier. 7. Under **Target key** and **Target value**, provide the instance ID. You can use tag names and tag values to add multiple instances. 8. Under **Configure automation parameter s**, choose **Constant** for incremental or different ial backups. If you want full backup, choose **No Parameters**. 9. Choose whether to create a new role or to use an existing role. If you use an existing role, make sure that it has the policies required to invoke the target. 10. Keep the default additional settings, and choose **Next**. 11. The **Configure tags** screen is optional. Choose **next**.	

Task	Description	Skills required
	12.On the **Review and create** screen, review the rule settings, and choose **Create**. The rule should be successfully created. You can verify backup success from the S3 bucket path. `s3:/<your_bucket_name>/<target folder>/<SID>/usr/sap/<SID>/SYS/global/hdb/backint/DB_<SID>/` You can also verify backups from the SAP HANA backup catalog.	

Related resources

- AWS Backint Agent for SAP HANA
- Install and configure AWS Backint Agent for SAP HANA

Automatically generate a PynamoDB model and CRUD functions for Amazon DynamoDB by using a Python application

Vijit Vashishtha, Dheeraj Alimchandani, and Dhananjay Karanjkar, Amazon Web Services

Summary

It's common to require entities and create, read, update, and delete (CRUD) operations functions to efficiently perform Amazon DynamoDB database operations. PynamoDB is a Python-based interface that supports Python 3. It also provides features such as support for Amazon DynamoDB transactions, automatic attribute value serialization and deserialization, and compatibility with common Python frameworks, such as Flask and Django. This pattern helps developers working with Python and DynamoDB by providing a library that streamlines the automatic creation of PynamoDB models and CRUD operation functions. While it generates essential CRUD functions for database tables, it can also reverse engineer PynamoDB models and CRUD functions from Amazon DynamoDB tables. This pattern is designed to simplify database operations by using a Python-based application.

The following are the key features of this solution:

- **JSON schema to PynamoDB model** – Automatically generate PynamoDB models in Python by importing a JSON schema file.
- **CRUD function generation** – Automatically generate functions to perform CRUD operations on DynamoDB tables.
- **Reverse engineering from DynamoDB** – Use PynamoDB object-relational mapping (ORM) to reverse engineer PynamoDB models and CRUD functions for existing Amazon DynamoDB tables.

Prerequisites and limitations

Prerequisites

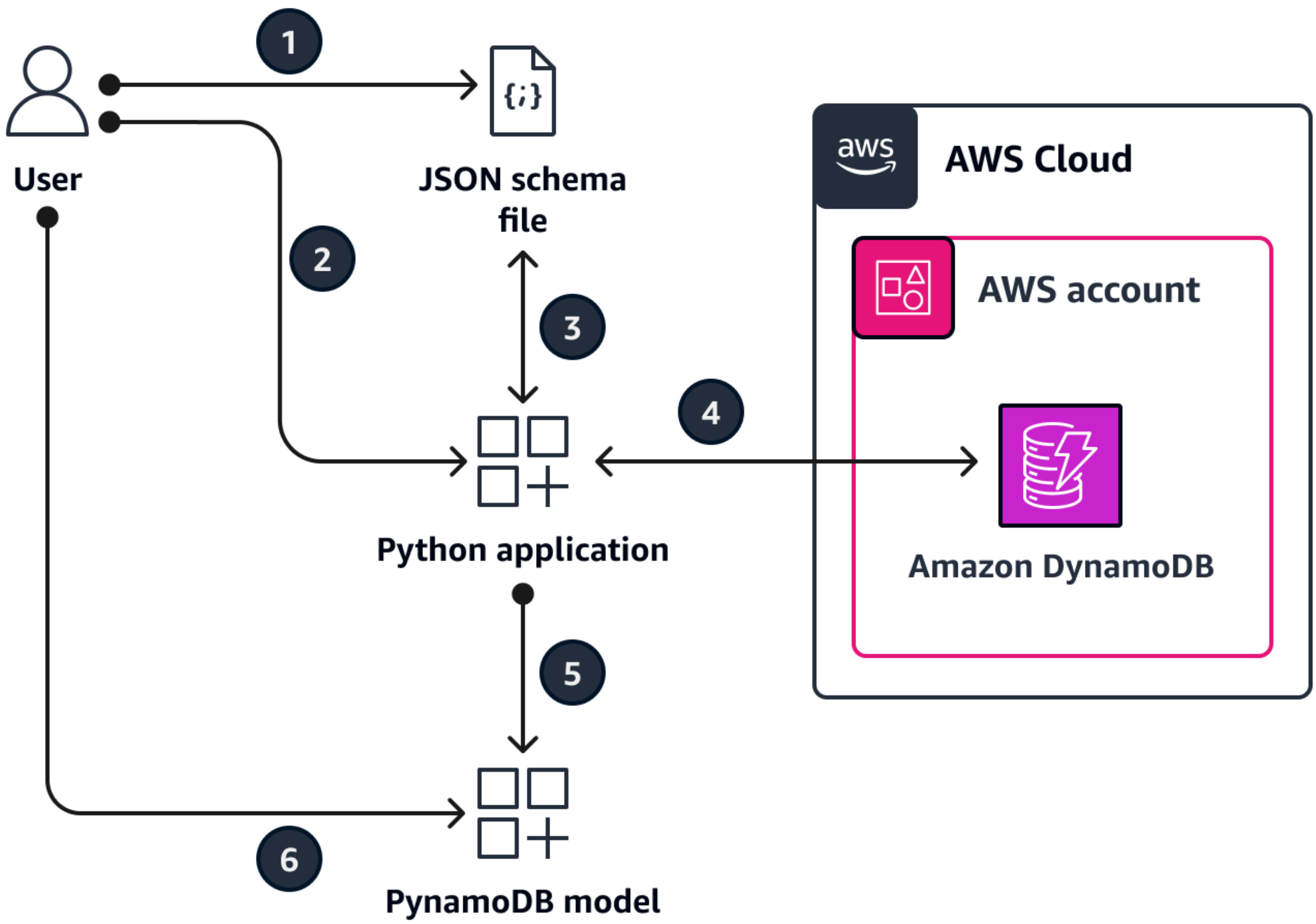
- An active AWS account
- Python version 3.8 or later, downloaded and installed
- Jinja2 version 3.1.2 or later, downloaded and installed
- Amazon DynamoDB tables for which you want to generate ORM
- AWS Command Line Interface (AWS CLI), installed and configured
- PynamoDB version 5.4.1 or later, installed

Architecture

Target technology stack

- JSON script
- Python application
- PynamoDB model
- Amazon DynamoDB database instance

Target architecture



1. You create an input JSON schema file. This JSON schema file represents the attributes of the respective DynamoDB tables that you want to create PynamoDB models from and CRUD functions for. It contains the following three important keys:
   - `name` –The name of the target DynamoDB table.

  - `region` – The AWS Region where the table is hosted
  - `attributes` – The attributes that are part of the target table, such as the partition key (also known as a *hash attribute*), sort key, local secondary indexes, global secondary indexes, and any non-key attributes. This tool expects the input schema to only provide the non-key attributes as the application fetches the key attributes directly from the target table. For an example of how to specify attributes in the JSON schema file, see the Additional information section of this pattern.

2. Run the Python application and provide the JSON schema file as an input.
3. The Python application reads the JSON schema file.
4. The Python application connects to the DynamoDB tables to derive the schema and data types. The application runs the describe_table operation and fetches the key and index attributes for the table.
5. The Python application combines the attributes from the JSON schema file and DynamoDB table. It uses the Jinja template engine to generate a PynamoDB model and corresponding CRUD functions.
6. You access the PynamoDB model to perform CRUD operations on the DynamoDB table.

Tools

AWS services

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.

Other tools

- Jinja is an extensible templating engine that compiles templates into optimized Python code. This pattern uses Jinja to to generate dynamic content by embedding placeholders and logic within templates.
- PynamoDB is a Python-based interface for Amazon DynamoDB.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub [Auto-generate PynamoDB models and CRUD functions]() repository. The repository is divided into two main parts: the controller package and the templates.

Controller package

The controller Python package contains the main application logic that helps generate the PynamoDB model and the CRUD functions. It contains the following:

- `input_json_validator.py` – This Python scripts validates the input JSON schema file and creates the Python objects that contain the list of target DynamoDB tables and the required attributes for each.
- `dynamo_connection.py` – This script establishes a connection to the DynamoDB table and uses the `describe_table` operation to extract the attributes that are necessary to create the PynamoDB model.
- `generate_model.py` – This script contains a Python class `GenerateModel` that creates the PynamoDB model based on the input JSON schema file and the `describe_table` operation.
- `generate_crud.py` – For the DynamoDB tables that are defined in the JSON schema file, this script uses the `GenerateCrud` operation to create the Python classes.

Templates

This Python directory contains the following Jinja templates:

- `model.jinja` – This Jinja template contains the template expression for generating the PynamoDB model script.
- `crud.jinja` – This Jinja template contains the template expression for generating the CRUD functions script.

Epics

Set up the environment

Task	Description	Skills required
Clone the repository.	Enter the following command to clone the [Auto-generate]()	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>PynamoDB models and CRUD functions repository.
<pre><code>git clone https://github.com/aws-samples/amazon-reverse-engineer-dynamodb.git</code></pre></td><td></td></tr>
<tr><td>Set up the Python environment.</td><td>1. Navigate into the top-level directory in the cloned repository.
<pre><code>cd amazon-reverse-engineer-dynamodb</code></pre>2. Enter the following command to install the required libraries and packages.
<pre><code>pip install -r requirements.txt</code></pre></td><td>App developer</td></tr>
</table>

Generate the PynamoDB model and CRUD functions

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Modify the JSON schema file.</td><td>1. Navigate into the top-level directory in the cloned repository.
<pre><code>cd amazon-reverse-engineer-dynamodb</code></pre>2. Open the <code>test.json</code> file in your preferred editor.</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
	You can use this file as a reference to create your own JSON schema file, or you can update the values in this file to match your environment. 3. Modify the name, AWS Region, and attributes values for your target DynamoDB tables. **Note** If you define a table that does not exist in the JSON schema file, this solution does not generate models or CRUD functions for that table. 4. Save and close the `test.json` file. We recommend that you save this file with a new name.	

Task	Description	Skills required
Run the Python application.	Enter the following command to generate the PynamoDB models and CRUD functions, where `<input_schema.json>` is the name of your JSON schema file. `python main.py --file <input_schema.json>`	App developer

Verify the PynamoDB model and CRUD functions

Task	Description	Skills required
Verify the generated PynamoDB model.	1. In the top-level directory of the cloned repository, enter the following command to navigate into the `models` repository. `cd models` 2. By default, this solution names the PynamoDB model file `demo_model.py`. Validate that this file is present.	App developer
Verify the generated CRUD functions.	1. In the top-level directory of the cloned repository, enter the following command to navigate into the `crud` repository.	App developer

Task	Description	Skills required
	`cd crud` 2. By default, this solution names the script `demo_crud.py` . Validate that this file is present. 3. Use the Python classes in the `demo_crud.py` file to perform a CRUD operation on the target DynamoDB table. Confirm that the operation completed successfully.	

Related resources

- Core components of Amazon DynamoDB (DynamoDB documentation)
- Improving data access with secondary indexes (DynamoDB documentation)

Additional information

Sample attributes for the JSON schema file

```
[
{
"name": "test_table",
"region": "ap-south-1",
"attributes": [
{
"name": "id",
"type": "UnicodeAttribute"
},
{
"name": "name",
"type": "UnicodeAttribute"
},
```

```
{
"name": "age",
"type": "NumberAttribute"
}
]
}
]
```

Block public access to Amazon RDS by using Cloud Custodian

abhay kumar and Dwarika Patra, Amazon Web Services

Summary

Many organizations run their workloads and services on multiple cloud vendors. In these hybrid cloud environments, the cloud infrastructure needs strict cloud governance, in addition to the security provided by the individual cloud providers. A cloud database such as Amazon Relational Database Service (Amazon RDS) is one important service that must be monitored for any access and permission vulnerabilities. Although you can restrict access to the Amazon RDS database by configuring a security group, you can add a second layer of protection to prohibit actions such as public access. Blocking public access helps you with General Data Protection Regulation (GDPR), Health Insurance Portability and Accountability Act (HIPAA), National Institute of Standards and Technology (NIST), and Payment Card Industry Data Security Standard (PCI DSS) compliance.

Cloud Custodian is an open-source rules engine that you can use to enforce access restrictions for Amazon Web Services (AWS) resources such as Amazon RDS. With Cloud Custodian, you can set rules that validate the environment against defined security and compliance standards. You can use Cloud Custodian to manage your cloud environments by helping to ensure compliance with security policies, tag policies, and garbage collection of unused resources and cost management. With Cloud Custodian, you can use a single interface for implementing governance in a hybrid cloud environment. For example, you could use the Cloud Custodian interface to interact with AWS and Microsoft Azure, reducing the effort of working with mechanisms such as AWS Config, AWS security groups, and Azure policies.

This pattern provides instructions for using Cloud Custodian on AWS to enforce restriction of public accessibility on Amazon RDS instances.

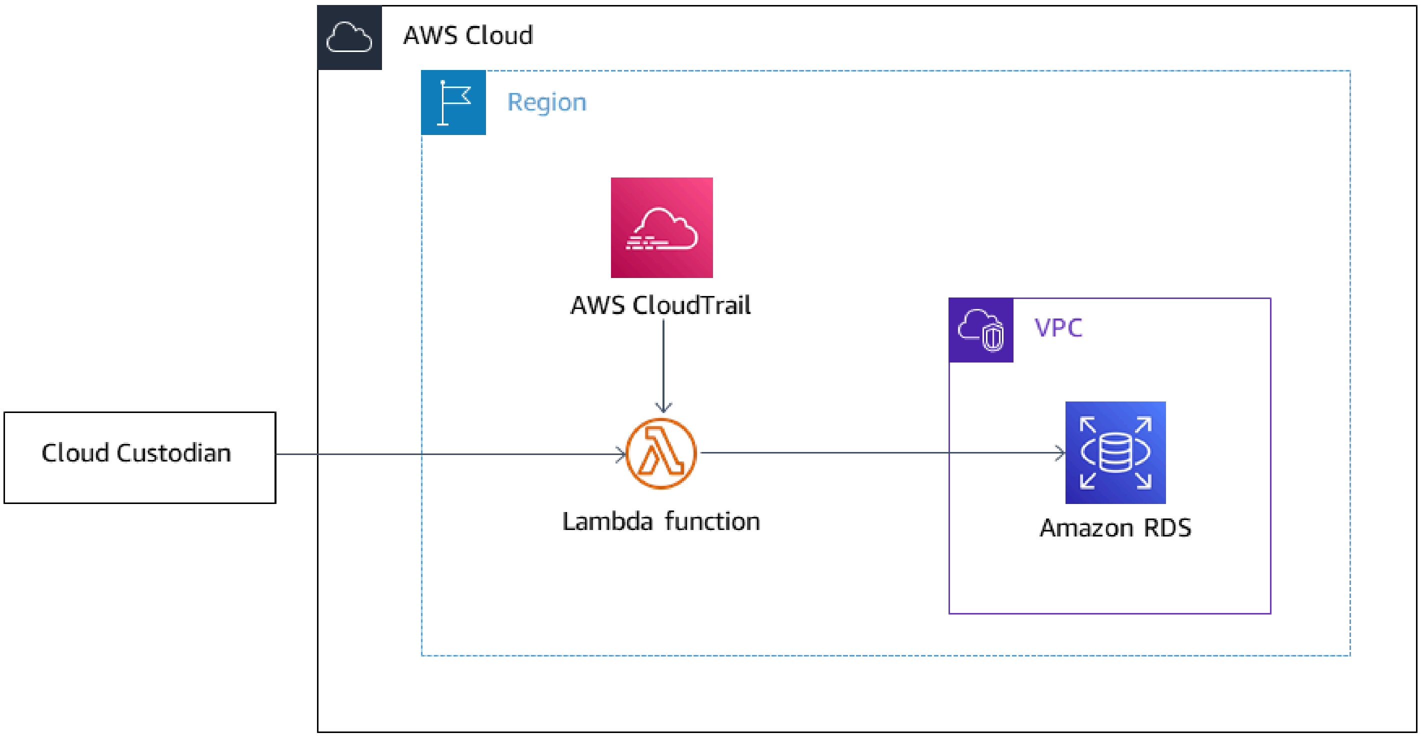
Prerequisites and limitations

Prerequisites

- An active AWS account
- [A key pair](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html)
- AWS Lambda installed

Architecture

The following diagram shows Cloud Custodian deploying the policy to AWS Lambda, AWS CloudTrail initiating the `CreateDBInstance` event, and the Lambda function setting `PubliclyAccessible` to false on Amazon RDS.



Tools

AWS services

- [AWS CloudTrail](https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html) helps you audit the governance, compliance, and operational risk of your AWS account.
- [AWS Command Line Interface (AWS CLI)](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html) is an open-source tool that helps you interact with AWS services through commands in your command line shell.
- [AWS Identity and Access Management (IAM)](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Lambda](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [Amazon Relational Database Service (Amazon RDS)](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html) helps you set up, operate, and scale a relational database in the AWS Cloud.

Other tools

- Cloud Custodian unifies the tools and scripts that many organizations use to manage their public cloud accounts into one open source tool. It uses a stateless rules engine for policy definition and enforcement, with metrics, structured outputs, and detailed reporting for cloud infrastructure. It integrates tightly with a serverless runtime to provide real-time remediation and response with low operational overhead.

Epics

Set up the AWS CLI

Task	Description	Skills required
Install the AWS CLI.	To install the AWS CLI, follow the instructions in the AWS documentation.	AWS administrator

Set up AWS credentials.	Configure the settings that the AWS CLI uses to interact with AWS, including the AWS Region and the output format that you want to use.
	
	`$>aws configure`
	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: <your_secret_access_key>`
	`Default region name [None]:`
	`Default output format [None]:`
	
	For more information, see the AWS documentation.
		AWS administrator

Task	Description	Skills required
Create an IAM role.	To create an IAM role with the Lambda execution role, run the following command. `aws iam create-role --role-name lambda-ex --assume-role-policy-document '{"Version": "2012-10-17","Statement": [{ "Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action": "sts:AssumeRole"}]}`	AWS DevOps

Set up Cloud Custodian

Task	Description	Skills required
Install Cloud Custodian.	To install Cloud Custodian for your operating system and environment, follow the instructions in the Cloud Custodian documentation.	DevOps engineer
Check the Cloud Custodian schema.	To see the complete list of Amazon RDS resources against which you can run policies, use the following command. `custodian schema aws.rds`	DevOps engineer
Create the Cloud Custodian policy.	Save the code that's under *Cloud Custodian policy file*	DevOps engineer

Task	Description	Skills required
	in the Additional information section using a YAML extension.	
Define Cloud Custodian actions to change the publicly accessible flag.	1. Locate the custodian code (for example, `/Users/abcd/custodian/lib/python3.9/site-packages/c7n/resources/rds.py`). 2. Locate the `RDSSetPublicAvailability` class in `rds.py`, and modify this class by using the code that's under *c7n resources rds.py file* in the Additional information section.	DevOps engineer
Perform a dry run.	(Optional) To check which resources are identified by the policy without running any actions on the resources, use the following command. `custodian run -dryrun <policy_name>.yaml -s <output_directory>`	DevOps engineer

Deploy the policy

Task	Description	Skills required
Deploy the policy by using Lambda.	To create the Lambda function that will run the	DevOps engineer

Task	Description	Skills required
	policy, use the following command. ```custodian run -s policy.yaml``` This policy will then be initiated by the AWS CloudTrail `CreateDBInstance` event. As a result, AWS Lambda will set the publicly accessible flag to `false` for instances that match the criteria.	

Related resources

- AWS Lambda website
- Amazon RDS website
- Cloud Custodian documentation

Additional information

Cloud Custodian policy YAML file

```
policies:
  - name: "block-public-access"
    resource: rds
    description: |
      This Enforcement blocks public access for RDS instances.
    mode:
      type: cloudtrail
      events:
        - event: CreateDBInstance # Create RDS instance cloudtrail event
          source: rds.amazonaws.com
```

```
          ids: requestParameters.dBInstanceIdentifier
      role: arn:aws:iam::1234567890:role/Custodian-compliance-role
    filters:
      - type: event
        key: 'detail.requestParameters.publiclyAccessible'
        value: true
    actions:
      - type: set-public-access
        state: false
```

c7n resources rds.py file

```
@actions.register('set-public-access')
 class RDSSetPublicAvailability(BaseAction):

     schema = type_schema(
         "set-public-access",
         state={'type': 'boolean'})
     permissions = ('rds:ModifyDBInstance',)

     def set_accessibility(self, r):
         client = local_session(self.manager.session_factory).client('rds')
         waiter = client.get_waiter('db_instance_available')
         waiter.wait(DBInstanceIdentifier=r['DBInstanceIdentifier'])
         client.modify_db_instance(
             DBInstanceIdentifier=r['DBInstanceIdentifier'],
             PubliclyAccessible=self.data.get('state', False))


     def process(self, rds):
         with self.executor_factory(max_workers=2) as w:
             futures = {w.submit(self.set_accessibility, r): r for r in rds}
             for f in as_completed(futures):
                 if f.exception():
                     self.log.error(
                         "Exception setting public access on %s  \n %s",
                         futures[f]['DBInstanceIdentifier'], f.exception())
         return rds
```

Security Hub integration

Cloud Custodian can be integrated with AWS Security Hub to send security findings and attempt remediation actions. For more information, see Announcing Cloud Custodian Integration with AWS Security Hub.

Configure cross-account access to Amazon DynamoDB

Shashi Dalmia, Imhoertha Ojior, and Esteban Serna Parra, Amazon Web Services

Summary

This pattern explains the steps for configuring cross-account access to Amazon DynamoDB by using resource-based policies. For workloads that use DynamoDB, it's becoming more common to use workload isolation strategies to minimize security threats and to meet compliance requirements. Implementing workload isolation strategies often requires cross-account and cross-Region access to DynamoDB resources by using AWS Identity and Access Management (IAM) identity-based policies. This involves setting IAM permissions and establishing a trust relationship between the AWS accounts.

Resource-based policies for DynamoDB greatly simplify the security posture for cross-account workloads. This pattern provides steps and sample code to demonstrate how you can configure AWS Lambda functions in one AWS account to write data to a DynamoDB database table in a different account.

Prerequisites and limitations

Prerequisites

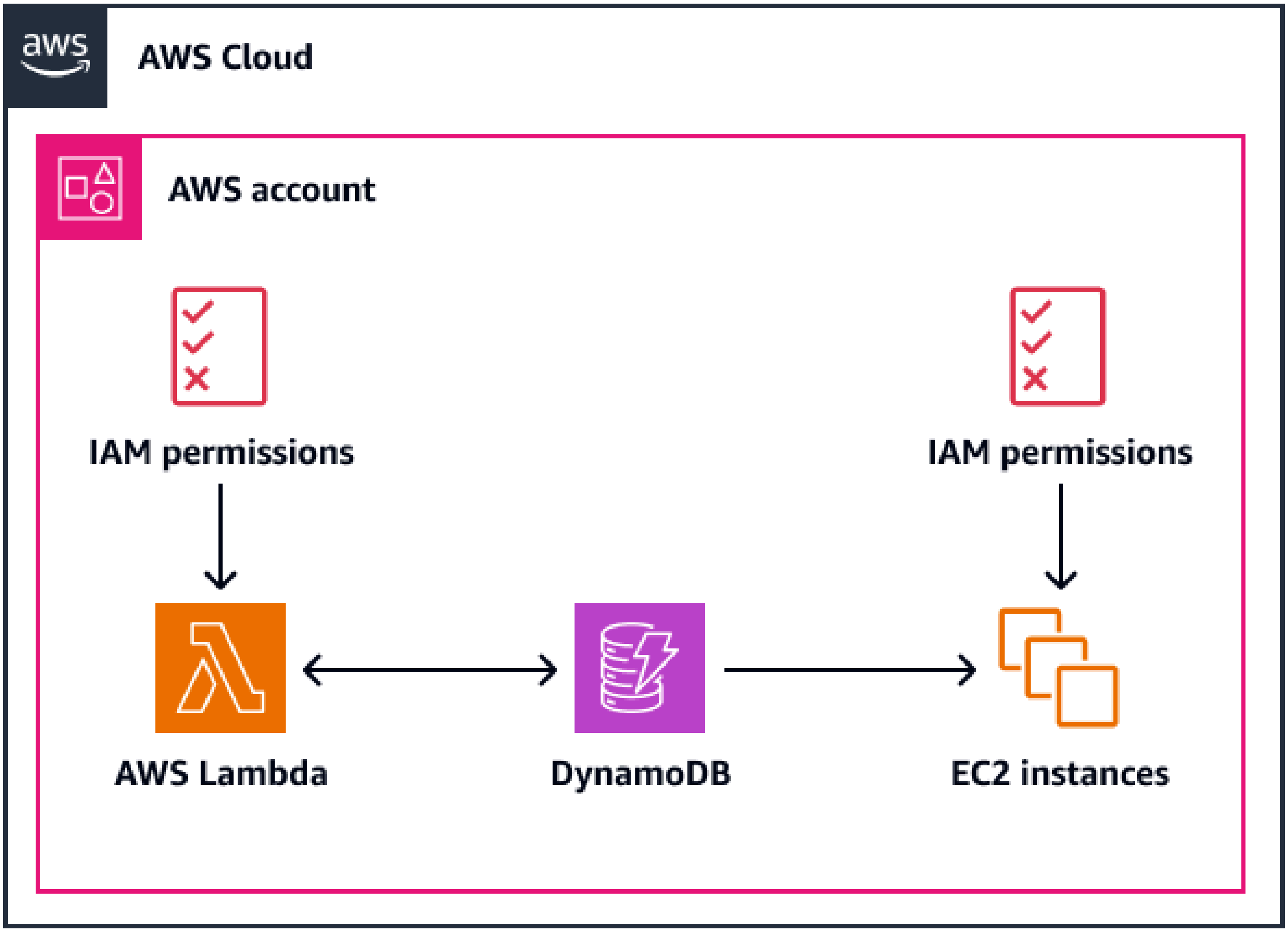
- Two active AWS accounts. This pattern refers to these accounts as *Account A* and *Account B*.
- AWS Command Line Interface (AWS CLI) installed and configured to access Account A, to create the DynamoDB table. The other steps in this pattern provide instructions for using the IAM, DynamoDB, and Lambda consoles. If you're planning to use AWS CLI instead, configure it to access both accounts.

Limitations

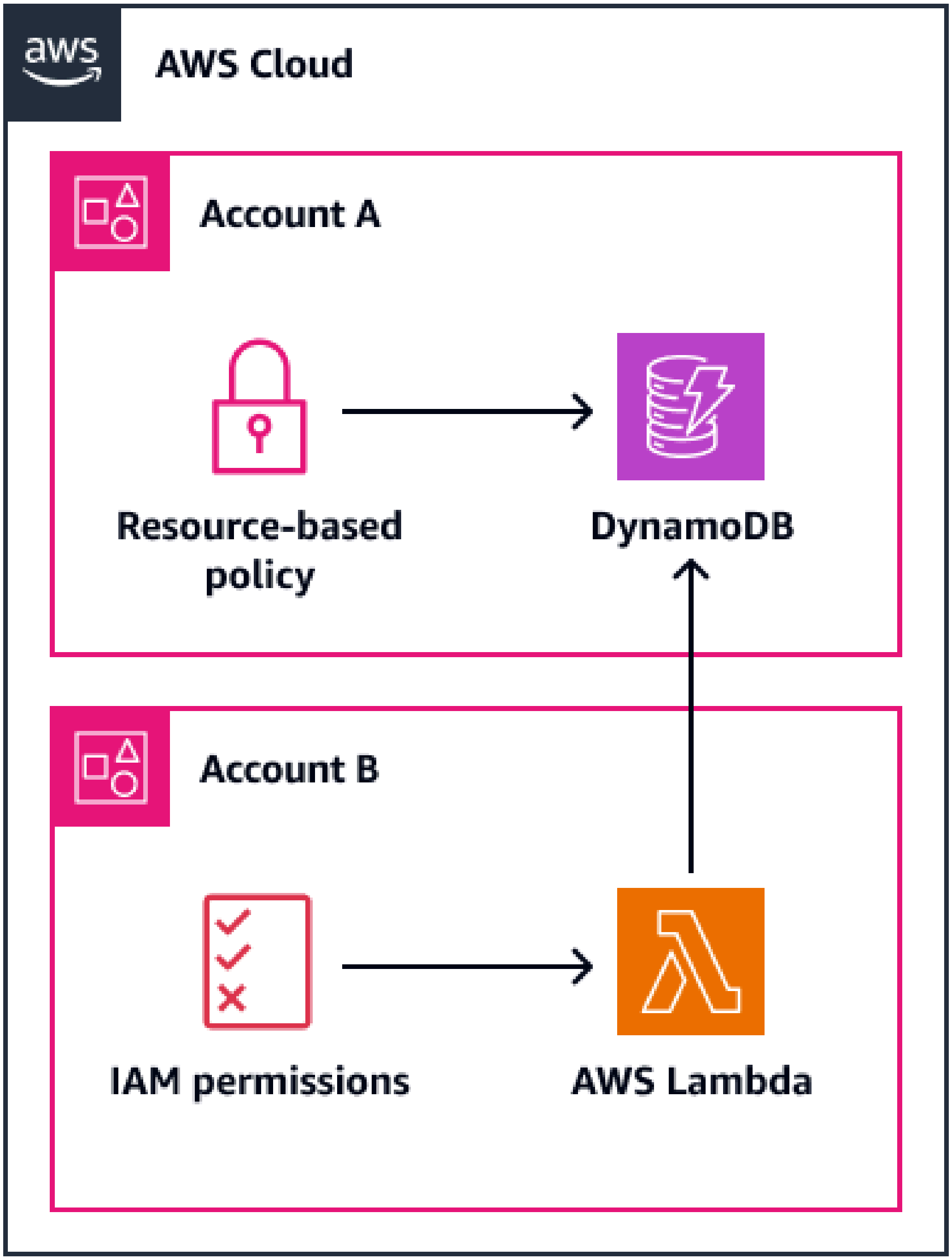
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

The following diagram shows a single-account architecture. AWS Lambda, Amazon Elastic Compute Cloud (Amazon EC2), and DynamoDB are all in the same account. In this scenario, Lambda functions and Amazon EC2 instances can access DynamoDB. To grant access to the DynamoDB table, you can create an identity-based policy in IAM, or you can create a resource-based policy in DynamoDB.



The following diagram shows a multi-account architecture. If resources in one AWS account require access to a DynamoDB table in a different account, you need to set up a resource-based policy in DynamoDB to grant the required access. For example, in the following diagram, access to the DynamoDB table in Account A is granted to a Lambda function in Account B by using a resource-based policy.



This pattern describes cross-account access between Lambda and DynamoDB. You can use similar steps for other AWS services if the appropriate permissions are configured on both accounts. For example, if you want to provide a Lambda function access to an Amazon Simple Storage Service (Amazon S3) bucket in Account A, you can create a [resource-based policy] in Amazon S3 and add the permissions to the [Lambda execution role] in Account B.

Tools

AWS services

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Code

This pattern includes sample code in the Additional information section to show how you can configure a Lambda function in Account B to write to the DynamoDB table in Account A. The code is provided only for illustration and testing purposes. If you're implementing this pattern in a production environment, use the code as a reference, and customize it for your own environment.

Best practices

- Follow the best practices for resource-based policies in the DynamoDB documentation.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Create an IAM policy and role for the Lambda function in Account B

Task	Description	Skills required
Create a policy in Account B.	This IAM policy allows the PutItem action for a DynamoDB table in Account A.	General AWS

Task	Description	Skills required

	1. Sign in to Account B in the AWS Management Console.
	2. Open the [IAM console](https://console.aws.amazon.com/iam/).
	3. In the navigation pane, choose **Policies**, and then choose **Create policy**.
	4. On the **Specify permissions** page, for the policy editor, select **JSON**.
	5. Enter the following policy.
	<pre>{
	  "Version": "2012-10-17",
	  "Statement": [
	    {
	      "Sid": "Statement1",
	      "Effect": "Allow",
	      "Action": "dynamodb:PutItem",
	      "Resource": "arn:aws:dynamodb:<Region>:<Account-A-ID>:table/Table-Account-A"
	    }
	  ]
	}</pre>6. Replace `<Region>` and `<Account-A-ID>` with your values, and then choose **Next**.
	7. For **Policy name**, enter a unique name for your	

Task	Description	Skills required
	policy, such as `DynamoDB-PutItem-Policy` . 8. (Optional) Add a policy description. 9. Choose **Create policy**.	

Task	Description	Skills required
Create a role in Account B.	The Lambda function in Account B uses this IAM role to access the DynamoDB table in Account A. 1. Open the [IAM console](https://console.aws.amazon.com/iam/). 2. In the navigation pane, choose **Roles**, and then choose **Create role**. 3. For **Select trusted entity**, choose **AWS service**. 4. In the **Use case** section, choose **Lambda**. 5. Choose **Next: Permissions**. 6. In the **Filter policies** box, enter **DynamoDB**. 7. In the list of DynamoDB policies, choose `DynamoDB-PutItem-Policy` . 8. Clear the **Filter policies** box, and then enter **Lambda**. 9. In the list of Lambda policies, choose **AWSLambdaExecute**. 10. Choose **Next: Name, review, and create**. 11. For **Role name**, enter a unique name for your role, such as `DynamoDB-PutItemAccess` .	General AWS

Task	Description	Skills required
	12.(Optional) Add a role description. 13.(Optional) Add metadata to the role by attaching tags as key-value pairs. 14.Choose **Create role**. For more information about creating roles, see the [IAM documentation]().	
Note the role ARN.	1. Open the [IAM console](). 2. In the navigation pane, choose **Roles**. 3. In the search box, enter `DynamoDB-PutItemAccess` , and then choose the role. 4. On the summary page for the role, copy the Amazon Resource Name (ARN). You use the ARN when you set up the Lambda function.	General AWS

Create a DynamoDB table in Account A

Task	Description	Skills required
Create a DynamoDB table.	Use the following AWS CLI command to create a DynamoDB table. `aws dynamodb create-table \`	General AWS

Task	Description	Skills required

	<pre> --table-name Table-
	Account-A \\
	    --attribute-defini
	tions \\
	      Attribute
	Name=category,Attr
	ibuteType=S \\
	      Attribute
	Name=item,Attribut
	eType=S \\
	    --key-schema \\
	      Attribute
	Name=category,KeyT
	ype=HASH \\
	      Attribute
	Name=item,KeyType=
	RANGE \\
	    --provisioned-thro
	ughput \\
	      ReadCapac
	ityUnits=5,WriteCa
	pacityUnits=5 \\
	    --resource-policy \\
	      '{
	          "Version":
	 "2012-10-17",
	          "Statement":
	 [
	            {
	
	               "Sid":
	 "Statement1",
	               "Effect":
	 "Allow",
	               "Principa
	l": {
	                  "AWS":
	 "arn:aws:iam::<Acc
	ount-B-ID>:role/<R
	ole-Name>"
	               },</pre>	

Task	Description	Skills required

	``
	                "Action":
	 "dynamodb:PutItem",
	                "Resource
	": "arn:aws:dynamodb:
	<Region>:<Account-
	A-ID>:table/Table-
	Account-A"
	            }
	
	         ]
	      }'
	``
	
	Replace the following in this code sample:
	
	• `<Account-B-ID>` is the ID of Account B.
	• `<Role-Name>` is the name of the IAM role that you created, such as `DynamoDB-PutItemAccess`.
	• `<Region>` is the AWS Region where you are creating the DynamoDB table.
	• `<Account-A-ID>` is the ID of Account A.
	
	Note
	
	You specify the resource-based policy configuration in the `create-table` statement by using	

Task	Description	Skills required
	the `--resource-policy` flag. This policy refers to the ARN for the DynamoDB table in Account A. For more information about creating tables, see the [DynamoDB documentation]().	

Create a Lambda function in Account B

Task	Description	Skills required
Create a Lambda function to write data to DynamoDB.	1. Sign in to Account B in the AWS Management Console. 2. Open the [Lambda console](). 3. In the navigation pane, choose **Functions**, and then choose **Create function**. 4. For **Name**, enter `lambda_write_function` . 5. For **Runtime**, choose **Python 3.8** or later. 6. Under **Change default execution role**, choose **Use an existing role**. 7. For **Existing role**, choose the IAM role that	General AWS

Task	Description	Skills required
	you created, such as `DynamoDB-PutItemAccess` . 8. Choose **Create function**. 9. In the **Code** tab, paste the sample code that is provided in the [Additional information](#) section of this pattern. Replace the following in this code sample: • `<Account-A-ID>` is the ID of Account A. • `<Region>` is the AWS Region where you created the DynamoDB table. 10. Choose **Deploy**. 11. Choose **Test**. This prompts you to configure a test event. Create a new event with your preferred name, such as `MyTestEventForWrite` , and then save the configuration. 12. Choose **Test** again. This runs the Lambda function with the event name you provided. 13. Check the output from the function. It should indicate that the function accessed the DynamoDB table in	

Task	Description	Skills required
	Account A and was able to write data to it. For more information about creating Lambda functions, see the [Lambda documenta tion]().	

Clean up

Task	Description	Skills required
Delete resources.	To avoid incurring costs associated with the resources created in this pattern, do the following to delete these resources: 1. In Account B, delete the Lambda function you created to connect to DynamoDB. For instructi ons, see the [Lambda documentation](). 2. In Account A, delete the DynamoDB table you created. For instructi ons, see the [DynamoDB documentation](). 3. For security best practices , delete the IAM policy (`DynamoDB-PutItem-Policy`) when it is no longer needed. For more	General AWS

Task	Description	Skills required
	information, see the IAM documentation. 4. For security best practices , delete the IAM role (`DynamoDB-PutItemAccess`) when it is no longer needed. For more information, see the IAM documentation.	

Troubleshooting

Issue	Solution
When creating the Lambda function, you receive a `ResourceNotFoundException` error.	Confirm that you have correctly entered the AWS Region and ID of Account A. These are part of the ARN for the DynamoDB table.

Related resources

- Getting started with DynamoDB (DynamoDB documentation)
- Getting started with Lambda (Lambda documentation)
- Using resource-based policies for DynamoDB (DynamoDB documentation)
- Creating IAM policies (IAM documentation)
- Cross-account policy evaluation logic (IAM documentation)
- IAM JSON policy elements reference (IAM documentation)

Additional information

Sample code

```
import boto3
```

```
from datetime import datetime

dynamodb_client = boto3.client('dynamodb')

def lambda_handler(event, context):
     now = datetime.now().isoformat()
     data = dynamodb_client.put_item(TableName='arn:aws:dynamodb:<Region>:<Account-
A-ID>:table/Table-Account-A', Item={"category": {"S": "Fruit"},"item": {"S":
 "Apple"},"time": {"S": now}})
     return data
```

> **Note**
>
> When the DynamoDB client is instantiated, the ARN of the DynamoDB table is provided instead of the table name. This is required so that the Lambda function connects to the correct DynamoDB table when it runs.

Configure read-only routing in an Always On availability group in SQL Server on AWS

Subhani Shaik, Amazon Web Services

Summary

This pattern covers how to use the standby secondary replica in SQL Server Always On by offloading the read-only workloads from the primary replica to the secondary replica.

Database mirroring has one-to-one mapping. You can't read the secondary database directly, so you must create snapshots. The Always On availability group feature was introduced in Microsoft SQL Server 2012. In later versions, major functionalities have been introduced, including read-only routing. In Always On availability groups, you can read the data directly from the secondary replica by changing the replica mode to read-only.

The Always On availability groups solution supports high availability (HA), disaster recovery (DR), and an alternative to database mirroring. Always On availability groups work at the database level and maximize the availability of a set of user databases.

SQL Server uses the read-only routing mechanism to redirect the incoming read-only connections to the secondary read replica. To achieve this, you should add the following parameters and values in the connection string:

- `ApplicationIntent=ReadOnly`
- `Initial Catalog=<database name>`

Prerequisites and limitations

Prerequisites

- An active AWS account with a virtual private cloud (VPC), two Availability Zones, private subnets, and a security group
- Two Amazon Elastic Compute Cloud (Amazon EC2) machines with [SQL Server 2019 Enterprise Edition Amazon Machine Image] with [Windows Server Failover Clustering (WSFC)] configured at the instance level and an Always On availability group configured at the SQL Server level between the primary node (`WSFCNODE1`) and the secondary node (`WSFCNODE2`), which are part of the AWS Directory Service for Microsoft Active Directory directory named `tagechtalk.com`

- One or more nodes configured to accept `read-only` in the secondary replica
- A listener named `SQLAG1` for the Always On availability group
- SQL Server Database Engine running with the same service account on two nodes
- SQL Server Management Studio (SSMS)
- A test database named `test`

Product Versions

- SQL Server 2014 and later

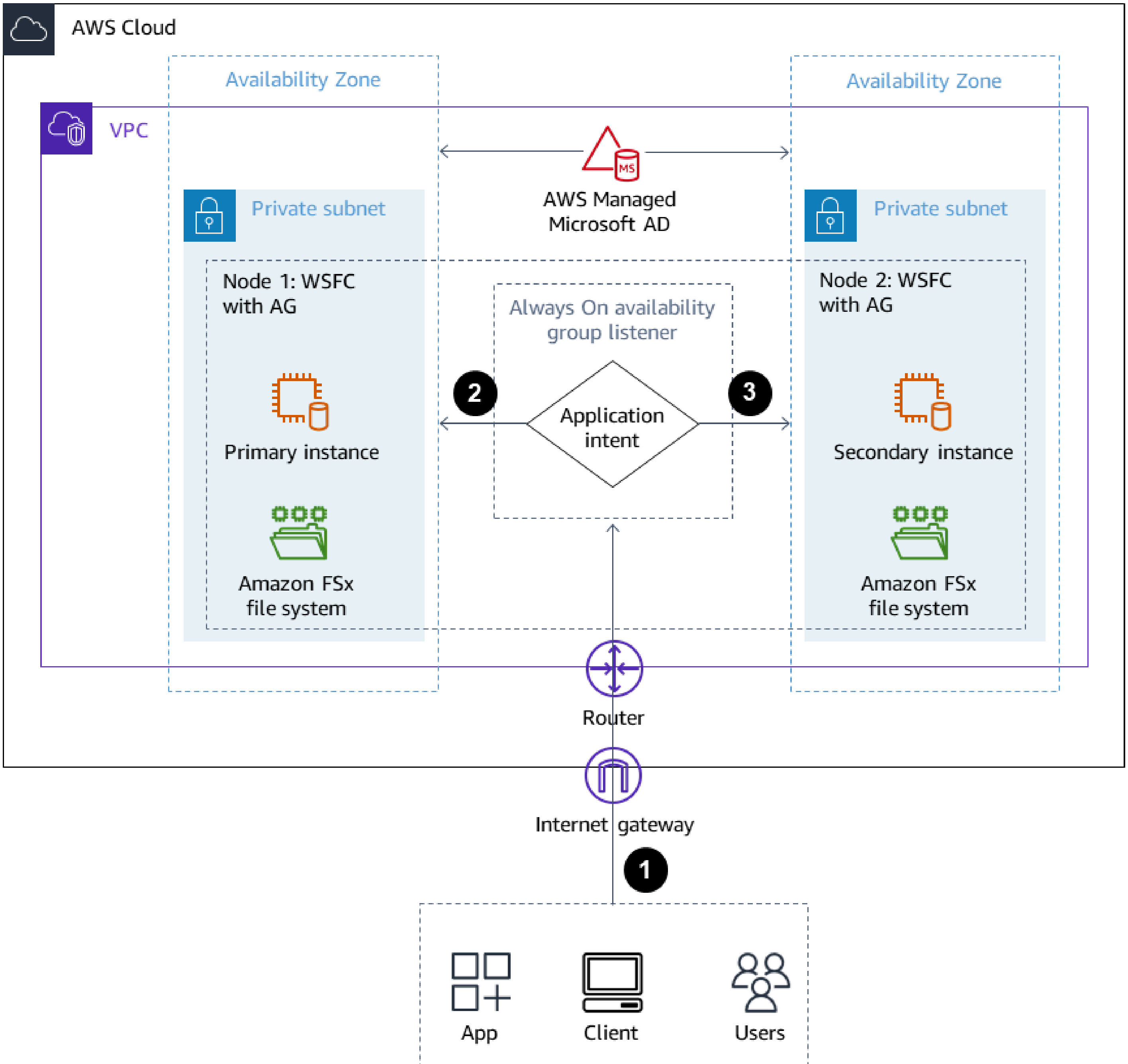
Architecture

Target technology stack

- Amazon EC2
- AWS Managed Microsoft AD
- Amazon FSx

Target architecture

The following diagram shows how the Always On availability group (AG) listener redirects queries that contain the `ApplicationIntent` parameter in the connection to the appropriate secondary node.



1. A request is sent to the Always On availability group listener.
2. If the connection string does not have the `ApplicationIntent` parameter, the request is sent to the primary instance.
3. If the connection string contains `ApplicationIntent=ReadOnly`, the request is sent to the secondary instance with read-only routing configuration , which is WSFC with an Always On availability group.

Tools

AWS services

- AWS Directory Service for Microsoft Active Directory enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon FSx provides file systems that support industry-standard connectivity protocols and offer high availability and replication across AWS Regions.

Other services

- SQL Server Management Studio (SSMS) is a tool for connecting, managing, and administering the SQL Server instances.
- sqlcmd is a command-line utility.

Best practices

For more information about Always On availability groups, see the SQL Server documentation.

Epics

Set up read-only routing

Task	Description	Skills required
Update the replicas to read-only.	To update both the primary and the secondary replica to read-only, connect to the primary replica from SSMS, and run the *Step 1* code from the *Additional information* section.	DBA
Create the routing URL.	To create routing URL for both replicas, run the *Step 2* code from the *Additional*	DBA

Task	Description	Skills required
	information section. In this code, `tagechtalk.com` is the name of the AWS Managed Microsoft AD directory.	
Create the routing list.	To create the routing list for both replicas, run the Step 3 code from the *Additional information* section.	DBA
Validate the routing list.	Connect to the primary instance from SQL Server Management Studio, and run the *Step 4* code from the *Additional information* section to validate the routing list.	DBA

Test the read-only routing

Task	Description	Skills required
Connect by using the ApplicationIntent parameter.	1. From SSMS, connect to the Always On availability group listener name with `ApplicationIntent= ReadOnly;Initial Catalog=test` . 2. The connection is establish ed with the secondary replica. To test this, run the following command to show the connected server name.	DBA

Task	Description	Skills required
	`SELECT SERVERPROPERTY('ComputernamePhysicalNetBios')` The output will show the current secondary replica name (`WSFCNODE2`).	

Task	Description	Skills required
Perform a failover.	1. From SSMS, connect to the Always On availability group listener name. 2. Verify that the primary and secondary database are in sync, with no data loss. 3. Perform a failover so that the current primary replica becomes the secondary replica, and the secondary replica becomes the primary replica. 4. From SSMS, connect to the Always On availability group listener name with `ApplicationIntent=ReadOnly;Initial Catalog=test` . 5. The connection is established with the secondary replica. To test this, show the connected server name by running the following command. `SELECT SERVERPROPERTY('ComputernamePhysicalNetBios')` It will display the current secondary replica name (`WSFCNODE1`).	DBA

Connect by using the sqlcmd command-line utility

Task	Description	Skills required
Connect by using sqlcmd.	To connect from sqlcmd, run the *Step 5* code from the *Additional information* section at the command prompt. After you are connected, run the following command to show the connected server name. `SELECT SERVERPROPERTY('ComputernamePhysicalNetBios') .` The output will display the current secondary replica name (`WSFCNODE1`).	DBA

Troubleshooting

Issue	Solution
Creating the listener fails with the message 'The WSFC cluster could not bring the Network Name resource online'.	For information, see the Microsoft blog post Create Listener Fails with Message 'The WSFC cluster could not bring the Network Name resource online'.
Potential issues, including other listener issues or network access issues.	See Troubleshoot Always On Availability Groups Configuration (SQL Server) in the Microsoft documentation.

Related resources

- Configure read-only routing for an Always On availability group
- Troubleshoot Always On Availability Groups Configuration (SQL Server)

Additional information

Step 1. Update the replicas to read-only

```
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE1' WITH (SECONDARY_ROLE
 (ALLOW_CONNECTIONS = READ_ONLY))
GO
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE2' WITH (SECONDARY_ROLE
 (ALLOW_CONNECTIONS = READ_ONLY))
GO
```

Step 2. Create the routing URL

```
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE1' WITH (SECONDARY_ROLE
 (READ_ONLY_ROUTING_URL = N'TCP://WSFCNode1.tagechtalk.com:1433'))
GO
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE2' WITH (SECONDARY_ROLE
 (READ_ONLY_ROUTING_URL = N'TCP://WSFCNode2.tagechtalk.com:1433'))
GO
```

Step 3. Create the routing list

```
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE1' WITH
 (PRIMARY_ROLE(READ_ONLY_ROUTING_LIST=('WSFCNODE2','WSFCNODE1')));
GO
ALTER AVAILABILITY GROUP [SQLAG1] MODIFY REPLICA ON N'WSFCNODE2' WITH (PRIMARY_ROLE
 (READ_ONLY_ROUTING_LIST=('WSFCNODE1','WSFCNODE2')));
GO
```

Step 4. Validate the routing list

```
SELECT AGSrc.replica_server_name AS PrimaryReplica, AGRepl.replica_server_name AS
 ReadOnlyReplica, AGRepl.read_only_routing_url AS RoutingURL , AGRL.routing_priority
 AS RoutingPriority FROM sys.availability_read_only_routing_lists AGRL INNER JOIN
 sys.availability_replicas AGSrc ON AGRL.replica_id = AGSrc.replica_id INNER JOIN
```

```
sys.availability_replicas AGRepl ON AGRL.read_only_replica_id = AGRepl.replica_id
INNER JOIN sys.availability_groups AV ON AV.group_id = AGSrc.group_id ORDER BY
PrimaryReplica
```

Step 5. SQL Command Utility

```
sqlcmd -S SQLAG1,1433 -E -d test -K ReadOnly
```

Connect by using an SSH tunnel in pgAdmin

Jeevan Shetty and Bhanu Ganesh Gudivada, Amazon Web Services

Summary

For security reasons, it's always good to place databases in a private subnet. Queries against the database can be run by connecting through an Amazon Elastic Compute Cloud (Amazon EC2) bastion host in a public subnet on the Amazon Web Services (AWS) Cloud. This requires installing software, such as pgAdmin or DBeaver, which are commonly used by developers or database administrators, on the Amazon EC2 host.

Running pgAdmin on a Linux server and accessing it through a web browser requires the installation of additional dependencies, permissions setup, and configuration.

As an alternate solution, developers or database administrators can connect to a PostgreSQL database by using pgAdmin to enable an SSH tunnel from their local system. In this approach, pgAdmin uses the Amazon EC2 host in the public subnet as an intermediary host before connecting to the database. The diagram in the *Architecture* section shows the setup.

> **Note**
>
> Ensure that the security group attached to the PostgreSQL database allows connection on port 5432 from the Amazon EC2 host.

Prerequisites and limitations

Prerequisites

- An existing AWS account
- A virtual private cloud (VPC) with a public subnet and a private subnet
- An EC2 instance with a security group attached
- An Amazon Aurora PostgreSQL-Compatible Edition database with a security group attached
- A Secure Shell (SSH) key pair for setting up the tunnel

Product versions

- pgAdmin version 6.2+

- Amazon Aurora PostgreSQL-Compatible Edition version 12.7+

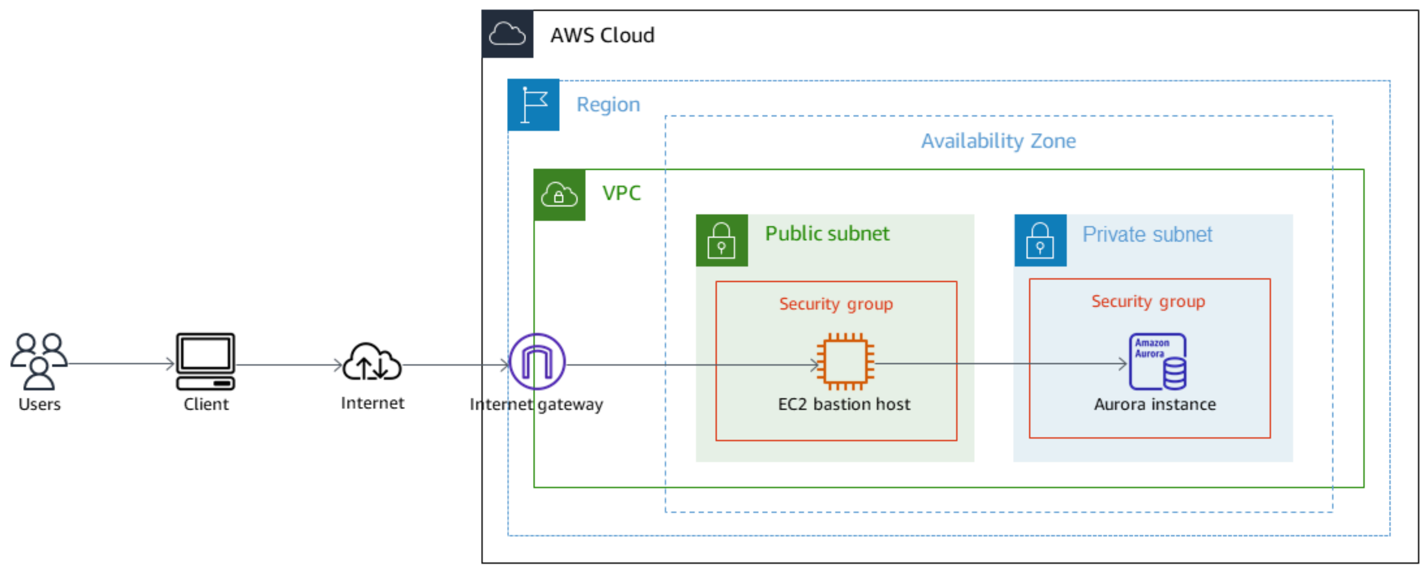
Architecture

Target technology stack

- Amazon EC2
- Amazon Aurora PostgreSQL-Compatible

Target architecture

The following diagram shows using pgAdmin with an SSH tunnel to connect through an internet gateway to the EC2 instance, which connects to the database.



Tools

AWS services

- [Amazon Aurora PostgreSQL-Compatible Edition](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.AuroraPostgreSQL.html) is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- [Amazon Elastic Compute Cloud (Amazon EC2)](https://docs.aws.amazon.com/ec2/) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.

Other services

- pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.

Epics

Create the connection

Task	Description	Skills required
Create a server.	In pgAdmin, choose **Create**, and then choose **Server**. For additional help with setting up pgAdmin to register a server, configure a connectio n, and connect through SSH tunneling by using the Server Dialog, see the links in the *Related resources* section.	DBA
Provide a name for the server.	On the **General** tab, enter a name.	DBA
Enter the database details.	On the **Connection** tab, enter values for the following: • **Host name/address** • **Port** • **Maintenance database** • **Username** • **Password**	DBA
Enter the Amazon EC2 server details.	On the **SSH Tunnel** tab, provide the details of the Amazon EC2 instance that is in the public subnet.	DBA

Task	Description	Skills required
	• Set **Use SSH tunneling** to **Yes** to specify that pgAdmin should use an SSH tunnel when connecting to the specified server. • In the **Tunnel host** field, specify the name or IP address of the SSH host (for example, 10.x.x.x). • In the **Tunnel port** field, specify the port of the SSH host (for example, 22). • In the **Username** field, specify the name of a user with login privileges for the SSH host (for example, ec2-user). • Specify the type of authentication as **Identity file** so that pgAdmin will use a private key file when connecting. • Include the location of the Privacy Enhanced Mail (PEM) file in the **Identity file** field. The .pem file is the Amazon EC2 key pair.	
Save and connect.	Choose **Save** to complete the setup and connect to the Aurora PostgreSQL-Compatible database by using the SSH tunnel.	DBA

Related resources

- Server Dialog
- Connect to Server

Convert JSON Oracle queries into PostgreSQL database SQL

Pinesh Singal and Lokesh Gurram, Amazon Web Services

Summary

This migration process for moving from on-premises to the Amazon Web Services (AWS) Cloud uses the AWS Schema Conversion Tool (AWS SCT) to convert the code from an Oracle database into a PostgreSQL database. Most of the code is automatically converted by AWS SCT. However, JSON-related Oracle queries are not automatically converted.

Starting from Oracle 12.2 version, Oracle Database supports various JSON functions that help in converting JSON-based data into ROW-based data. However, AWS SCT doesn't automatically convert JSON-based data into language that is supported by PostgreSQL.

This migration pattern primarily focuses on manually converting the JSON-related Oracle queries with functions such as `JSON_OBJECT`, `JSON_ARRAYAGG`, and `JSON_TABLE` from an Oracle database to a PostgreSQL database.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An on-premises Oracle database instance (up and running)
- An Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition database instance (up and running)

Limitations

- JSON-related queries require a fixed `KEY` and `VALUE` format. Not using that format returns the wrong result.
- If any change in JSON structure adds new `KEY` and `VALUE` pairs in the result section, the corresponding procedure or function must be changed in the SQL query.
- Some JSON-related functions are supported in earlier versions of Oracle and PostgreSQL but with fewer capabilities.

Product versions

- Oracle Database version 12.2 and later
- Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible version 9.5 and later
- AWS SCT latest version (tested using version 1.0.664)

Architecture

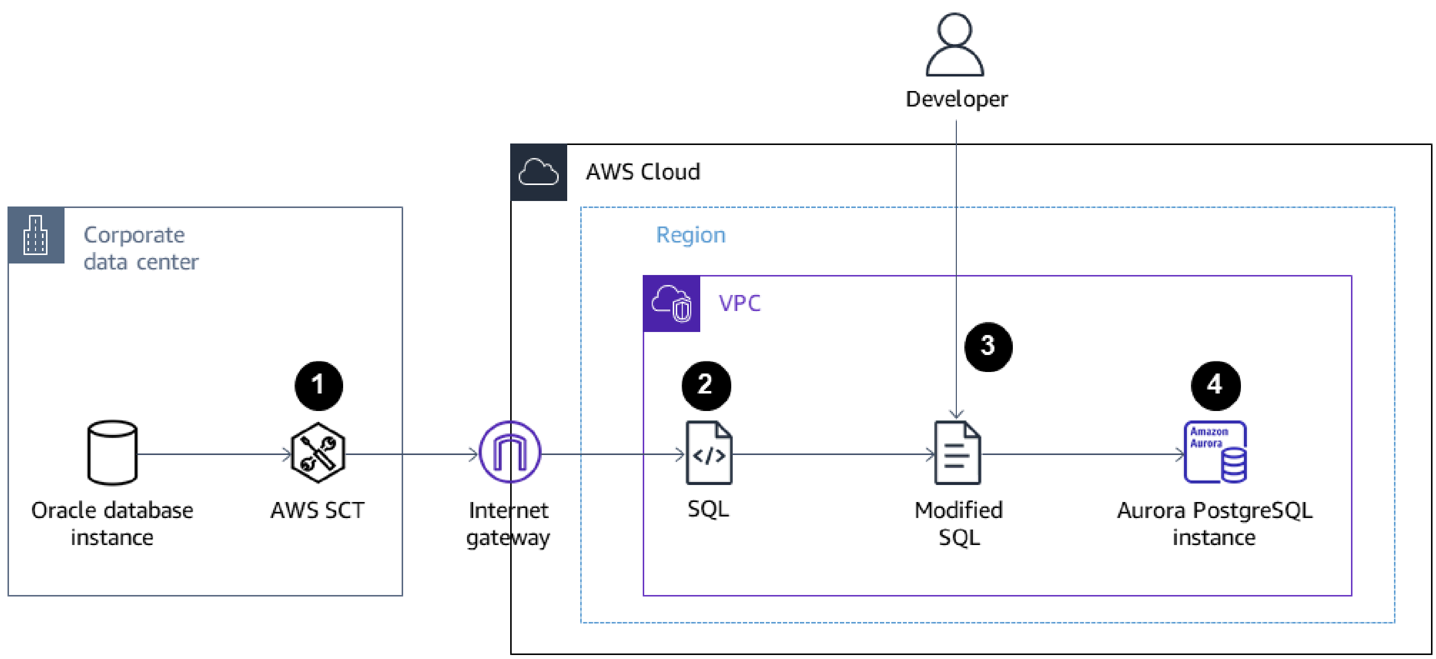
Source technology stack

- An Oracle database instance with version 19c

Target technology stack

- An Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database instance with version 13

Target architecture



1. Use AWS SCT with the JSON function code to convert the source code from Oracle to PostgreSQL.
2. The conversion produces PostgreSQL-supported migrated .sql files.
3. Manually convert the non-converted Oracle JSON function codes to PostgreSQL JSON function codes.

4. Run the .sql files on the target Aurora PostgreSQL-Compatible DB instance.

Tools

AWS services

- Amazon Aurora is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL.
- Amazon Relational Database Service (Amazon RDS) for PostgreSQL helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that’s compatible with the target database.

Other services

- Oracle SQL Developer is an integrated development environment that simplifies the development and management of Oracle databases in both traditional and cloud-based deployments.
- pgAdmin or DBeaver. pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects. DBeaver is a universal database tool.

Best practices

Oracle query has type `CAST` as the default when using the `JSON_TABLE` function. A best practice is to use `CAST` in PostgreSQL too, using double greater-than characters (>>).

For more information, see *Postgres_SQL_Read_JSON* in the *Additional information* section.

Epics

Generate the JSON data in the Oracle and PostgreSQL databases

Task	Description	Skills required
Store the JSON data in the Oracle database.	Create a table in the Oracle database, and store the JSON	Migration engineer

Task	Description	Skills required
	data in the CLOB column. Use the *Oracle_Table_Creation_Insert_Script* that's in the *Additional information* section.	
Store the JSON data in the PostgreSQL database.	Create a table in the PostgreSQL database, and store the JSON data in the TEXT column. Use the *Postgres_Table_Creation_Insert_Script* that's in the *Additional information* section.	Migration engineer

Convert the JSON into ROW format

Task	Description	Skills required
Convert the JSON data on the Oracle database.	Write an Oracle SQL query to read the JSON data into ROW format. For more details and example syntax, see *Oracle_SQL_Read_JSON* in the *Additional information* section.	Migration engineer
Convert the JSON data on the PostgreSQL database.	Write a PostgreSQL query to read the JSON data into ROW format. For more details and example syntax, see *Postgres_SQL_Read_JSON* in the *Additional information* section.	Migration engineer

Manually convert the JSON data using the SQL query and report the output in JSON format

Task	Description	Skills required
Perform aggregations and validation on the Oracle SQL query.	To manually convert the JSON data, perform a join, aggregation, and validation on the Oracle SQL query, and report the output in JSON format. Use the code under *Oracle_SQL_JSON_Aggregation_Join* in the *Additional information* section. 1. JOIN – The JSON-formatted data is passed as an input parameter to the query. An inner JOIN is made between this static data and the JSON data in the Oracle DB table `aws_test_table`. 2. Aggregation with validation – The JSON data has `KEY` and `VALUE` parameters with values such as `accountNumber`, `parentAccountNumber`, `businessUnitId` and `positionId`, which are used for `SUM` and `COUNT` aggregations. 3. JSON format – After the join and the aggregation, the data is reported	Migration engineer

Task	Description	Skills required
	in JSON format by using `JSON_OBJECT` and `JSON_ARRAYAGG` .	

Task	Description	Skills required
Perform aggregations and validation on the Postgres SQL query.	To manually convert the JSON data, perform a join, aggregation, and validatio n on the PostgreSQL query, and report the output in JSON format. Use the code under *Postgres_SQL_JSON_ Aggregation_Join* in the *Additional information* section. 1. JOIN – The JSON-form atted data (`tab1`) is passed as an input parameter to the `WITH` clause query. A JOIN is made between this static data and the JSON data, which is in the `tab` table. A JOIN is also made with the `WITH` clause, which has JSON data in the `aws_test_pg_table` table. 2. Aggregation – The JSON data has `KEY` and `VALUE` parameters with values such as `accountNumber` , `parentAccountNumbe r` , `businessUnitId` , and `positionId` , which are used for the `SUM` and `COUNT` aggregations. 3. JSON format – After the join and the aggregati	Migration engineer

Task	Description	Skills required
	on, the data is reported in JSON format by using `JSON_BUILD_OBJECT` and `JSON_AGG`.	

Convert the Oracle procedure into a PostgreSQL function that contains JSON queries

Task	Description	Skills required
Convert the JSON queries in the Oracle procedure into rows.	For the example Oracle procedure, use the previous Oracle query and the code under *Oracle_procedure_with_JSON_Query* in the *Additional information* section.	Migration engineer
Convert the PostgreSQL functions that have JSON queries into row-based data.	For the example PostgreSQL functions, use the previous PostgreSQL query and the code that's under *Postgres_function_with_JSON_Query* in the *Additional information* section.	Migration engineer

Related resources

- Oracle JSON functions
- PostgreSQL JSON functions
- Oracle JSON Functions Examples
- PostgreSQL JSON function examples
- AWS Schema Conversion Tool

Additional information

To convert JSON code from the Oracle database to PostgreSQL database, use the following scripts, in order.

1. Oracle_Table_Creation_Insert_Script

```
create table aws_test_table(id number,created_on date default sysdate,modified_on
 date,json_doc clob);

REM INSERTING into EXPORT_TABLE
SET DEFINE OFF;
Insert into aws_test_table (ID,CREATED_ON,MODIFIED_ON,json_doc)
values (1,to_date('02-AUG-2022 12:30:14','DD-MON-YYYY HH24:MI:SS'),to_date('02-AUG-2022
 12:30:14','DD-MON-YYYY HH24:MI:SS'),TO_CLOB(q'[{
  "metadata" : {
    "upperLastNameFirstName" : "ABC XYZ",
    "upperEmailAddress" : "abc@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "032323323",
    "displayName" : "Abc, Xyz",
    "firstName" : "Xyz",
    "lastName" : "Abc",
    "emailAddress" : "abc@gmail.com",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0100",
    "arrayPattern" : " -'",
    "a]')
|| TO_CLOB(q'[ccount" : {
      "companyId" : "SMGE",
      "businessUnitId" : 7,
      "accountNumber" : 42000,
      "parentAccountNumber" : 32000,
      "firstName" : "john",
      "lastName" : "doe",
      "street1" : "retOdertcaShr ",
      "city" : "new york",
      "postalcode" : "XY ABC",
      "country" : "United States"
    },
    "products" : [
```

```
      {
        "appUserGuid" : "i0acc4450000001823fbad478e2eab8a0",
        "id" : "0000000046",
]')
|| TO_CLOB(q'[        "name" : "ProView",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}]'));
Insert into aws_test_table (ID,CREATED_ON,MODIFIED_ON,json_doc) values (2,to_date('02-
AUG-2022 12:30:14','DD-MON-YYYY HH24:MI:SS'),to_date('02-AUG-2022 12:30:14','DD-MON-
YYYY HH24:MI:SS'),TO_CLOB(q'[{
  "metadata" : {
    "upperLastNameFirstName" : "PQR XYZ",
    "upperEmailAddress" : "pqr@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "54534343",
    "displayName" : "Xyz, pqr",
    "firstName" : "pqr",
    "lastName" : "Xyz",
    "emailAddress" : "pqr@gmail.com",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0090",
    "arrayPattern" : " -'",
    "account" : {
      "companyId" : "CARS",
      "busin]')
|| TO_CLOB(q'[essUnitId" : 6,
      "accountNumber" : 42001,
      "parentAccountNumber" : 32001,
      "firstName" : "terry",
      "lastName" : "whitlock",
      "street1" : "UO  123",
      "city" : "TOTORON",
      "region" : "NO",
      "postalcode" : "LKM 111",
      "country" : "Canada"
    },
    "products" : [
```

```
      {
        "appUserGuid" : "ia744d7790000016899f8cf3f417d6df6",
        "id" : "0000000014",
        "name" : "ProView eLooseleaf",
      ]')
|| TO_CLOB(q'[  "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}]'));

commit;
```

2. Postgres_Table_Creation_Insert_Script

```
create table aws_test_pg_table(id int,created_on date ,modified_on date,json_doc text);
insert into aws_test_pg_table(id,created_on,modified_on,json_doc)
values(1,now(),now(),'{
  "metadata" : {
    "upperLastNameFirstName" : "ABC XYZ",
    "upperEmailAddress" : "abc@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "032323323",
    "displayName" : "Abc, Xyz",
    "firstName" : "Xyz",
    "lastName" : "Abc",
    "emailAddress" : "abc@gmail.com",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0100",
    "arrayPattern" : " -",
    "account" : {
      "companyId" : "SMGE",
      "businessUnitId" : 7,
      "accountNumber" : 42000,
      "parentAccountNumber" : 32000,
      "firstName" : "john",
      "lastName" : "doe",
      "street1" : "retOdertcaShr ",
      "city" : "new york",
```

```
      "postalcode" : "XY ABC",
      "country" : "United States"
    },
    "products" : [
      {
        "appUserGuid" : "i0acc4450000001823fbad478e2eab8a0",
        "id" : "0000000046",
        "name" : "ProView",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}');


insert into aws_test_pg_table(id,created_on,modified_on,json_doc)
values(2,now(),now(),'{
  "metadata" : {
    "upperLastNameFirstName" : "PQR XYZ",
    "upperEmailAddress" : "pqr@gmail.com",
    "profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "54534343",
    "displayName" : "Xyz, pqr",
    "firstName" : "pqr",
    "lastName" : "Xyz",
    "emailAddress" : "a*b**@h**.k**",
    "productRegistrationStatus" : "Not registered",
    "positionId" : "0090",
    "arrayPattern" : " -",
    "account" : {
      "companyId" : "CARS",
      "businessUnitId" : 6,
      "accountNumber" : 42001,
      "parentAccountNumber" : 32001,
      "firstName" : "terry",
      "lastName" : "whitlock",
      "street1" : "UO  123",
      "city" : "TOTORON",
      "region" : "NO",
      "postalcode" : "LKM 111",
```

```
      "country" : "Canada"
    },
    "products" : [
      {
        "appUserGuid" : "ia744d7790000016899f8cf3f417d6df6",
        "id" : "0000000014",
        "name" : "ProView eLooseleaf",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}');
```

3. Oracle_SQL_Read_JSON

The following code blocks show how to convert Oracle JSON data into row format.

Example query and syntax

```
SELECT   JSON_OBJECT(
 'accountCounts' VALUE JSON_ARRAYAGG(
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
                        'clerkCount' VALUE clerk_count
                               ) ) ) ) FROM
    (SELECT   tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE 0 END) tax_count,
            SUM(CASE    WHEN tab_data.position_id = '0100' THEN 1 ELSE 0 END)
 attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0090' THEN 1 ELSE 0 END)
 non_attorney_count,
```

```
            SUM(CASE    WHEN tab_data.position_id = '0050' THEN 1 ELSE 0 END)
clerk_count
       FROM aws_test_table scco,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
 COLUMNS (
         parent_account_number NUMBER PATH
          '$.data.account.parentAccountNumber',
           account_number NUMBER PATH '$.data.account.accountNumber',
           business_unit_id NUMBER PATH '$.data.account.businessUnitId',
           position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
           ) AS tab_data
           INNER JOIN JSON_TABLE ( '{
       "accounts": [{
         "accountNumber": 42000,
         "parentAccountNumber": 32000,
         "businessUnitId": 7
       }, {
         "accountNumber": 42001,
         "parentAccountNumber": 32001,
         "businessUnitId": 6
       }]
     }', '$.accounts[*]' ERROR ON ERROR
     COLUMNS (
     parent_account_number PATH '$.parentAccountNumber',
     account_number PATH '$.accountNumber',
     business_unit_id PATH '$.businessUnitId')
     ) static_data
     ON ( static_data.parent_account_number = tab_data.parent_account_number
          AND static_data.account_number = tab_data.account_number
          AND static_data.business_unit_id = tab_data.business_unit_id )
       GROUP BY
           tab_data.business_unit_id,
           tab_data.parent_account_number,
           tab_data.account_number );
```

The JSON document stores the data as collections. Each collection can have KEY and VALUE pairs. Every VALUE can have nested KEY and VALUE pairs. The following table provides information about reading the specific VALUE from the JSON document.

KEY	HIERARCHY or PATH to be used to get the VALUE	VALUE
`profileType`	`metadata` -> `profileType`	"P"

`positionId`	`data` -> `positionId`	"0100"
`accountNumber`	`data` -> `account` -> `accountNumber`	42000

In the previous table, the KEY `profileType` is a VALUE of the `metadata` KEY. The KEY `positionId` is a VALUE of the `data` KEY. The KEY `accountNumber` is a VALUE of the `account` KEY, and the `account` KEY is a VALUE of the `data` KEY.

Example JSON document

```
{
  "metadata" : {
    "upperLastNameFirstName" : "ABC XYZ",
    "upperEmailAddress" : "abc@gmail.com",
"profileType" : "P"
  },
  "data" : {
    "onlineContactId" : "032323323",
    "displayName" : "Abc, Xyz",
    "firstName" : "Xyz",
    "lastName" : "Abc",
    "emailAddress" : "abc@gmail.com",
    "productRegistrationStatus" : "Not registered",
"positionId" : "0100",
    "arrayPattern" : " -",
    "account" : {
      "companyId" : "SMGE",
      "businessUnitId" : 7,
"accountNumber" : 42000,
      "parentAccountNumber" : 32000,
      "firstName" : "john",
      "lastName" : "doe",
      "street1" : "retOdertcaShr ",
      "city" : "new york",
      "postalcode" : "XY ABC",
      "country" : "United States"
    },
    "products" : [
      {
        "appUserGuid" : "i0acc4450000001823fbad478e2eab8a0",
```

```
        "id" : "0000000046",
        "name" : "ProView",
        "domain" : "EREADER",
        "registrationStatus" : false,
        "status" : "11"
      }
    ]
  }
}
```

SQL query that is used to get the selected fields from the JSON document

```
select parent_account_number,account_number,business_unit_id,position_id from
 aws_test_table aws,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
COLUMNS (
parent_account_number NUMBER PATH '$.data.account.parentAccountNumber',
account_number NUMBER PATH '$.data.account.accountNumber',
business_unit_id NUMBER PATH '$.data.account.businessUnitId',
position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'
)) as sc
```

In the previous query, `JSON_TABLE` is a built-in function in Oracle that converts the JSON data into row format. The JSON_TABLE function expects parameters in JSON format.

Every item in `COLUMNS` has a predefined `PATH`, and there an appropriate `VALUE` for a given `KEY` is returned in row format.

Result from the previous query

PARENT_ACCOUNT_NUMBER	ACCOUNT_NUMBER	BUSINESS_UNIT_ID	POSITION_ID
32000	42000	7	0100
32001	42001	6	0090

4. Postgres_SQL_Read_JSON

Example query and syntax

```
select *
```

```
from (
select (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER as
 parentAccountNumber,
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER as accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER as businessUnitId,
(json_doc::json->'data'->>'positionId')::VARCHAR as positionId
from aws_test_pg_table) d ;
```

In Oracle, PATH is used to identify the specific KEY and VALUE. However, PostgreSQL uses a HIERARCHY model for reading KEY and VALUE from JSON. The same JSON data mentioned under `Oracle_SQL_Read_JSON` is used in the following examples.

SQL query with type CAST not allowed

(If you force type CAST, the query fails with a syntax error.)

```
select *
from (
select (json_doc::json->'data'->'account'->'parentAccountNumber') as
 parentAccountNumber,
(json_doc::json->'data'->'account'->'accountNumber')as accountNumber,
(json_doc::json->'data'->'account'->'businessUnitId') as businessUnitId,
(json_doc::json->'data'->'positionId')as positionId
from aws_test_pg_table) d ;
```

Using a single greater-than operator (>) will return the VALUE defined for that KEY. For example, KEY: `positionId`, and VALUE: `"0100"`.

Type CAST is not allowed when you use the single greater-than operator (>).

SQL query with type CAST allowed

```
select *
from (
select (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER as
 parentAccountNumber,
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER as accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER as businessUnitId,
(json_doc::json->'data'->>'positionId')::varchar as positionId
from aws_test_pg_table) d ;
```

To use type CAST, you must use the double greater-than operator. If you use the single greater-than operator, the query returns the VALUE defined (for example, KEY: `positionId`, and VALUE: `"0100"`). Using the double greater-than operator (>>) will return the actual value defined for that KEY (for example, KEY: `positionId`, and VALUE: `0100`, without double quotation marks).

In the preceding case, `parentAccountNumber` is type CAST to INT, `accountNumber` is type CAST to INT, `businessUnitId` is type CAST to INT, and `positionId` is type CAST to VARCHAR.

The following tables show query results that explain role of the single greater-than operator (>) and the double greater-than operator (>>).

In the first table table, the query uses the single greater-than operator (>). Each column is in JSON type and can't be converted into another data type.

parentAccountNumber	accountNumber	businessUnitId	positionId
2003565430	2003564830	7	"0100"
2005284042	2005284042	6	"0090"
2000272719	2000272719	1	"0100"

In the second table, the query uses the double greater-than operator (>>). Each column supports type CAST based on the column value. For example, INTEGER in this context.

parentAccountNumber	accountNumber	businessUnitId	positionId
2003565430	2003564830	7	0100
2005284042	2005284042	6	0090
2000272719	2000272719	1	0100

5. Oracle_SQL_JSON_Aggregation_Join

Example query

```
SELECT
    JSON_OBJECT(
        'accountCounts' VALUE JSON_ARRAYAGG(
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
                        'clerkCount' VALUE clerk_count
                               ) ) ) )
FROM
    (SELECT
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE 0 END) tax_count,
            SUM(CASE    WHEN tab_data.position_id = '0100' THEN 1 ELSE 0 END)
 attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0090' THEN 1 ELSE 0 END)
 non_attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0050' THEN 1 ELSE 0 END)
 clerk_count
        FROM aws_test_table scco,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
  COLUMNS (
          parent_account_number NUMBER PATH
           '$.data.account.parentAccountNumber',
            account_number NUMBER PATH '$.data.account.accountNumber',
            business_unit_id NUMBER PATH '$.data.account.businessUnitId',
            position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
            ) AS tab_data
            INNER JOIN JSON_TABLE ( '{
        "accounts": [{
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }, {
          "accountNumber": 42001,
```

```
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }]
      }', '$.accounts[*]' ERROR ON ERROR
      COLUMNS (
      parent_account_number PATH '$.parentAccountNumber',
      account_number PATH '$.accountNumber',
      business_unit_id PATH '$.businessUnitId')
      ) static_data
      ON ( static_data.parent_account_number = tab_data.parent_account_number
           AND static_data.account_number = tab_data.account_number
           AND static_data.business_unit_id = tab_data.business_unit_id )
        GROUP BY
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number
    );
```

To convert the row-level data into JSON format, Oracle has built-in functions such as `JSON_OBJECT`, `JSON_ARRAY`, `JSON_OBJECTAGG`, and `JSON_ARRAYAGG`.

- `JSON_OBJECT` accepts two parameters: `KEY` and `VALUE`. The `KEY` parameter should be hardcoded or static in nature. The `VALUE` parameter is derived from table output.
- `JSON_ARRAYAGG` accepts `JSON_OBJECT` as a parameter. This helps in grouping the set of `JSON_OBJECT` elements as a list. For example, if you have a `JSON_OBJECT` element that has multiple records (multiple `KEY` and `VALUE` pairs in the dataset), `JSON_ARRAYAGG` appends the dataset and creates a list. According to the Data Structure language, `LIST` is group of elements. In this context, `LIST` is a group of `JSON_OBJECT` elements.

The following example shows one `JSON_OBJECT` element.

```
{
   "taxProfessionalCount": 0,
   "attorneyCount": 0,
   "nonAttorneyCount": 1,
   "clerkCount": 0
}
```

This next example shows two `JSON_OBJECT` elements, with `LIST` indicated by square braces (`[ ]`).

```
[
    {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
,
    {
        "taxProfessionalCount": 2,
        "attorneyCount": 1,
        "nonAttorneyCount": 3,
        "clerkCount":4
      }
]
```

Example SQL query

```
SELECT
    JSON_OBJECT(
        'accountCounts' VALUE JSON_ARRAYAGG(
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
                        'clerkCount' VALUE clerk_count
                               )
                        )
                                          )
               )
FROM
    (SELECT
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE   0 END
```

```
            )      tax_count,
            SUM(CASE    WHEN tab_data.position_id = '0100' THEN        1    ELSE
  0 END
            )      attorney_count,

            SUM(CASE    WHEN tab_data.position_id = '0090' THEN        1    ELSE
  0 END
            )      non_attorney_count,

            SUM(CASE    WHEN tab_data.position_id = '0050' THEN        1    ELSE
  0 END
            )      clerk_count

        FROM
            aws_test_table scco,  JSON_TABLE ( json_doc, '$' ERROR ON ERROR
            COLUMNS (
            parent_account_number NUMBER PATH '$.data.account.parentAccountNumber',
            account_number NUMBER PATH '$.data.account.accountNumber',
            business_unit_id NUMBER PATH '$.data.account.businessUnitId',
            position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
            ) AS tab_data
            INNER JOIN JSON_TABLE ( '{
        "accounts": [{
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }, {
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }]
      }', '$.accounts[*]' ERROR ON ERROR
      COLUMNS (
      parent_account_number PATH '$.parentAccountNumber',
      account_number PATH '$.accountNumber',
      business_unit_id PATH '$.businessUnitId')
      ) static_data ON ( static_data.parent_account_number =
 tab_data.parent_account_number
                        AND static_data.account_number = tab_data.account_number

                        AND static_data.business_unit_id =
 tab_data.business_unit_id )
        GROUP BY
            tab_data.business_unit_id,
```

```
            tab_data.parent_account_number,
            tab_data.account_number
    );
```

Example output from the previous SQL query

```
{
  "accountCounts": [
    {
      "businessUnitId": 6,
      "parentAccountNumber": 32001,
      "accountNumber": 42001,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
    {
      "businessUnitId": 7,
      "parentAccountNumber": 32000,
      "accountNumber": 42000,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

6. Postgres_SQL_JSON_Aggregation_Join

The PostgreSQL built-in functions `JSON_BUILD_OBJECT` and `JSON_AGG` convert the ROW-level data into JSON format. PostgreSQL `JSON_BUILD_OBJECT` and `JSON_AGG` are equivalent to Oracle `JSON_OBJECT` and `JSON_ARRAYAGG`.

Example query

```
select
JSON_BUILD_OBJECT ('accountCounts',
    JSON_AGG(
        JSON_BUILD_OBJECT ('businessUnitId',businessUnitId
        ,'parentAccountNumber',parentAccountNumber
        ,'accountNumber',accountNumber
        ,'totalOnlineContactsCount',online_contacts_count,
        'countByPosition',
            JSON_BUILD_OBJECT (
            'taxProfessionalCount',tax_professional_count
            ,'attorneyCount',attorney_count
            ,'nonAttorneyCount',non_attorney_count
            ,'clerkCount',clerk_count
            )
        )
    )
)
from (
with tab as (select * from (
select (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER as
 parentAccountNumber,
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER as accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER as businessUnitId,
(json_doc::json->'data'->>'positionId')::varchar as positionId
from aws_test_pg_table) a ) ,
tab1 as ( select
(json_array_elements(b.jc -> 'accounts') ->> 'accountNumber')::integer accountNumber,
(json_array_elements(b.jc -> 'accounts') ->> 'businessUnitId')::integer
 businessUnitId,
(json_array_elements(b.jc -> 'accounts') ->> 'parentAccountNumber')::integer
 parentAccountNumber
from (
select '{
        "accounts": [{
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }, {
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }]
      }'::json as jc) b)
```

```
select
tab.businessUnitId::text,
tab.parentAccountNumber::text,
tab.accountNumber::text,
SUM(1) online_contacts_count,
SUM(CASE WHEN tab.positionId::text = '0095' THEN 1 ELSE 0  END)
  tax_professional_count,
SUM(CASE WHEN tab.positionId::text = '0100' THEN 1 ELSE 0  END)      attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0090' THEN      1  ELSE      0 END)
  non_attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0050' THEN      1  ELSE      0 END)
  clerk_count
from tab1,tab
where tab.parentAccountNumber::INTEGER=tab1.parentAccountNumber::INTEGER
and tab.accountNumber::INTEGER=tab1.accountNumber::INTEGER
and tab.businessUnitId::INTEGER=tab1.businessUnitId::INTEGER
GROUP BY      tab.businessUnitId::text,
            tab.parentAccountNumber::text,
            tab.accountNumber::text) a;
```

Example output from the preceding query

Output from Oracle and PostgreSQL is exactly the same.


```
{
  "accountCounts": [
    {
      "businessUnitId": 6,
      "parentAccountNumber": 32001,
      "accountNumber": 42001,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
    {
      "businessUnitId": 7,
      "parentAccountNumber": 32000,
      "accountNumber": 42000,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
```

```
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

7.Oracle_procedure_with_JSON_Query

This code converts the Oracle procedure into a PostgreSQL function that has JSON SQL queries. It shows how the query transposes JSON into rows and the reverse.

```
CREATE OR REPLACE PROCEDURE p_json_test(p_in_accounts_json IN varchar2,
 p_out_accunts_json  OUT varchar2)
IS
BEGIN
/*
p_in_accounts_json paramter should have following format:
       {
        "accounts": [{
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }, {
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }]
      }
*/
SELECT
    JSON_OBJECT(
        'accountCounts' VALUE JSON_ARRAYAGG(
            JSON_OBJECT(
                'businessUnitId' VALUE business_unit_id,
                        'parentAccountNumber' VALUE parent_account_number,
                        'accountNumber' VALUE account_number,
                        'totalOnlineContactsCount' VALUE online_contacts_count,
                        'countByPosition' VALUE
                    JSON_OBJECT(
                        'taxProfessionalCount' VALUE tax_count,
```

```
                        'attorneyCount' VALUE attorney_count,
                        'nonAttorneyCount' VALUE non_attorney_count,
                        'clerkCount' VALUE clerk_count
                               ) ) ) )
into p_out_accunts_json
FROM
    (SELECT
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number,
            SUM(1) online_contacts_count,
            SUM(CASE WHEN tab_data.position_id = '0095' THEN  1 ELSE 0 END) tax_count,
            SUM(CASE    WHEN tab_data.position_id = '0100' THEN 1 ELSE 0 END)
 attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0090' THEN 1 ELSE 0 END)
 non_attorney_count,
            SUM(CASE    WHEN tab_data.position_id = '0050' THEN 1 ELSE 0 END)
 clerk_count
        FROM aws_test_table scco,JSON_TABLE ( json_doc, '$' ERROR ON ERROR
            COLUMNS (
            parent_account_number NUMBER PATH '$.data.account.parentAccountNumber',
            account_number NUMBER PATH '$.data.account.accountNumber',
            business_unit_id NUMBER PATH '$.data.account.businessUnitId',
            position_id VARCHAR2 ( 4 ) PATH '$.data.positionId'    )
            ) AS tab_data
            INNER JOIN JSON_TABLE ( p_in_accounts_json, '$.accounts[*]' ERROR ON ERROR

      COLUMNS (
      parent_account_number PATH '$.parentAccountNumber',
      account_number PATH '$.accountNumber',
      business_unit_id PATH '$.businessUnitId')
      ) static_data
      ON ( static_data.parent_account_number = tab_data.parent_account_number
           AND static_data.account_number = tab_data.account_number
           AND static_data.business_unit_id = tab_data.business_unit_id )
        GROUP BY
            tab_data.business_unit_id,
            tab_data.parent_account_number,
            tab_data.account_number
    );
EXCEPTION
WHEN OTHERS THEN
   raise_application_error(-20001,'Error while running the JSON query');
END;
```

```
/
```

Running the procedure

The following code block explains how you can run the previously created Oracle procedure with example JSON input to the procedure. It also gives you the result or output from this procedure.

```
set serveroutput on;
declare
v_out varchar2(30000);
v_in varchar2(30000):= '{
        "accounts": [{
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }, {
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
        }]
      }';
begin
  p_json_test(v_in,v_out);
  dbms_output.put_line(v_out);
end;
/
```

Procedure output

```
{
  "accountCounts": [
    {
      "businessUnitId": 6,
      "parentAccountNumber": 32001,
      "accountNumber": 42001,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
```

```
    {
      "businessUnitId": 7,
      "parentAccountNumber": 32000,
      "accountNumber": 42000,
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

8.Postgres_function_with_JSON_Query

Example function

```
CREATE OR REPLACE  FUNCTION f_pg_json_test(p_in_accounts_json  text)
RETURNS text
LANGUAGE plpgsql
AS
$$
DECLARE
 v_out_accunts_json   text;
BEGIN
SELECT
JSON_BUILD_OBJECT ('accountCounts',
    JSON_AGG(
        JSON_BUILD_OBJECT ('businessUnitId',businessUnitId
        ,'parentAccountNumber',parentAccountNumber
        ,'accountNumber',accountNumber
        ,'totalOnlineContactsCount',online_contacts_count,
        'countByPosition',
            JSON_BUILD_OBJECT (
            'taxProfessionalCount',tax_professional_count
            ,'attorneyCount',attorney_count
            ,'nonAttorneyCount',non_attorney_count
            ,'clerkCount',clerk_count
            ))))
INTO v_out_accunts_json
FROM (
WITH tab AS (SELECT * FROM (
```

```
SELECT (json_doc::json->'data'->'account'->>'parentAccountNumber')::INTEGER AS
 parentAccountNumber,
(json_doc::json->'data'->'account'->>'accountNumber')::INTEGER AS accountNumber,
(json_doc::json->'data'->'account'->>'businessUnitId')::INTEGER AS businessUnitId,
(json_doc::json->'data'->>'positionId')::varchar AS positionId
FROM aws_test_pg_table) a ) ,
tab1 AS ( SELECT
(json_array_elements(b.jc -> 'accounts') ->> 'accountNumber')::integer accountNumber,
(json_array_elements(b.jc -> 'accounts') ->> 'businessUnitId')::integer businessUnitId,
(json_array_elements(b.jc -> 'accounts') ->> 'parentAccountNumber')::integer
 parentAccountNumber
FROM (
SELECT p_in_accounts_json::json AS jc) b)
SELECT
tab.businessUnitId::text,
tab.parentAccountNumber::text,
tab.accountNumber::text,
SUM(1) online_contacts_count,
SUM(CASE WHEN tab.positionId::text = '0095' THEN 1 ELSE 0  END)
  tax_professional_count,
SUM(CASE WHEN tab.positionId::text = '0100' THEN 1 ELSE 0  END)      attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0090' THEN      1  ELSE      0 END)
  non_attorney_count,
SUM(CASE  WHEN tab.positionId::text = '0050' THEN      1  ELSE      0 END)
  clerk_count
FROM tab1,tab
WHERE tab.parentAccountNumber::INTEGER=tab1.parentAccountNumber::INTEGER
AND tab.accountNumber::INTEGER=tab1.accountNumber::INTEGER
AND tab.businessUnitId::INTEGER=tab1.businessUnitId::INTEGER
GROUP BY      tab.businessUnitId::text,
            tab.parentAccountNumber::text,
            tab.accountNumber::text) a;
RETURN v_out_accunts_json;
END;
$$;
```

Running the function

```
select    f_pg_json_test('{
        "accounts": [{
          "accountNumber": 42001,
          "parentAccountNumber": 32001,
          "businessUnitId": 6
```

```
        }, {
          "accountNumber": 42000,
          "parentAccountNumber": 32000,
          "businessUnitId": 7
        }]
      }')   ;
```

Function output

The following output is similar to the Oracle procedure output. The difference is that this output is in Text format.

```
{
  "accountCounts": [
    {
      "businessUnitId": "6",
      "parentAccountNumber": "32001",
      "accountNumber": "42001",
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 0,
        "nonAttorneyCount": 1,
        "clerkCount": 0
      }
    },
    {
      "businessUnitId": "7",
      "parentAccountNumber": "32000",
      "accountNumber": "42000",
      "totalOnlineContactsCount": 1,
      "countByPosition": {
        "taxProfessionalCount": 0,
        "attorneyCount": 1,
        "nonAttorneyCount": 0,
        "clerkCount": 0
      }
    }
  ]
}
```

Copy Amazon DynamoDB tables across accounts using AWS Backup

Ramkumar Ramanujam, Amazon Web Services

Summary

When working with Amazon DynamoDB on AWS, a common use case is to copy or sync DynamoDB tables in development, testing, or staging environments with the table data that is in the production environment. As a standard practice, each environment uses a different AWS account.

AWS Backup supports cross-Region and cross-account backup and restoration of data for DynamoDB, Amazon Simple Storage Service (Amazon S3), and other AWS services. This pattern provides the steps for using AWS Backup cross-account backup and restore to copy DynamoDB tables between AWS accounts.

Prerequisites and limitations

Prerequisites

- Two active AWS accounts that belong to the same organization in AWS Organizations
- Permissions to create DynamoDB tables in both accounts
- AWS Identity and Access Management (IAM) permissions to create and use AWS Backup vaults

Limitations

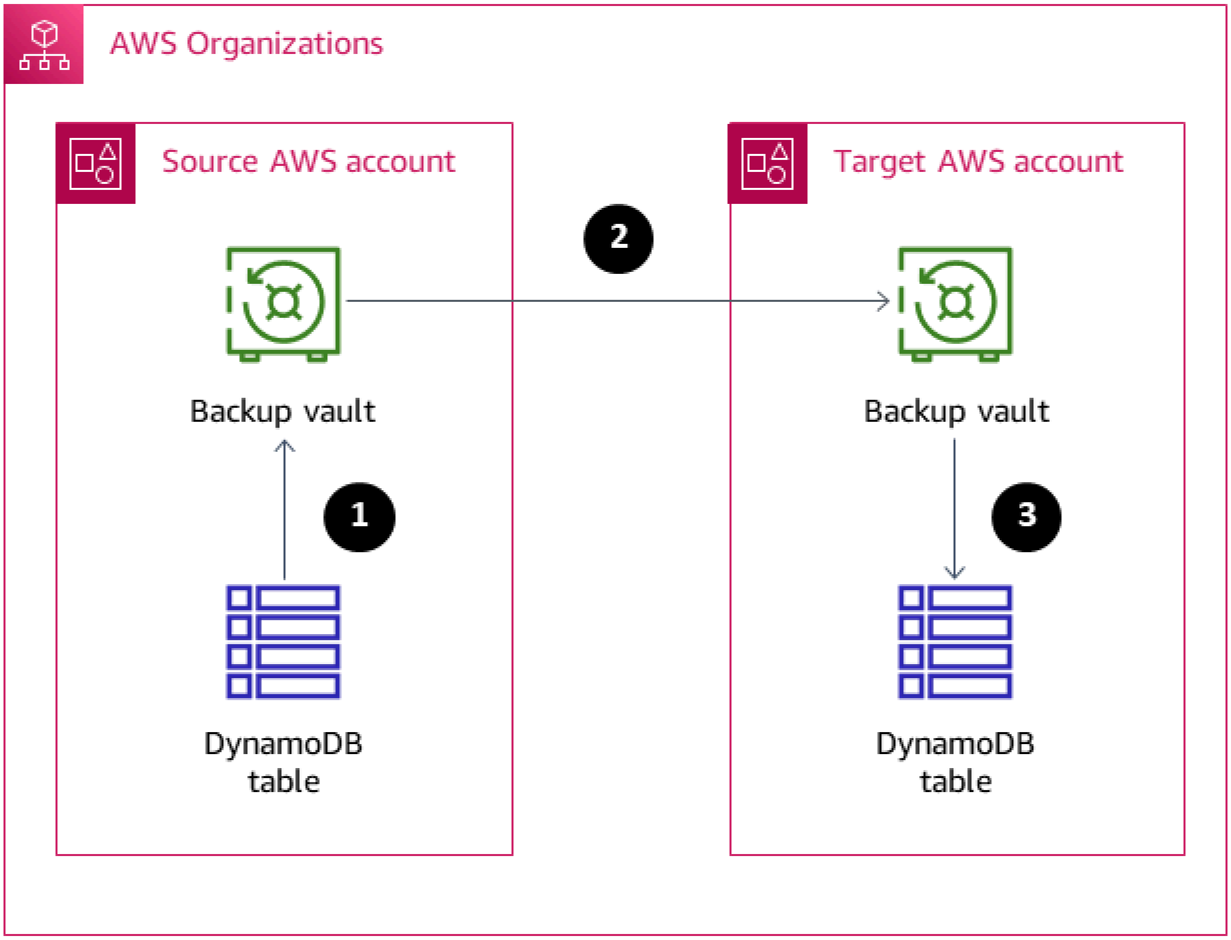
- Source and target AWS accounts should be part of the same organization in AWS Organizations.

Architecture

Target technology stack

- AWS Backup
- Amazon DynamoDB

Target architecture



1. Create the DynamoDB table backup in the AWS Backup backup vault in the source account.
2. Copy the backup to the backup vault in the target account.
3. Restore the DynamoDB table in the target account by using the backup from the backup vault in the target account.

Automation and scale

You can use AWS Backup to schedule backups to run at specific intervals.

Tools

- AWS Backup is a fully-managed service for centralizing and automating data protection across AWS services, in the cloud, and on premises. Using this service, you can configure backup policies and monitor activity for your AWS resources in one place. It allows you to automate and consolidate backup tasks that were previously performed service by service, and removes the need to create custom scripts and manual processes.

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

Epics

Turn on AWS Backup features in the source and target accounts

Task	Description	Skills required
Turn on advanced features for DynamoDB and cross-account backup.	In both the source and the target AWS accounts, do the following: 1. On the AWS Managemen t Console, open the AWS Backup console. 2. Choose **Settings**. 3. Under **Advanced features for Amazon DynamoDB backups**, confirm that **Advanced features** is enabled, or choose **Enable**. 4. Under **Cross-account management**, for **Cross-account backup**, choose **Enable**.	AWS DevOps, Migration engineer

Create backup vaults in the source and target accounts

Task	Description	Skills required
Create backup vaults.	In both the source and the target AWS accounts, do the following:	AWS DevOps, Migration engineer

Task	Description	Skills required
	1. On the [AWS Backup console](#), choose **Backup vaults**. 2. Choose **Create Backup vault**. 3. Copy the Amazon Resource Name (ARN) of the backup vault and save it. The ARNs of both the source and the target backup vaults will be required when you copy the DynamoDB table backup between the source and target accounts.	

Perform backup and restore using backup vaults

Task	Description	Skills required
In the source account, create a DynamoDB table backup.	To create a backup for the DynamoDB table in the source account, do the following: 1. On the AWS Backup **Dashboard** page, choose **Create on-demand backup**. 2. In the **Settings** section, for **Resource type**, select **DynamoDB**, and then select the table name.	AWS DevOps, DBA, Migration engineer

Task	Description	Skills required
	3. In the **Backup vault** dropdown list, select the backup vault that you created in the source account. 4. Select the **Retention period** that you want. 5. Choose **Create on-demand backup**. A new backup job is created. To monitor the status of the backup job, on the AWS Backup **Jobs** page, choose the **Backup Jobs** tab. All active, in-progress, and completed backup jobs are listed on this tab.	

Task	Description	Skills required
Copy the backup from the source account to the target account.	After the backup job is complete, copy the DynamoDB table backup from the backup vault in the source account to the backup vault in the target account. To copy the backup vault, in the source account, do the following: 1. On the [AWS Backup console](), choose **Backup vaults**. 2. Under **Backups**, choose the DynamoDB table backup. 3. Choose **Actions**, **Copy**. 4. Enter the AWS Region of the target account. 5. For **External vault ARN**, enter the ARN of the backup vault that you created in the target account. 6. To copy backups from the source account to the target account, in the target account backup vault, enable access from a different account.	AWS DevOps, Migration engineer, DBA

Task	Description	Skills required
Restore the backup in the target account.	In the target AWS account, do the following: 1. On the AWS Backup console, choose **Backup vaults**. 2. Under **Backups**, select the backup that you copied from the source account. 3. Choose **Actions**, **Restore**. 4. Enter the name of the target DynamoDB table that you want to restore.	AWS DevOps, DBA, Migration engineer

Related resources

- Using AWS Backup with DynamoDB
- Creating backup copies across AWS accounts
- AWS Backup pricing

Copy Amazon DynamoDB tables across accounts using a custom implementation

Ramkumar Ramanujam, Amazon Web Services

Summary

When working with Amazon DynamoDB on Amazon Web Services (AWS), a common use case is to copy or sync DynamoDB tables in development, testing, or staging environments with the table data that are in the production environment. As a standard practice, each environment uses a different AWS account.

DynamoDB now supports cross-account backup using AWS Backup. For information about associated storage costs when using AWS Backup, see AWS Backup pricing. When you use AWS Backup to copy across accounts, the source and target accounts must be part of an AWS Organizations organization. There are other solutions for cross-account backup and restore using AWS services such as AWS Glue. Using those solutions, however, increases the application footprint, because there are more AWS services to deploy and maintain.

You can also use Amazon DynamoDB Streams to capture table changes in the source account. Then you can initiate an AWS Lambda function, and make the corresponding changes in the target table in the target account. But that solution applies to use cases in which source and target tables must always be kept in sync. It might not apply to development, testing, and staging environments where data are updated frequently.

This pattern provides steps to implement a custom solution to copy a Amazon DynamoDB table from one account to another. This pattern can be implemented using common programming languages such as C#, Java, and Python. We recommend using a language that is supported by an AWS SDK.

Prerequisites and limitations

Prerequisites

- Two active AWS accounts
- DynamoDB tables in both the accounts
- Knowledge of AWS Identity and Access Management (IAM) roles and policies
- Knowledge of how to access Amazon DynamoDB tables using any common programming language, such as C#, Java, or Python

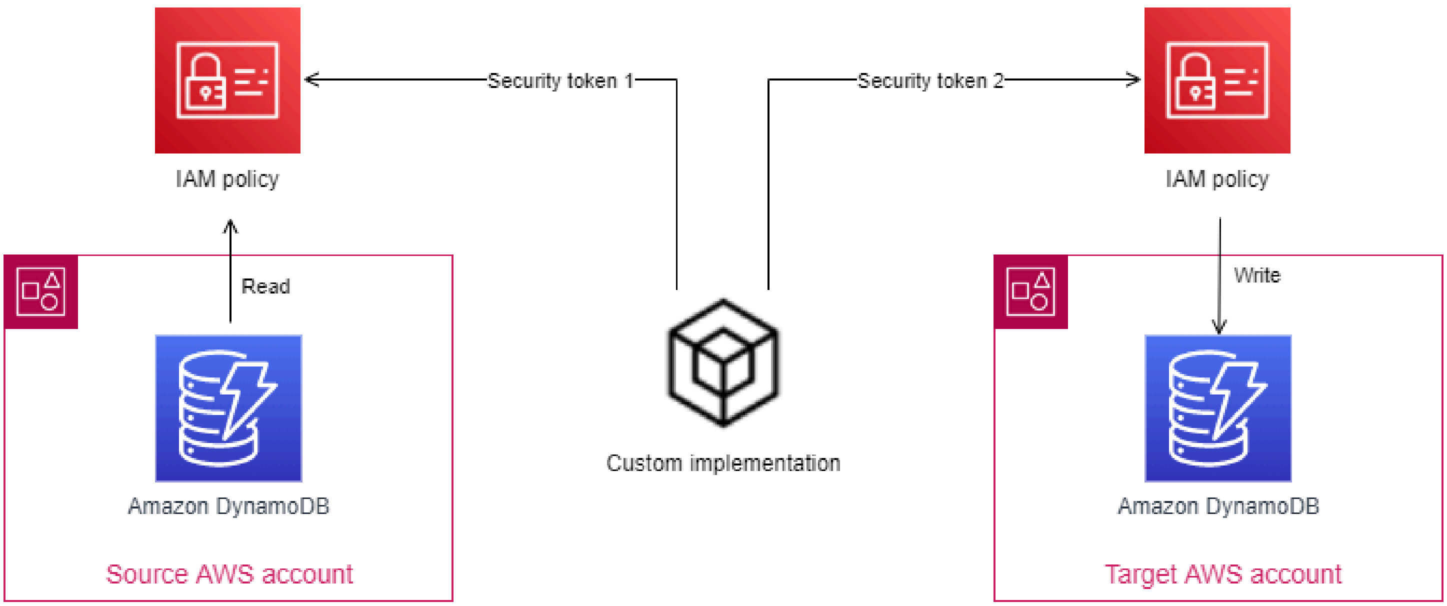
Limitations

This pattern applies to DynamoDB tables that are around 2 GB or smaller. With additional logic to handle connection or session interruptions, throttling, and failures and retries, it can be used for larger tables.

The DynamoDB scan operation, which reads items from the source table, can fetch only up to 1 MB of data in a single call. For larger tables, greater than 2 GB, this limitation can increase the total time to perform a full table copy.

Architecture

The following diagram shows the custom implementation between the source and target AWS accounts. IAM policies and security tokens are used with the custom implementation. Data is read from Amazon DynamoDB in the source account and written to DynamoDB in the target account.



Automation and scale

This pattern applies to DynamoDB tables that are smaller in size, around 2 GB.

To apply this pattern for larger tables, address the following issues:

- During the table copy operation, two active sessions are maintained, using different security tokens. If the table copy operation takes longer than the token expiration times, you must put in place logic to refresh the security tokens.

- If enough read capacity units (RCUs) and write capacity units (WCUs) are not provisioned, reads or writes on the source or target table might get throttled. Be sure to catch and handle these exceptions.
- Handle any other failures or exceptions and put a retry mechanism in place to retry or continue from where the copy operation failed.

Tools

Tools

- Amazon DynamoDB – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- The additional tools required will differ based on the programming language that you choose for the implementation. For example, if you use C#, you will need Microsoft Visual Studio and the following NuGet packages:
  - `AWSSDK`
  - `AWSSDK.DynamoDBv2`

Code

The following Python code snippet deletes and recreates a DynamoDB table using the Boto3 library.

Do not use the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` of an IAM user because these are long-term credentials, which should be avoided for programmatic access to AWS services. For more information about temporary credentials, see the *Best practices* section.

The `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `TEMPORARY_SESSION_TOKEN` used in the following code snippet are temporary credentials fetched from AWS Security Token Service (AWS STS).

```
import boto3
import sys
import json

#args = input-parameters = GLOBAL_SEC_INDEXES_JSON_COLLECTION,
 ATTRIBUTES_JSON_COLLECTION, TARGET_DYNAMODB_NAME, TARGET_REGION, ...
```

```
#Input param: GLOBAL_SEC_INDEXES_JSON_COLLECTION
#[{"IndexName":"Test-index","KeySchema":[{"AttributeName":"AppId","KeyType":"HASH"},
{"AttributeName":"AppType","KeyType":"RANGE"}],"Projection":
{"ProjectionType":"INCLUDE","NonKeyAttributes":["PK","SK","OwnerName","AppVersion"]}}]

#Input param: ATTRIBUTES_JSON_COLLECTION
#[{"AttributeName":"PK","AttributeType":"S"},
{"AttributeName":"SK","AttributeType":"S"},
{"AttributeName":"AppId","AttributeType":"S"},
{"AttributeName":"AppType","AttributeType":"N"}]

region = args['TARGET_REGION']
target_ddb_name = args['TARGET_DYNAMODB_NAME']

global_secondary_indexes = json.loads(args['GLOBAL_SEC_INDEXES_JSON_COLLECTION'])
attribute_definitions = json.loads(args['ATTRIBUTES_JSON_COLLECTION'])

# Drop and create target DynamoDB table
dynamodb_client = boto3.Session(
        aws_access_key_id=args['AWS_ACCESS_KEY_ID'],
        aws_secret_access_key=args['AWS_SECRET_ACCESS_KEY'],
        aws_session_token=args['TEMPORARY_SESSION_TOKEN'],
    ).client('dynamodb')

# Delete table
print('Deleting table: ' + target_ddb_name + ' ...')

try:
    dynamodb_client.delete_table(TableName=target_ddb_name)

    #Wait for table deletion to complete
    waiter = dynamodb_client.get_waiter('table_not_exists')
    waiter.wait(TableName=target_ddb_name)
    print('Table deleted.')
except dynamodb_client.exceptions.ResourceNotFoundException:
    print('Table already deleted / does not exist.')
    pass

print('Creating table: ' + target_ddb_name + ' ...')

table = dynamodb_client.create_table(
    TableName=target_ddb_name,
    KeySchema=[
        {
```

```
            'AttributeName': 'PK',
            'KeyType': 'HASH'  # Partition key
        },
        {
            'AttributeName': 'SK',
            'KeyType': 'RANGE'  # Sort key
        }
    ],
    AttributeDefinitions=attribute_definitions,
    GlobalSecondaryIndexes=global_secondary_indexes,
    BillingMode='PAY_PER_REQUEST'
)

waiter = dynamodb_client.get_waiter('table_exists')
waiter.wait(TableName=target_ddb_name)

print('Table created.')
```

Best practices

Temporary credentials

As a security best practice, while accessing AWS services programmatically, avoid using the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` of an IAM user because these are long-term credentials. Always try to use temporary credentials to access AWS services programmatically.

As an example, a developer hardcodes the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` of an IAM user in the application during development but fails to remove the hardcoded values before pushing changes to code repository. These exposed credentials can be used by unintended or malicious users, which can have serious implications (especially if the exposed credentials have admin privileges). These exposed credentials should be deactivated or deleted immediately by using the IAM console or AWS Command Line Interface (AWS CLI).

To get temporary credentials for programmatic access to AWS services, use AWS STS. Temporary credentials are valid only for the specified time (from 15 minutes up to 36 hours). The maximum allowed duration of temporary credentials varies depending on such factors as role settings and role chaining. For more information about AWS STS, see the documentation.

Epics

Set up DynamoDB tables

Task	Description	Skills required
Create DynamoDB tables.	Create DynamoDB tables, with indexes, in both source and target AWS accounts. Set the capacity provision ing as on-demand mode, which allows DynamoDB to scale read/write capacitie s dynamically based on the workload. Alternatively, you can use provisioned capacity with 4000 RCUs and 4000 WCUs.	App developer, DBA, Migration engineer
Populate the source table.	Populate the DynamoDB table in the source account with test data. Having at least 50 MB or more of test data helps you to see the peak and average RCUs consumed during table copy. You can then change the capacity provisioning as needed.	App developer, DBA, Migration engineer

Set up credentials to access the DynamoDB tables

Task	Description	Skills required
Create IAM roles to access the source and target DynamoDB tables.	Create an IAM role in the source account with permissions to access (read) the DynamoDB table in the source account. Add the source account as a trusted entity for this role. Create an IAM role in the target account with permissions to access (create, read, update, delete) the DynamoDB table in the target account. Add the target account as a trusted entity for this role.	App developer, AWS DevOps

Copy table data from one account to another

Task	Description	Skills required
Get temporary credentials for the IAM roles.	Get temporary credentials for IAM role created in source account. Get temporary credentials for IAM role created in target account. One way to get the temporary credentials for the	App developer, Migration engineer

Task	Description	Skills required
	IAM role is to use AWS STS from the AWS CLI. `aws sts assume-role --role-arn arn:aws:iam::<account-id>:role/<role-name> --role-session-name <session-name> --profile <profile-name>` Use the appropriate AWS profile (corresponding to the source or target account). For more information about different ways to get temporary credentials, see the following: • AWS Security Token Service API Reference • Getting IAM role credentials for CLI access	

Task	Description	Skills required
Initialize the DynamoDB clients for source and target DynamoDB access.	Initialize the DynamoDB clients, which are provided by the AWS SDK, for the source and target DynamoDB tables. • For the source DynamoDB client, use the temporary credentials fetched from the source account. • For the target DynamoDB client, use the temporary credentials fetched from the target account. For more information about making requests by using IAM temporary credentials, see the AWS documentation.	App developer

Task	Description	Skills required
Drop and recreate the target table.	Delete and recreate the target DynamoDB table (along with indexes) in the target account, using the target account DynamoDB client. Deleting all records from a DynamoDB table is a costly operation because it consumes provisioned WCUs. Deleting and recreating the table avoids those extra costs. You can add indexes to a table after you create it, but this takes 2–5 minutes longer. Creating indexes during table creation, by passing the indexes collection to the `createTable` call, is more efficient.	App developer

Task	Description	Skills required
Perform the table copy.	Repeat the following steps until all data are copied: • Perform a scan on the table in the source account, using the source DynamoDB client. Each DynamoDB scan retrieves only 1 MB of data from the table, so you must repeat this operation until all items, or records, are read. • For each set of scanned items, write the items to the table in the target account, with the target DynamoDB client, using the `BatchWriteItem` call in AWS SDK for DynamoDB. This reduces the number of `PutItem` requests made to DynamoDB. • `BatchWriteItem` has a limitation of 25 writes or puts, or up to 16 MB. You must add logic to accumulate scanned items in counts of 25 before calling `BatchWriteItem` . `BatchWriteItem` returns a list of items that could not be successfully copied. Using this list, add retry logic to perform another	App developer

Task	Description	Skills required
	`BatchWriteItem` call with only those items that did not succeed. For more information, see the reference implementation in C# (for dropping, creating, and populating tables) in the *Attachments* section. An example table config JavaScript Object Notation (JSON) file is also attached.	

Related resources

- Amazon DynamoDB documentation
- Creating an IAM user in your AWS account
- AWS SDKs
- Using temporary credentials with AWS resources

Additional information

This pattern was implemented using C# to copy a DynamoDB table with 200,000 items (average item size of 5 KB and table size of 250 MB). The target DynamoDB table was set up with provisioned capacity of 4000 RCUs and 4000 WCUs.

The complete table copy operation (from source account to target account), including dropping and recreating the table, took 5 minutes. Total capacity units consumed: 30,000 RCUs and approximately 400,000 WCUs.

For more information on DynamoDB capacity modes, see Read/Write capacity mode in the AWS documentation.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Create detailed cost and usage reports for Amazon RDS and Amazon Aurora

Lakshmanan Lakshmanan, Amazon Web Services

Sudarshan Narasimhan, None

Summary

This pattern shows how to track usage costs for Amazon Relational Database Service (Amazon RDS) or Amazon Aurora clusters by configuring user-defined cost allocation tags. You can use these tags to create detailed cost and usage reports in AWS Cost Explorer for clusters across multiple dimensions. For example, you can track usage costs at the team, project, or cost center level, and then analyze the data in Amazon Athena.

Prerequisites and limitations

Prerequisites

- An active AWS account
- One or more Amazon RDS or Amazon Aurora instances

Limitations

For tagging restrictions, see the AWS Billing User Guide.

Architecture

Target technology stack

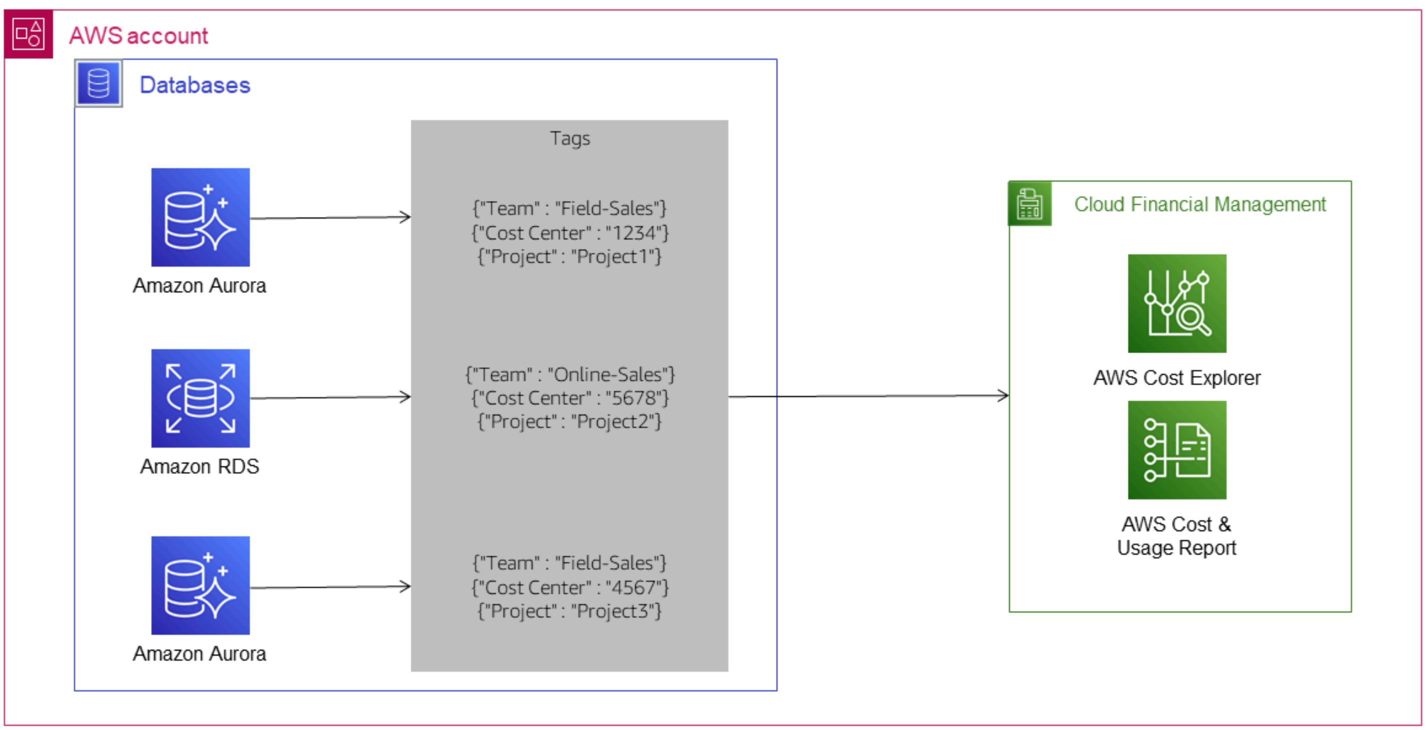
- Amazon RDS or Amazon Aurora
- AWS Cost and Usage Report
- AWS Cost Explorer
- Amazon Athena

Workflow and architecture

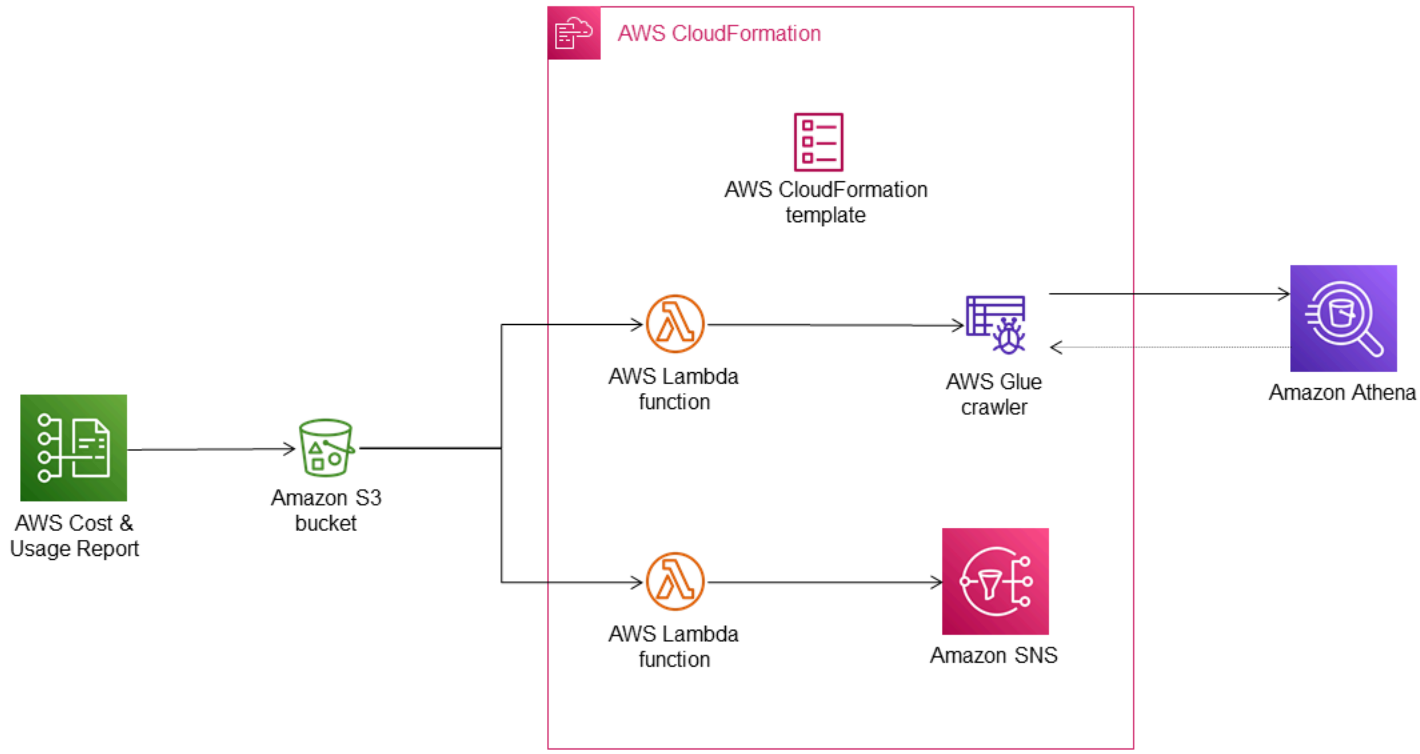
The tagging and analysis workflow consists of these steps:

1. A data engineer, database administrator, or AWS administrator creates user-defined cost allocation tags for the Amazon RDS or Aurora clusters.
2. An AWS administrator activates the tags.
3. The tags report metadata to AWS Cost Explorer.
4. A data engineer, database administrator, or AWS administrator creates a monthly cost allocation report.
5. A data engineer, database administrator, or AWS administrator analyzes the monthly cost allocation report by using Amazon Athena.

The following diagram shows how to apply tags to track usage costs for Amazon RDS or Aurora instances.



The following architecture diagram shows how the cost allocation report is integrated with Amazon Athena for analysis.



The monthly cost allocation report is stored in an Amazon S3 bucket that you specify. When you set up Athena with the AWS CloudFormation template, as described in the *Epics* section, the template provisions several additional resources, including an AWS Glue crawler, an AWS Glue database, an Amazon Simple Notification System (Amazon SNS) event, AWS Lambda functions, and AWS Identity and Access Management (IAM) roles for the Lambda functions. As new cost data files arrive in the S3 bucket, event notifications are used to forward these files to a Lambda function for processing. The Lambda function initiates an AWS Glue crawler job to create or update the table in the AWS Glue Data Catalog. This table is then used to query data in Athena.

Tools

- Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.
- Amazon Aurora is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.

- AWS CloudFormation is an infrastructure as code (IaC) service that allows you to easily model, provision, and manage AWS and third-party resources.
- AWS Cost Explorer helps you view and analyze your AWS costs and usage.

Epics

Create and activate tags for your Amazon RDS or Aurora cluster

Task	Description	Skills required
Create user-defined cost allocation tags for your Amazon RDS or Aurora cluster.	To add tags to a new or existing Amazon RDS or Aurora cluster, follow the instructions in Adding, listing, and removing tags in the *Amazon Aurora User Guide*. **Note** For information about how to set up an Amazon Aurora cluster, see the instructions for MySQL and PostgreSQL in the *Amazon Aurora User Guide*.	AWS administrator, Data engineer, DBA
Activate the user-defined cost allocation tags.	Follow the instructions in Activating user-defined cost allocation tags in the *AWS Billing User Guide*.	AWS administrator

Create cost and usage reports

Task	Description	Skills required
Create and configure cost and usage reports for your clusters.	1. Sign in to the AWS Management Console and open the [AWS Billing console](https://console.aws.amazon.com/billing/). 2. In the left navigation pane, choose **Cost & Usage Reports**. 3. Choose **Create report**. 4. Provide a report name, keep the default settings for other options, and then choose **Next**. 5. Choose **Configure** and provide the details of an existing S3 bucket. You can also choose to create a new S3 bucket from this screen. Choose **Next**. 6. Verify the default policy that will be applied to your bucket, select the confirmation check box, and then choose **Save**. 7. For **Report path prefix**, specify the prefix you want to prepend to the report name. 8. For **Time granularity**, choose **Hourly**, **Daily** or **Monthly**, depending on how often you want data	App owner, AWS administrator, DBA, General AWS, Data engineer

Task	Description	Skills required
	to be collected for the report. 9. For **Report versioning**, choose whether you want new versions of the report to be created separately or overwrite the existing report with each version. 10. For **Enable report data integration for**, choose **Amazon Athena**. Verify that the compression type is set to Parquet. 11. Choose **Next**. 12. Review the report settings, and then choose **Review and Complete**. The data will be available in 24 hours.	

Analyze cost and usage report data

Task	Description	Skills required
Analyze the cost and usage report data.	1. Set up and use Athena to analyze the report data. For instructions, see [Querying Cost and Usage Reports using Amazon Athena](#) in the *AWS Cost and Usage Reports User Guide*. We recommend	App owner, AWS administr ator, DBA, General AWS, Data engineer

Task	Description	Skills required
	that you use the AWS CloudFormation template provided by Athena. 2. Run Athena queries. For example, you can use the following SQL query to check the status of the data refresh. `select status from cost_and_usage_data_status` For more information, see Running Amazon Athena queries in the *AWS Cost and Usage Reports User Guide*. **Note** When you run your SQL query, make sure that the correct database is selected from the dropdown list.	

Related resources

References

- Setting up Athena using AWS CloudFormation templates (recommended)
- Setting up Athena manually

- Running Amazon Athena queries
- Loading report data to other resources

Tutorials and videos

- Analyze Cost and Usage Reports using Amazon Athena (YouTube video)

Deploy SQL Server failover cluster instances on Amazon EC2 and Amazon FSx by using Terraform

Mark Hudson and Matt Burgess, Amazon Web Services

Summary

This pattern uses Terraform to deploy SQL Server failover cluster instances (FCIs) across Windows Server Failover Cluster (WSFC) nodes on Amazon Elastic Compute Cloud (Amazon EC2). In addition, the pattern uses Amazon FSx shared storage for data and log files.

When SQL Server databases are migrated to AWS, the first choice is Amazon RDS for SQL Server. However, sometimes Amazon RDS for SQL Server isn't suitable and SQL Server must be deployed on Amazon EC2 in a highly available architecture. In this solution, SQL Server FCIs are installed across WSFC nodes.

The Terraform module included with this pattern provisions up to two Amazon EC2 SQL Server instances. An Amazon FSx for Windows File Server file system acts as the quorum witness and stores shared data and log files. Regardless of the number of instances configured, the SQL Server instance nodes will always create and join an FCI cluster to ensure environmental parity. (Typically, one instance is configured for development and two instances for production environments.) For configurations that use two nodes for high availability, an internal Network Load Balancer is provisioned. The Network Load Balancer uses a health probe configured on the FCI cluster to identify which node is the primary.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Amazon Virtual Private Cloud (Amazon VPC) with two subnets in separate Availability Zones.
- Amazon VPC [DHCP option set](). Configure the domain name to resolve to your Active Directory domain name and the domain and NetBIOS name servers to point to your Active Directory domain controllers. For more information, see *VPC configuration* in [Additional information]().
- AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD).
- Custom Amazon Machine Image (AMI). For more details, see *AMI configuration* in [Additional information]().

- Amazon Simple Storage Service (Amazon S3) bucket containing the SQL Server ISO image. This prerequisite is only required if using EC2 Image Builder with the provided `component.yaml` file to build the custom AMI.
- AWS Key Management Service (AWS KMS) encryption key.
- By default, SQL Server is installed using a developer edition product key. Production systems are expected to use a valid product key passed to the module by the relevant variable.

Limitations

- This solution requires AWS Managed Microsoft AD. However, if you prefer, you can use a self-managed Active Directory implementation instead. To do so, modify the included Amazon FSx Terraform module to remove the `active_directory_id` attribute. Then, add the four attributes that are required for self-managed Active Directory as shown in the Terraform documentation.
- SQL Server is set up to use mixed mode authentication. If you prefer, you can use Windows-only authentication. To do so, in the provided user data script, remove the `/SECURITYMODE` and `/SAPWD` parameters that are supplied to the `setup.exe` command. You can remove the `sql_accounts.tf` file, and you can modify the `instances.tf` file to remove the `sql_sa_password` entry.
- When deleting a deployed cluster, you must remove the corresponding virtual computer objects and individual computer objects in Active Directory. To remove the objects, use Active Directory administrative tools.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

This solution was tested with the following versions:

- Windows Server 2019
- SQL Server 2019
- Terraform v0.13.0

Architecture

Source technology stack

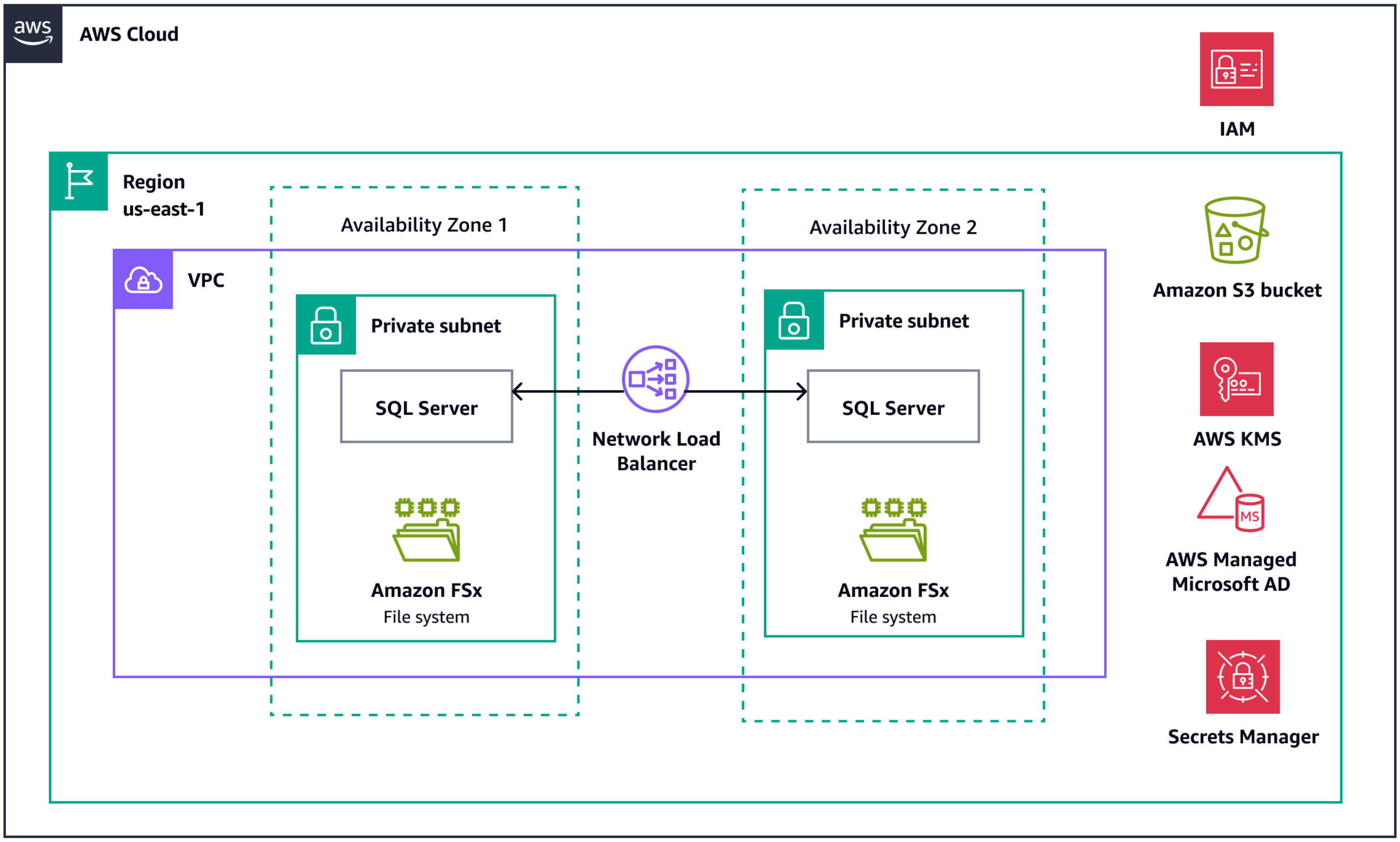
- SQL Server

Target technology stack

- SQL Server FCI on WSFC nodes using Amazon EC2
- Amazon FSx for Windows File Server
- Amazon S3 bucket
- AWS Secrets Manager
- AWS Managed Microsoft AD
- AWS KMS
- AWS Identity and Access Management (IAM)

Target architecture

The following diagram shows the architecture for this solution.



The diagram shows the following:

- An IAM role providing the EC2 instances access to AWS KMS and Secrets Manager
- Two SQL Server nodes deployed on Amazon EC2 instances in private subnets across two Availability Zones
- A Network Load Balancer for facilitating connections to the active SQL Server instance (not deployed when setting up a single node cluster)
- Amazon FSx for Windows File Server file system deployed in both private subnets for shared storage by the SQL Server nodes
- Secrets Manager for storing Active Directory and SQL Server credentials and configuration
- Amazon S3 bucket for storing the SQL Server installation image
- AWS Managed Microsoft AD for Windows authentication
- AWS KMS for creating the encryption key

Automation and scale

You can automate the deployment of the target architecture by using the Terraform modules that are in the GitHub repository. You must modify the `terraform.tfvars` file to include variable values that are specific to your environment. The Amazon S3 bucket, AWS Managed Microsoft AD components, AWS KMS encryption key, and some secrets are prerequisites for this deployment and are not included in the Terraform code.

Tools

AWS services

- AWS Directory Service for Microsoft Active Directory enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud. In this pattern, AWS Managed Microsoft AD is used for Windows Server and SQL Server authentication and for DNS.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down. In this pattern, SQL Server failover cluster instances are installed on Amazon EC2 instances.
- EC2 Image Builder helps you automate the creation, management, and deployment of customized server images.
- Amazon FSx for Windows File Server provides fully managed shared storage on Windows Server. In this pattern, FSx for Windows File Server provides shared storage for SQL Server data and log files and the quorum witness.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data. In this pattern, it's used to encrypt Secrets Manager secrets, the SQL Server storage on Amazon Elastic Block Store (Amazon EBS) volumes, and the FSx for Windows File Server file system.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. In this pattern, Active Directory credentials for installing and running SQL Server, the `sa` user credentials, and the database connection information are stored in Secrets Manager.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data. This pattern uses an Amazon S3 bucket to store the SQL Server installation image.

- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- Microsoft SQL Server FCIs are installed across Windows Server cluster nodes. In addition, they can be installed across multiple subnets. In this pattern, SQL Server FCI instances are installed across WSFC nodes.
- Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources. In this pattern, Terraform is used to create the resources and configure the SQL Server FCI instances.
- Windows Server Failover Clustering provides infrastructure features that support the high-availability of hosted server applications such as SQL Server. In this pattern, the FCI nodes make use of WSFC functionality to provide local high availability through redundancy at the instance level.

Code repository

The code for this pattern is available in the GitHub cluster-amazon-elastic-compute-cloud-amazon-fsx-microsoft-sql-server repository. The following resources are available in the repository:

- A `README.md` file that provides an overview of the solution and additional installation and usage information
- A base set of Terraform configuration files and an Amazon FSx specific module to provision the components for this pattern
- An instance setup script that is run as an Amazon EC2 user data script
- A `component.yaml` file that Image Builder can use to create a custom AMI

Best practices

Security and patching

- The AMI prerequisite installations and configurations are the minimum requirements to deploy SQL Server FCI clusters. Additional software and configurations might be required to comply with your organization's standards and security requirements.

- After deployment, patch Windows on an ongoing basis. Either directly patch the running instances, or create a new AMI with the latest Windows patches and replace the instances (one at a time) using the new AMI. AWS releases new Windows AMIs monthly that contain the latest operating system patches, drivers, and launch agents. We recommend that you check for the latest AMI when you launch new instances, or when you build your own custom images.
- The Amazon EC2 instances are configured to allow all outgoing traffic. When deployed in a production environment, outbound rules in the security group should be put in place to restrict this traffic to the required destinations.
- The FSx for Windows File Server file system can automatically record audit logs for file share and file and folder access and ship them to your desired destination if this is a requirement in your environment.
- Rotate Secrets Manager secrets automatically on a regular basis. For the Amazon EC2 instance key pair, consider an automated rotation solution as described in [How to use AWS Secrets Manager to securely store and rotate SSH key pairs](). For the Active Directory credentials and SQL Server `sa` credential secrets, set up automated rotation according to your policies for password management.

Active Directory management

- As part of the FCI cluster, Windows generates a Computer Name Object (CNO) in Active Directory. The CNO responds to DNS requests and forwards traffic to the active SQL node. We do *not* recommend using this Active Directory-provided DNS. The TTL is too high to provide a reasonable failover time and often takes upwards of 5 minutes to reflect the new primary IP address. In contrast, for highly available installations, the internal Network Load Balancer is configured to failover within 30 seconds.
- An Active Directory domain administrator is required to create the cluster. This requirement is because of the elevated permissions required to create the cluster objects and modify permissions in Active Directory. However, the SQL Server services don't need to run as a domain administrator. Therefore, we recommend that you create a second Active Directory user for this purpose. However, you can eliminate this user if the services will run as the domain administrator user. In that case, the domain administrator user must be added to the Active Directory administrators group that is created as part of this pattern.

Epics

Set up cluster credentials

Task	Description	Skills required
Create Active Directory groups.	In AWS Managed Microsoft AD, create the following groups: • **Cluster administrator group** – This group will be added to the local Administrators group on each cluster node. • **Cluster remote desktop group** – This group will be added to the local Remote Desktop Users group on each cluster node. For more details, see Creating an AWS Managed Microsoft AD group in the AWS documentation.	AD administrator
Create Active Directory users.	In AWS Managed Microsoft AD, create the following users • **Domain administrator user** – Use this account to create the cluster. • **Domain user** – The SQL Server services will use this account to run. Add this user to the cluster administrator group that	AD administrator

Task	Description	Skills required
	you created in the previous task. For more details, see [Creating an AWS Managed Microsoft AD user](#) in the AWS documentation.	
Add Active Directory credentials to secrets.	Use Secrets Manager to create four secrets to store the following information: • Domain administrator user's username • Domain administrator user's password • Domain user's username • Domain user's password For more details, see [Create an AWS Secrets Manager secret](#) in the AWS documenta tion.	AWS administrator

Prepare the environment

Task	Description	Skills required
Create the Windows AMI.	Create a custom Windows AMI that includes the prerequisite software and configurations. For more	AWS administrator, AWS DevOps

Task	Description	Skills required
	details, see Additional Information.	
Install Terraform.	To install Terraform, follow the instructions on the Terraform website.	AWS DevOps
Clone the repository.	Clone this pattern's repository. For more details, see Cloning a repository on the GitHub website.	AWS DevOps

Install the cluster

Task	Description	Skills required
Modify the Terraform variables.	Update the supplied `terraform.tfvars` file to set all variables to values appropriate for your environment. For example, update the `domain_group_administrators` and `domain_group_rdp_users` variables to use your Active Directory domain name and the name of the previously created Active Directory groups.	AWS DevOps
Initialize Terraform.	To see the proposed deployment, navigate to the root of the repository. Use	AWS DevOps

Task	Description	Skills required
	the Terraform command line interface (CLI) to run `terraform init` and then run `terraform plan`.	
Deploy resources.	To deploy the SQL cluster and associated resources, use the Terraform CLI to run `terraform apply`.	AWS DevOps, AWS administr ator

Task	Description	Skills required
Validate the deployment.	To validate the deployment, use the following steps: 1. Connect to one of the deployed Windows Amazon EC2 instances by using Remote Desktop. 2. For more details, see Connect to your Windows instance using RDP in the AWS documentation. 3. Open the Windows Failover Cluster Manager. Validate that the cluster was created, and that the SQL Server role was created and is running. 4. To test connectivity and authentication with SQL Server, use a database tool such as SQL Server Management Studio to connect to the SQL Server endpoint. (The endpoint value is stored in Secrets Manager.) For more details, see Connect to Microsoft SQL Server on Amazon EC2 in the AWS documentation.	DBA, AWS systems administr ator

Troubleshooting

Issue	Solution
Terraform provisioning completed but Windows Failover Cluster Manager does not show a cluster has been created or the cluster is in a non-operable state.	The entire installation of the resources and configuration of the clusters can can take 45-60 minutes. After Terraform has completed , the user data script must run to completion, which requires multiple reboots. To monitor progress, you can use the `Checkpoints` directory in the `C:\` drive and the SQL Server installation logs in `C:\Program Data\Microsoft SQL Server\150\Log` . When finished, the **Installation Complete** message is available in the `C:\ProgramData\Amazon\EC2-Windows\Launch\Log\UserdataExecution.log` file.
After provisioning a functioning cluster, using Terraform to delete and re-create the cluster does not succeed. Terraform completes, but the cluster is not set up properly.	Part of the provisioning process involves registering machines and virtual objects to Active Directory and Active Directory DNS. When computer names exist for the Amazon EC2 cluster nodes and the cluster node, FCI cannot initialize correctly and will fail provisioning. To fix this issue, perform the following steps: 1. Delete the Amazon EC2 nodes, cluster virtual computer, and cluster id in Active Directory users and computers. 2. Delete the DNS entry for the cluster virtual computer in Active Directory DNS. 3. Run the following command to delete the cluster id random string Terraform resource `terragrunt destroy -target=random_string.cluster_id` . This

Issue	Solution
	action will delete the existing Amazon EC2 instances. 4. Run `terraform apply`, and expect the following three new resources: Two FCI Amazon EC2 instances and 1 random string cluster id.

Related resources

AWS documentation

- Create custom images with Image Builder
- Create a KMS key
- Creating a general purpose bucket
- Creating a key policy
- Creating your AWS Managed Microsoft AD

Additional information

Terraform module information

This module uses a mixture of AMI configuration and user data configuration to obtain a good mix of provisioning time and stability. During provisioning, Windows requires multiple restarts and waits. A checkpoints method has been implemented to protect against infinite loops during restarts of the persistent user data. User data is configured to be persistent. Therefore, the user data configuration scripts have, and must continue, to be developed to be idempotent. Idempotency streamlines the updating process, allowing for instances to be swapped out during an update cycle without manual configuration to rejoin or re-create FCI clusters.

SQL Server connection strings and failover clustering

The module will publish a secret containing the endpoint address that should be used in connection strings for this database. The secret name follows the format `{environment_name}/sqlserver/{cluster_name}/endpoint`. For installations where only one node is used, you can expect this to be the IP address of the Amazon EC2 instance SQL Server interface. For high availability installations (two instances), you can expect this to be the DNS name of the internal Network Load Balancer.

Failover clustering virtual IPs aren't supported in this module. A virtual IP must remain in the same subnet in order to work. In AWS, a single subnet can't span multiple Availability Zones. Therefore, the use of virtual IPs would remove the ability for this module to be considered highly available.

Each Amazon EC2 instance is given three private IP addresses. Their usais as follows:

- **Primary IP for network traffic** – The source IP for egress traffic.
- **FCI communications –** Used to maintain the state and sync of the failover cluster.
- **SQL Server (TCP port 1433) –** Listener and also listens for heartbeat traffic to determine which instance is primary.

VPC configuration

The prerequisites list a DHCP options set that's configured to use Active Directory for DNS resolution. However, this prerequisite isn't a hard requirement. The hard requirement is that the EC2 instances must be able to resolve your Active Directory domain name. Meeting this requirement can be accomplished in other ways, such as by using Amazon Route 53 Resolver endpoints. For more information, see Integrating your Directory Service's DNS resolution with Amazon Route 53 Resolvers (AWS Blog post).

AMI configuration

The AMI used in this pattern must contain the following prerequisite software and configurations:

1. Download and expand the SQL Server 2019 installation files into `C:\SQL_Install_media`.
2. Install the following Windows features:
   - `Install-WindowsFeature Failover-Clustering`
   - `Install-WindowsFeature RSAT-AD-PowerShell`
   - `Install-WindowsFeature RSAT-AD-Tools`
   - `Install-WindowsFeature RSAT-Clustering-Mgmt`
   - `Install-WindowsFeature RSAT-Clustering-PowerShell`
   - `Install-WindowsFeature RSAT-Clustering-CmdInterface`
3. Disable the Windows firewall as follows:

- Get-NetFirewallProfile | Set-NetFirewallProfile -Enabled False

4. Enable CredSSP authentication method (replace <domain> with your organization's Windows domain name) as follows:
   - Enable-WSManCredSSP -Role "Server" -Force
   - Enable-WSManCredSSP -Role "Client" -DelegateComputer *.<domain>.com -Force
5. Set the following registry keys:
   - Allow NTLM authentication credentials:
     - HKLM:\Software\Policies\Microsoft\Windows\CredentialsDelegation
       - Name: AllowFreshCredentialsWhenNTLMOnly
       - Value: 1
       - Type: REG_DWORD
   - Allow local domain computers to use NTLM from PowerShell:
     - Path: HKLM:\Software\Policies\Microsoft\Windows\CredentialsDelegation\AllowFreshCredentialsWhenNTLMOnly
       - Name: 1
       - Value: wsman/*.<domain>.com
       - Type: REG_SZ
6. Set up the [PowerShell Gallery]() as follows:
   - [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
   - Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force
   - Set-PSRepository -Name PSGallery -InstallationPolicy Trusted
7. Install the following Windows PowerShell modules*:
   - Install-Module -Name ComputerManagementDsc
   - Install-Module -Name FailOverClusterDsc
   - Install-Module -Name PSDscResources
   - Install-Module -Name xSmbShare
   - Install-Module -Name xActiveDirectory
   - Install-Module -Name SqlServer

To use Image Builder to create the AMI, follow the instructions in Create an image pipeline using the EC2 Image Builderconsole wizard in the Image Builder documentation. To create the recipe's component with the previous prerequisites, use the following steps:

1. Download the component.yaml file from the `ami` folder of the GitHub repository.
2. Copy the contents into a new Image Builder component.
3. Update the following placeholders with your information:
   - `<domain>` – Your Active Directory domain name
   - `<bucket_name>` – Name of the Amazon S3 bucket that contains the SQL Server image

Emulate Oracle PL/SQL associative arrays in Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL

Rajkumar Raghuwanshi, Bhanu Ganesh Gudivada, and Sachin Khanna, Amazon Web Services

Summary

This pattern describes how to emulate Oracle PL/SQL associative arrays with empty index positions in Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL environments. It also describes some of the differences between Oracle PL/SQL associative arrays and PostgreSQL arrays with regard to how each handles empty index positions during migrations.

We provide a PostgreSQL alternative to using `aws_oracle_ext` functions for handling empty index positions when migrating an Oracle database. This pattern uses an additional column to store index positions, and it maintains Oracle's handling of sparse arrays while incorporating native PostgreSQL capabilities.

Oracle

In Oracle, collections can be initialized as empty and populated using the `EXTEND` collection method, which appends `NULL` elements to the array. When working with PL/SQL associative arrays indexed by `PLS_INTEGER`, the `EXTEND` method adds `NULL` elements sequentially, but elements can also be initialized at nonsequential index positions. Any index position that isn't explicitly initialized remains empty.

This flexibility allows for sparse array structures where elements can be populated at arbitrary positions. When iterating through collections using a `FOR LOOP` with `FIRST` and `LAST` bounds, only the initialized elements (whether `NULL` or with a defined value) are processed, while empty positions are skipped.

PostgreSQL (Amazon Aurora and Amazon RDS)

PostgreSQL handles empty values differently from `NULL` values. It stores empty values as distinct entities that use one byte of storage. When an array has empty values, PostgreSQL assigns sequential index positions just like non-empty values. But sequential indexing requires additional processing because the system must iterate through all indexed positions, including the empty ones. This makes traditional array creation inefficient for sparse datasets.

AWS Schema Conversion Tool

The AWS Schema Conversion Tool (AWS SCT) typically handles Oracle-to-PostgreSQL migrations using `aws_oracle_ext` functions. In this pattern, we propose an alternative approach that uses native PostgreSQL capabilities, which combines PostgreSQL array types with an additional column for storing index positions. The system can then iterate through arrays using just the index column.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An AWS Identity and Access Management (IAM) user with administrator permissions.
- An instance that's compatible with Amazon RDS or Aurora PostgreSQL.
- A basic understanding of relational databases.

Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Product versions

This pattern was tested with the following versions:

- Amazon Aurora PostgreSQL 13.3
- Amazon RDS for PostgreSQL 13.3
- AWS SCT 1.0.674
- Oracle 12c EE 12.2

Architecture

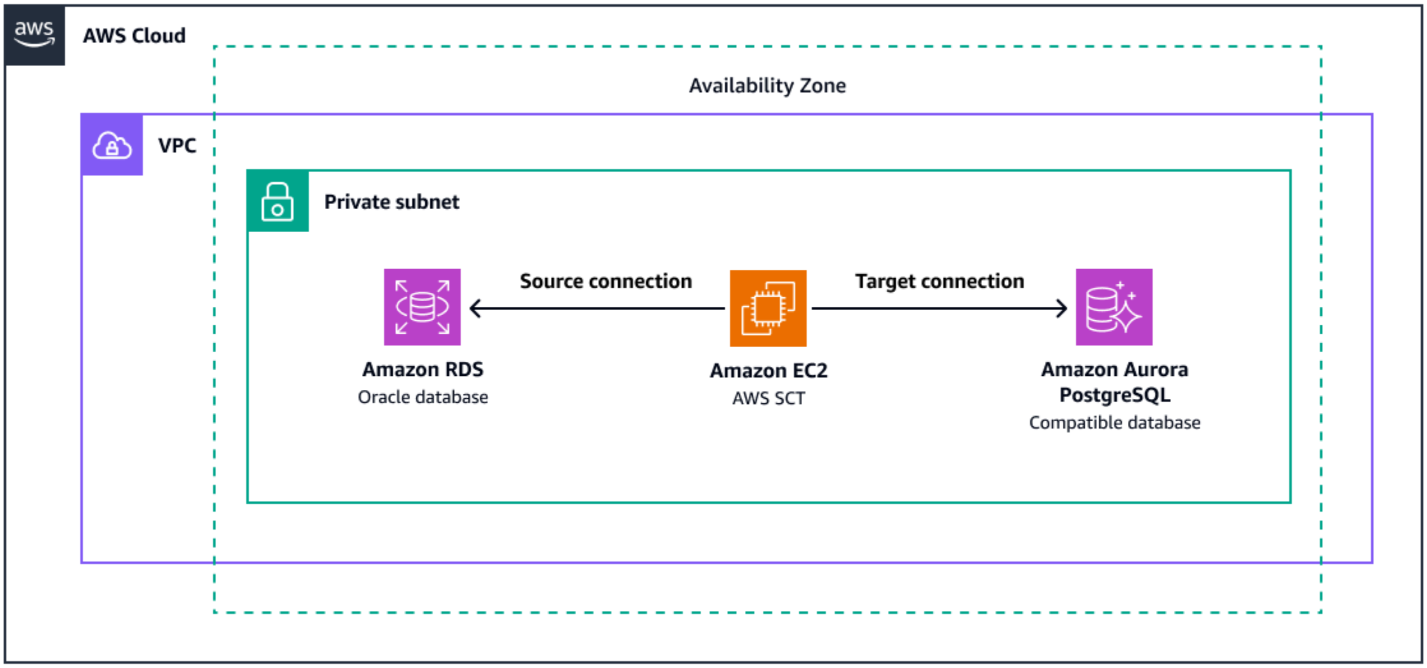
Source technology stack

- On-premises Oracle database

Target technology stack

- Amazon Aurora PostgreSQL
- Amazon RDS for PostgreSQL

Target architecture



The diagram shows the following:

- A source Amazon RDS for Oracle database instance
- An Amazon EC2 instance with AWS SCT for converting Oracle functions to the PostgreSQL equivalent
- A target database that's compatible with Amazon Aurora PostgreSQL

Tools

AWS services

- Amazon Aurora is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL.
- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.

- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Relational Database Service (Amazon RDS) for Oracle helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.
- Amazon Relational Database Service (Amazon RDS) for PostgreSQL helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.

Other tools

- Oracle SQL Developer is an integrated development environment that simplifies the development and management of Oracle databases in both traditional and cloud-based deployments.
- pgAdmin is an open source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects. In this pattern, pgAdmin connects to the RDS for PostgreSQL database instance and queries the data. Alternatively, you can use the psql command line client.

Best practices

- Test data set boundaries and edge scenarios.
- Consider implementing error handling for out-of-bounds index conditions.
- Optimize queries to avoid scanning sparse data sets.

Epics

Oracle associative array behavior (source)

Task	Description	Skills required
Create a source PL/SQL block in Oracle.	Create a source PL/SQL block in Oracle that uses the following associative array:	DBA

Task	Description	Skills required

	<pre>DECLARE
	    TYPE country_codes
	 IS TABLE OF VARCHAR2(
	100) INDEX BY pls_integ
	er;
	    cc country_codes;
	    cc_idx NUMBER :=
	 NULL;
	BEGIN
	    cc(7) := 'India';
	    cc(3) := 'UK';
	    cc(5) := 'USA';
	    cc(0) := 'China';
	    cc(-2) := 'Invalid'
	;
	    dbms_output.put_li
	ne('cc_length:' ||
	 cc.COUNT);
	    IF (cc.COUNT > 0)
	 THEN
	        cc_idx :=
	 cc.FIRST;
	        FOR i IN
	 1..cc.COUNT LOOP
	            dbms_outp
	ut.put_line('cc_id
	x:' || cc_idx || '
	 country:' || cc(cc_idx
	));
	            cc_idx :=
	 cc.next(cc_idx);
	        END LOOP;
	    END IF;
	END;</pre>	

Task	Description	Skills required
Run the PL/SQL block.	Run the source PL/SQL block in Oracle. If there are gaps between the index values of an associative array, no data is stored in those gaps. This allows the Oracle loop to iterate only through the index positions.	DBA

Review the output.	Five elements were inserted into the array (`cc`) at nonconsecutive intervals. The array count is shown in the following output:
	`cc_length:5`
	`cc_idx:-2 country:I`
	`nvalid`
	`cc_idx:0 country:China`
	`cc_idx:3 country:UK`
	`cc_idx:5 country:USA`
	`cc_idx:7 country:India`
		DBA

PostgreSQL associative array behavior (target)

Task	Description	Skills required

Create a target PL/pgSQL block in PostgreSQL.	Create a target PL/pgSQL block in PostgreSQL that uses the following associative array:
	`DO $$`
	`DECLARE`
	`    cc character`
	` varying(100)[];`
		DBA

Task	Description	Skills required

	``
	    cc_idx integer :=
	 NULL;
	BEGIN
	    cc[7] := 'India';
	    cc[3] := 'UK';
	    cc[5] := 'USA';
	    cc[0] := 'China';
	    cc[-2] := 'Invalid'
	;
	    RAISE NOTICE
	 'cc_length: %',
	 ARRAY_LENGTH(cc, 1);
	    IF (ARRAY_LE
	NGTH(cc, 1) > 0) THEN
	        FOR i IN
	 ARRAY_LOWER(cc,
	 1)..ARRAY_UPPER(cc,
	 1)
	        LOOP
	            RAISE NOTICE
	 'cc_idx:% country:%',
	 i, cc[i];
	        END LOOP;
	    END IF;
	END;
	$$;
	``	

Run the PL/pgSQL block.	Run the target PL/pgSQL block in PostgreSQL. If there are gaps between the index values of an associative array, no data is stored in those gaps. This allows the Oracle loop to iterate only through the index positions.	DBA

Task	Description	Skills required
Review the output.	The array length is greater than 5 because NULL is stored in the gaps between index positions. As shown in the following output, the loop completes 10 iterations to retrieve 5 values in the array. ``` cc_length:10 cc_idx:-2 country:Invalid cc_idx:-1 country:<NULL> cc_idx:0 country:China cc_idx:1 country:<NULL> cc_idx:2 country:<NULL> cc_idx:3 country:UK cc_idx:4 country:<NULL> cc_idx:5 country:USA cc_idx:6 country:<NULL> cc_idx:7 country:India ```	DBA

Emulate Oracle associative array behavior

Task	Description	Skills required
Create a target PL/pgSQL block with an array and user-defined type.	To optimize performance and match Oracle's functionality, we can create a user-defined type that stores both index positions and their corresponding data. This	DBA

Task	Description	Skills required

	approach reduces unnecessary iterations by maintaining direct associations between indices and values.
	
	``
	DO $$
	DECLARE
	    cc country_codes[];
	    cc_append country_codes := NULL;
	    i record;
	BEGIN
	    cc_append.idx = 7;
	    cc_append.val = 'India';
	    cc := array_append(cc, cc_append);
	    cc_append.idx = 3;
	    cc_append.val = 'UK';
	    cc := array_append(cc, cc_append);
	    cc_append.idx = 5;
	    cc_append.val = 'USA';
	    cc := array_append(cc, cc_append);
	    cc_append.idx = 0;
	    cc_append.val = 'China';
	    cc := array_append(cc, cc_append);
	    cc_append.idx = -2;
	    cc_append.val = 'Invalid';
	    cc := array_append(cc, cc_append);
	    RAISE NOTICE 'cc_length: %', ARRAY_LENGTH(cc, 1);
	``	

Task	Description	Skills required

	``
	    IF (ARRAY_LE
	NGTH(cc, 1) > 0) THEN
	        FOR i IN (
	            SELECT
	                *
	            FROM
	
	 unnest(cc)
	            ORDER BY
	                idx)
	                LOOP
	
	 RAISE NOTICE 'cc_idx:
	% country:%', i.idx,
	 i.val;
	                END
	 LOOP;
	    END IF;
	END;
	$$;
	``	

Run the PL/pgSQL block.	Run the target PL/pgSQL block. If there are gaps between the index values of an associative array, no data is stored in those gaps. This allows the Oracle loop to iterate only through the index positions.	DBA

Task	Description	Skills required

Review the output.	As shown in the following output, the user-defined type stores only populated data elements, which means that the array length matches the number of values. As a result, LOOP iterations are optimized to process only existing data, eliminating the need to track empty positions.
	`cc_length:5`
	`cc_idx:-2 country:Invalid`
	`cc_idx:0 country:China`
	`cc_idx:3 country:UK`
	`cc_idx:5 country:USA`
	`cc_idx:7 country:India`
		DBA

Related resources

AWS documentation

- AWS database blog
- Oracle to Aurora PostgreSQL migration playbook

Other documentation

- Oracle associative arrays
- PostgreSQL array functions and operators
- PostgreSQL user-defined types

Enable encrypted connections for PostgreSQL DB instances in Amazon RDS

Rohit Kapoor, Amazon Web Services

Summary

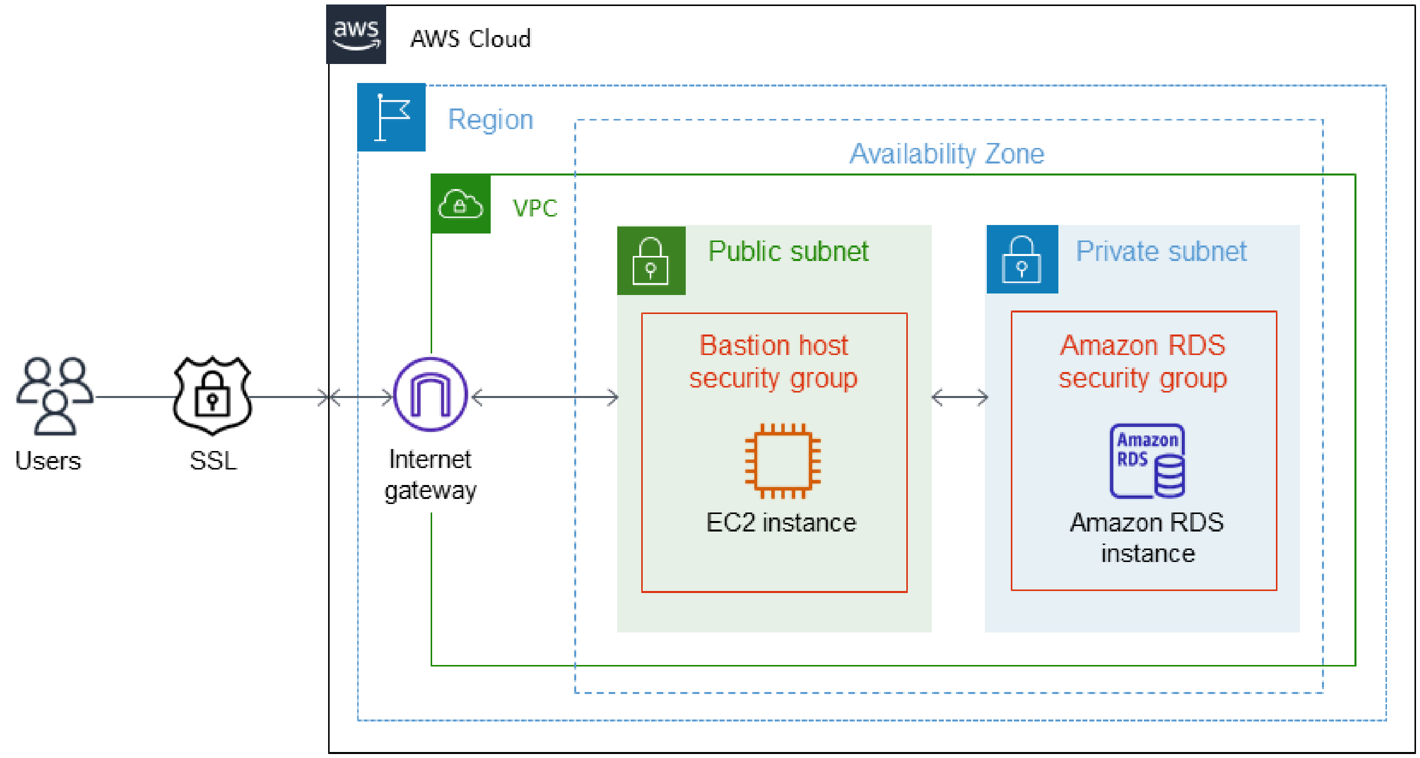
Amazon Relational Database Service (Amazon RDS) supports SSL encryption for PostgreSQL DB instances. Using SSL, you can encrypt a PostgreSQL connection between your applications and your Amazon RDS for PostgreSQL DB instances. By default, Amazon RDS for PostgreSQL uses SSL/TLS and expects all clients to connect by using SSL/TLS encryption. Amazon RDS for PostgreSQL supports TLS versions 1.1 and 1.2.

This pattern describes how you can enable encrypted connections for an Amazon RDS for PostgreSQL DB instance. You can use the same process to enable encrypted connections for Amazon Aurora PostgreSQL-Compatible Edition.

Prerequisites and limitations

- An active AWS account
- An Amazon RDS for PostgreSQL DB instance
- An SSL bundle

Architecture



Tools

- pgAdmin is an open-source administration and development platform for PostgreSQL. You can use pgAdmin on Linux, Unix, macOS, and Windows to manage your database objects in PostgreSQL 10 and later.
- PostgreSQL editors provide a more user-friendly interface to help you create, develop, and run queries, and to edit code according to your requirements.

Best practices

- Monitor unsecure database connections.
- Audit database access rights.
- Make sure that backups and snapshots are encrypted at rest.
- Monitor database access.
- Avoid unrestricted access groups.
- Enhance your notifications with Amazon GuardDuty.
- Monitor policy adherence regularly.

Epics

Download a trusted certificate and import it into your trust store

Task	Description	Skills required
Load a trusted certificate to your computer.	To add certificates to the Trusted Root Certification Authorities store for your computer, follow these steps. (These instructions use Window Server as a example.) 1. In Windows Server, choose **Start**, **Run**, and then type **mmc**. 2. In the console, choose **File**, **Add/Remove Snap-in**. 3. Under **Available snap-ins**, choose **Certificates**, and then choose **Add**. 4. Under **This snap-in will always manage certifica tes for**, choose **Computer account**, **Next**. 5. Choose **Local computer**, **Finish**. 6. If you have no more snap-ins to add to the console, choose **OK**. 7. In the console tree, double-click **Certificates**. 8. Right-click **Trusted Root Certification Authorities**.	DevOps engineer, Migration engineer, DBA

Task	Description	Skills required
	9. Choose **All Tasks**, **Import** to import the downloaded certificates. 10. Follow the steps in the Certificate Import Wizard.	

Force SSL connections

Task	Description	Skills required
Create a parameter group and set the rds.force_ssl parameter.	If the PostgreSQL DB instance has a custom parameter group, edit the parameter group and change `rds.force_ssl` to 1. If the DB instance uses the default parameter group that doesn't have `rds.force_ssl` enabled, create a new parameter group. You can modify the new parameter group by using the Amazon RDS API or manually as in the following instructions. To create a new parameter group: 1. Sign in to the AWS Management console and open [the Amazon RDS console](https://console.aws.amazon.com/rds/) for the AWS Region that hosts the DB instance.	DevOps engineer, Migration engineer, DBA

Task	Description	Skills required
	2. In the navigation pane, choose **Parameter groups**. 3. Choose **Create parameter group**, and set the following values: • For **Parameter group family**, choose **postgres14**. • For **Group name**, type **pgsql-<database_instance>-ssl**. • For **Description**, enter a free-form description for the parameter group you're adding. • Choose **Create**. 4. Choose the parameter group that you created. 5. From **Parameter group actions**, choose **Edit**. 6. Find **rds.force_ssl** and change its setting to **1**. **Note** Conduct client-side testing before changing this parameter. 7. Choose **Save changes**.	

Task	Description	Skills required
	To associate the parameter group with your PostgreSQL DB instance: 1. On the Amazon RDS console, in the navigatio n pane, choose **Databases** , and then choose the PostgreSQL DB instance. 2. Choose **Modify**. 3. Under **Additional configuration**, choose the new parameter group, and then choose **Continue**. 4. Under **Schedule modificat ions**, choose **Apply immediately**. 5. Choose **Modify DB instance**. For more information, see the Amazon RDS documentation.	
Force SSL connections.	Connect to the Amazon RDS for PostgreSQL DB instance. Connection attempts that don’t use SSL are rejected with an error message. For more information, see the Amazon RDS documentation.	DevOps engineer, Migration engineer, DBA

Install SSL extension

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install the SSL extension.</td><td>
1. Launch a psql or pgAdmin connection as a DBA.
2. Call the ssl_is_used() function to determine if SSL is being used.
<pre><code>select ssl_is_used();</code></pre>
The function returns <code>t</code> if the connection is using SSL; otherwise, it returns <code>f</code>.
3. Install the SSL extension.
<pre><code>create extension
 sslinfo;
show ssl;
select ssl_cipher();</code></pre>
For more information, see the Amazon RDS documentation.</td><td>DevOps engineer, Migration engineer, DBA</td></tr>
</table>

Configure your PostgreSQL client for SSL

Task	Description	Skills required
Configure a client for SSL.	By using SSL, you can start the PostgreSQL server with support for encrypted connections that use TLS protocols. The server listens for both standard and SSL connections on the same TCP	DevOps engineer, Migration engineer, DBA

Task	Description	Skills required

	port, and negotiates with any connecting client on whether to use SSL. By default, this is a client option.
	
	If you're using the psql client:
	
	1. Make sure that the Amazon RDS certificate has been loaded to your local computer.
	2. Launch an SSL client connection by adding the following:
	
	`psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser sslmode=verify-full sslrootcert=rds-ssl-ca-cert.pem`
	`select ssl_cipher();`
	
	For other PostgreSQL clients:
	
	• Modify the respective application public key parameter. This might be available as an option, as part of your connection string, or as a property on the connection page in GUI tools.	

Task	Description	Skills required
	Review the following pages for these clients: • pgAdmin documentation • JDBC documentation	

Troubleshooting

Issue	Solution
Cannot download the SSL certificate.	Check your connection to the website, and retry downloading the certificate to your local computer.

Related resources

- Amazon RDS for PostgreSQL documentation
- Using SSL with a PostgreSQL DB instance (Amazon RDS documentation)
- Secure TCP/IP Connections with SSL (PostgreSQL documentation)
- Using SSL (JDBC documentation)

Encrypt an existing Amazon RDS for PostgreSQL DB instance

Piyush Goyal, Shobana Raghu, and Yaser Raja, Amazon Web Services

Summary

This pattern explains how to encrypt an existing Amazon Relational Database Service (Amazon RDS) for PostgreSQL DB instance in the AWS Cloud with minimal downtime. This process works for Amazon RDS for MySQL DB instances as well.

You can enable encryption for an Amazon RDS DB instance when you create it, but not after it's created. However, you can add encryption to an unencrypted DB instance by creating a snapshot of your DB instance, and then creating an encrypted copy of that snapshot. You can then restore a DB instance from the encrypted snapshot to get an encrypted copy of your original DB instance. If your project allows for downtime (at least for write transactions) during this activity, this is all you need to do. When the new, encrypted copy of the DB instance becomes available, you can point your applications to the new database. However, if your project doesn't allow for significant downtime for this activity, you need an alternate approach that helps minimize the downtime. This pattern uses the AWS Database Migration Service (AWS DMS) to migrate and continuously replicate the data so that the cutover to the new, encrypted database can be done with minimal downtime.

Amazon RDS encrypted DB instances use the industry standard AES-256 encryption algorithm to encrypt your data on the server that hosts your Amazon RDS DB instances. After your data is encrypted, Amazon RDS handles authentication of access and decryption of your data transparently, with minimal impact on performance. You don't need to modify your database client applications to use encryption.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An unencrypted Amazon RDS for PostgreSQL DB instance
- Experience working with (creating, modifying, or stopping) AWS DMS tasks (see [Working with AWS DMS tasks](#) in the AWS DMS documentation)
- Familiarity with AWS Key Management Service (AWS KMS) for encrypting databases (see the [AWS KMS documentation](#))

Limitations

- You can enable encryption for an Amazon RDS DB instance only when you create it, not after the DB instance is created.
- Data in unlogged tables will not be restored using snapshots. For more information, review Best practices for working with PostgreSQL.
- You can't have an encrypted read replica of an unencrypted DB instance or an unencrypted read replica of an encrypted DB instance.
- You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
- AWS DMS does not automatically transfers the Sequences therefore additional steps are required to handle this.

For more information, see Limitations of Amazon RDS encrypted DB instances in the Amazon RDS documentation.

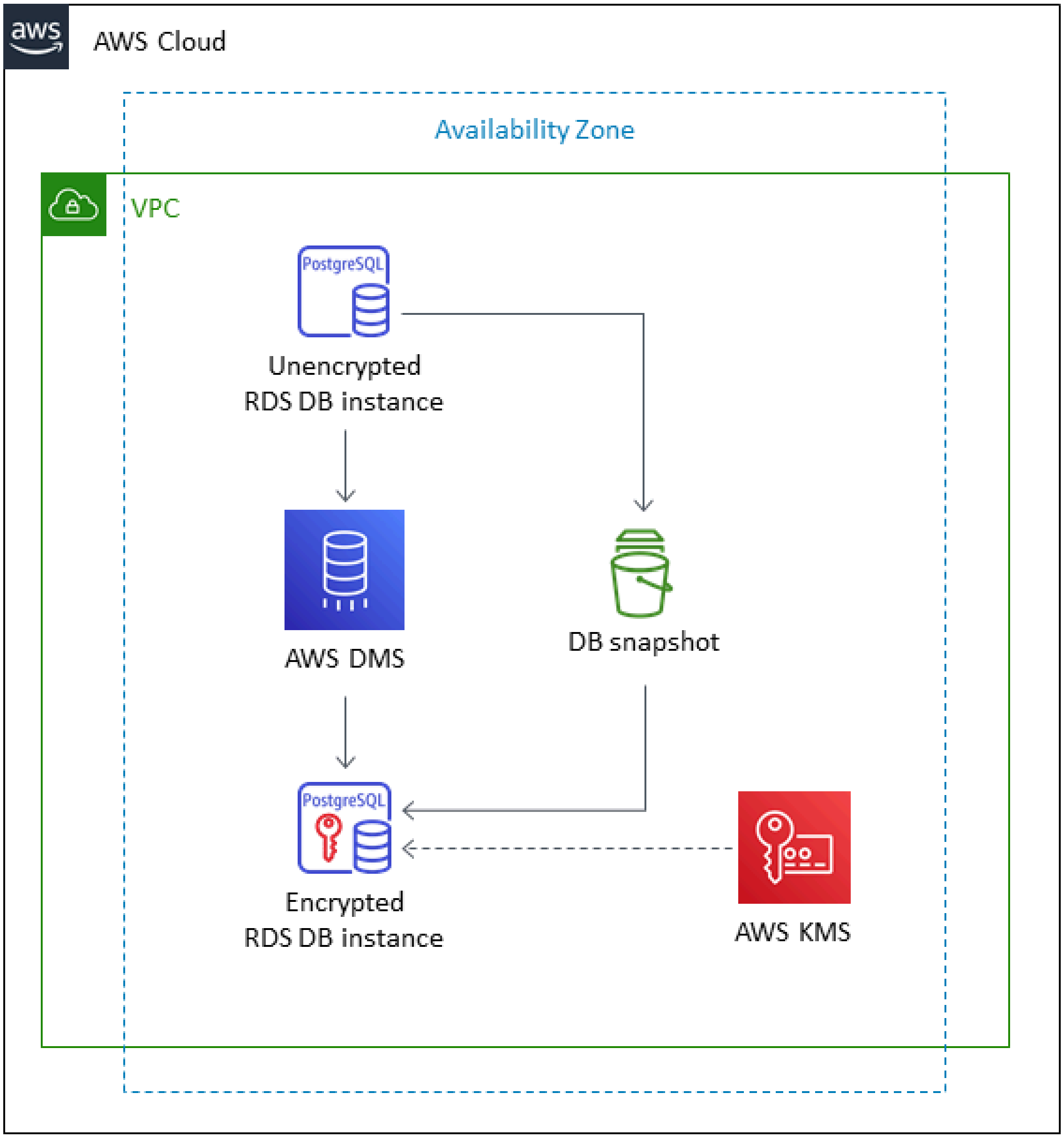
Architecture

Source architecture

- Unencrypted RDS DB instance

Target architecture

- Encrypted RDS DB instance
  - The destination RDS DB instance is created by restoring the DB snapshot copy of the source RDS DB instance.
  - An AWS KMS key is used for encryption while restoring the snapshot.
  - An AWS DMS replication task is used to migrate the data.



Tools

Tools used to enable encryption:

- AWS KMS key for encryption – When you create an encrypted DB instance, you can choose a customer managed key or the AWS managed key for Amazon RDS to encrypt your DB instance. If you don't specify the key identifier for a customer managed key, Amazon RDS uses the AWS managed key for your new DB instance. Amazon RDS creates an AWS managed key for Amazon RDS for your AWS account. Your AWS account has a different AWS managed key for Amazon RDS for each AWS Region. For more information about using KMS keys for Amazon RDS encryption, see [Encrypting Amazon RDS Resources](#).

Tools used for ongoing replication:

- AWS DMS – You can use AWS Database Migration Service (AWS DMS) to replicate changes from the source DB to the target DB. It is important to keep the source and target DB in sync to keep downtime to a minimum. For information about setting up AWS DMS and creating tasks, see the AWS DMS documentation.

Epics

Create a snapshot of the source DB instance and encrypt it

Task	Description	Skills required
Check the details for the source PostgreSQL DB instance.	On the Amazon RDS console, choose the source PostgreSQ L DB instance. On the **Configuration** tab, make sure that encryption isn't enabled for the instance. For a screen illustration, see the Additional information section.	DBA
Create the DB snapshot.	Create a DB snapshot of the instance you want to encrypt. The amount of time it takes to create a snapshot depends on the size of your database. For instructions, see Creating a DB snapshot in the Amazon RDS documentation.	DBA
Encrypt the snapshot.	In the Amazon RDS console navigation pane, choose **Snapshots**, and select the DB snapshot you created. For **Actions**, choose **Copy Snapshot**. Provide the destination AWS Region and	DBA

Task	Description	Skills required
	the name of the DB snapshot copy in the corresponding fields. Select the **Enable Encryption** checkbox. For **Master Key**, specify the KMS key identifier to use to encrypt the DB snapshot copy. Choose **Copy Snapshot**. For more information, see [Copying a snapshot](#) in the Amazon RDS documentation.	

Prepare the target DB instance

Task	Description	Skills required
Restore the DB snapshot.	On the Amazon RDS console, choose **Snapshots**. Choose the encrypted snapshot that you created. For **Actions**, choose **Restore Snapshot**. For **DB Instance Identifier**, provide a unique name for the new DB instance. Review the instance details, and then choose **Restore DB Instance**. A new, encrypted DB Instance will be created from your snapshot. For more informati on, see [Restoring from a DB snapshot](#) in the Amazon RDS documentation.	DBA
Migrate data by using AWS DMS.	On the AWS DMS console, create an AWS DMS task.	DBA

Task	Description	Skills required
	For **Migration type**, choose **Migrate existing data and replicate ongoing changes**. In **Task Settings**, for **Target table preparation mode**, choose **Truncate**. For more information, see Creating a task in the AWS DMS documentation.	
Enable data validation.	In **Task Settings**, choose **Enable validation**. This enables you to compare the source data to the target data to verify that the data was migrated accurately.	DBA
Disable constraints on the target DB instance.	Disable any triggers and foreign key constraints on the target DB instance, and then start the AWS DMS task. For more information about disabling triggers and foreign key constraints, see the AWS DMS documentation.	DBA
Verify data.	After the full load is complete, verify the data on the target DB instance to see if it matches the source data. For more information, see AWS DMS data validation in the AWS DMS documentation.	DBA

Cut over to the target DB instance

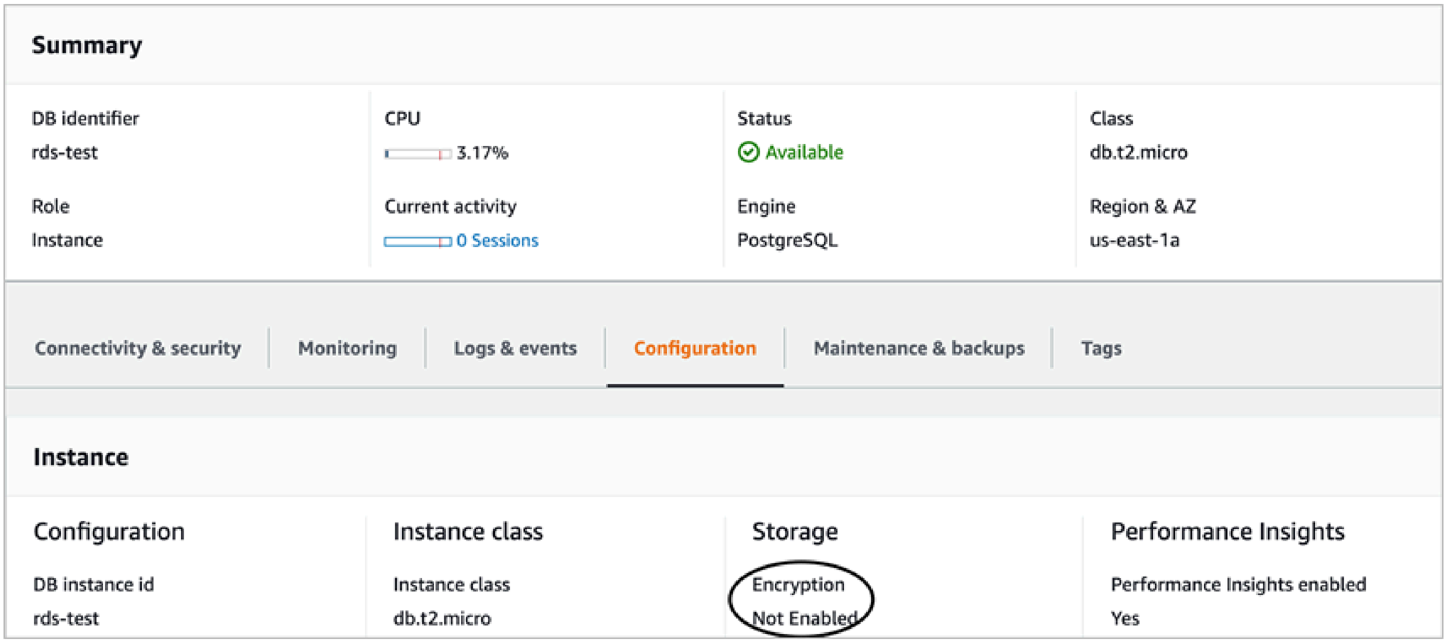
Task	Description	Skills required
Stop write operations on the source DB instance.	Stop the write operations on the source DB instance so that application downtime can begin. Verify that AWS DMS has completed the replication for the data in the pipeline. Enable triggers and foreign keys on the target DB instance.	DBA
Update database sequences	If the source database contains any sequence numbers, verify and update the sequences in the target database.	DBA
Configure the application endpoint.	Configure your application connections to use the new Amazon RDS DB instance endpoints. The DB instance is now encrypted.	DBA, Application owner

Related resources

- Creating an AWS DMS task
- Monitoring replication tasks using Amazon CloudWatch
- Monitoring AWS DMS tasks
- Updating the Amazon RDS encryption key

Additional information

Checking the encryption for the source PostgreSQL DB instance:



Additional notes for this pattern:

- Enable replication on PostgreSQL by setting the `rds.logical_replication` parameter to 1.

Important note: Replication slots retain the write ahead log (WAL) files until the files are externally consumed—for example, by `pg_recvlogical`; by extract, transform, and load (ETL) jobs; or by AWS DMS. When you set the `rds.logical_replication` parameter value to 1, AWS DMS sets the `wal_level`, `max_wal_senders`, `max_replication_slots`, and `max_connections` parameters. If logical replication slots are present but there is no consumer for the WAL files retained by the replication slot, you might see an increase in the transaction log disk usage and a constant decrease in free storage space. For more information and steps to resolve this issue, see the article [How can I identify what is causing the "No space left on device" or "DiskFull" error on Amazon RDS for PostgreSQL?]() in the AWS Support Knowledge Center.

- Any schema changes that you make to the source DB instance after you create the DB snapshot will not be present on the target DB instance.
- After you create an encrypted DB instance, you can't change the KMS key used by that DB instance. Be sure to determine your KMS key requirements before you create your encrypted DB instance.
- You must disable triggers and foreign keys on the target DB instance before you run the AWS DMS task. You can re-enable these when the task is complete.

Enforce automatic tagging of Amazon RDS databases at launch

Susanne Kangnoh and Archit Mathur, Amazon Web Services

Summary

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the Amazon Web Services (AWS) Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

You can use tagging to categorize your AWS resources in different ways. Relational database tagging is useful when you have many resources in your account and you want to quickly identify a specific resource based on the tags. You can use Amazon RDS tags to add custom metadata to your RDS DB instances. A tag consists of a user-defined key and value. We recommend that you create a consistent set of tags to meet your organization's requirements.

This pattern provides an AWS CloudFormation template to help you monitor and tag RDS DB instances. The template creates an Amazon CloudWatch Events event that watches for the AWS CloudTrail **CreateDBInstance** event. (CloudTrail captures API calls for Amazon RDS as events.) When it detects this event, it calls an AWS Lambda function that automatically applies tag keys and values that you define. The template also sends out a notification that the instance has been tagged, by using Amazon Simple Notification Service (Amazon SNS).

Prerequisites and limitations

Prerequisites

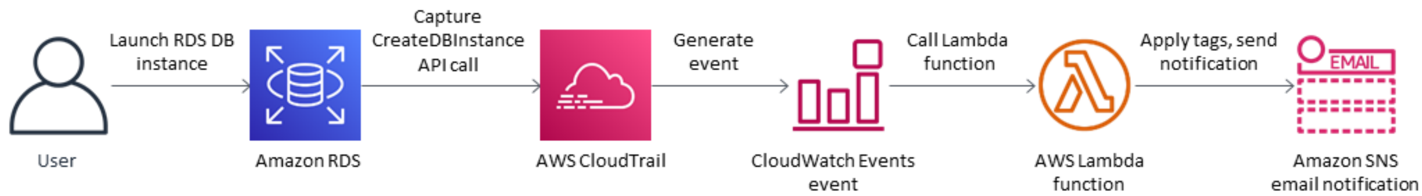
- An active AWS account.
- An Amazon Simple Storage Service (Amazon S3) bucket to upload the Lambda code.
- An email address where you would like to receive tagging notifications.

Limitations

- The solution supports CloudTrail **CreateDBInstance** events. It does not create notifications for any other events.

Architecture

Workflow architecture



Automation and scale

- You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run the template only once in each Region or account.

Tools

AWS services

- AWS CloudTrail – AWS CloudTrail is an AWS service that helps you with governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- AWS Lambda – AWS Lambda is a compute service that supports running code without needing to provision or manage servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud.

Code

This pattern includes an attachment with two files:

- `index.zip` is a compressed file that includes the Lambda code for this pattern.
- `rds.yaml` is a CloudFormation template that deploys the Lambda code.

See the *Epics* section for information about how to use these files.

Epics

Deploy the Lambda code

Task	Description	Skills required
Upload the code to an S3 bucket.	Create a new S3 bucket or use an existing S3 bucket to upload the attached `index.zip` file (Lambda code). This bucket must be in the same AWS Region as the resources (RDS DB instances) that you want to monitor.	Cloud architect
Deploy the CloudFormation template.	Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the `rds.yaml` file that's provided in the attachment. In the next epic, provide values for the template parameters.	Cloud architect

Complete the parameters in the CloudFormation template

Task	Description	Skills required
Provide the S3 bucket name.	Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the RDS DB instances that you want to monitor.	Cloud architect
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `index.zip` or `controls/index.zip`).	Cloud architect
Provide an email address.	Provide an active email address where you want to receive violation notifications.	Cloud architect
Specify a logging level.	Specify the logging level and verbosity. `Info` designates detailed informational messages on the application's progress and should be used only for debugging. `Error` designates error events that could still allow the application to continue running. `Warning` designates	Cloud architect

Task	Description	Skills required
	potentially harmful situation s.	
Enter the tag keys and values for your RDS DB instances.	Enter the required tag keys and values that you want to automatically apply to the RDS instance. For more information, see Tagging Amazon RDS resources in the AWS documentation.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the email subscript ion.	When the CloudFormation template deploys successfu lly, it sends a subscription email message to the email address you provided. To receive notifications when your instances are tagged, you must confirm this email subscription.	Cloud architect

Related resources

- Creating a bucket (Amazon S3 documentation)
- Tagging Amazon RDS resources (Amazon Aurora documentation)
- Uploading objects (Amazon S3 documentation)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail (Amazon CloudWatch documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Estimate the cost of a DynamoDB table for on-demand capacity

Moinul Al-Mamun, Amazon Web Services

Summary

[Amazon DynamoDB](#) is a NoSQL transactional database that provides single-digit millisecond latency even at petabytes scale. This Amazon Web Services (AWS) serverless offering is getting popular because of its consistent performance and scalability. You do not need to provision underlying infrastructure. Your single table can grow up to petabytes.

With on-demand capacity mode, you pay per request for the data reads and writes that your application performs on the tables. AWS charges are based on the accumulated read request units (RRUs) and write request units (WRUs) in a month. DynamoDB monitors the size of your table continuously throughout the month to determine your storage charges. It supports continuous backup with point-in-time-recovery (PITR). DynamoDB monitors the size of your PITR-enabled tables continuously throughout the month to determine your backup charges.

To estimate the DynamoDB cost for a project, it's important to calculate how much RRU, WRU, and storage will be consumed at different stages of your product lifecycle. For a rough cost estimation, you can use [AWS Pricing Calculator](#), but you must provide an approximate number of RRUs, WRUs, and storage requirements for your table. These can be difficult to estimate at the beginning of the project. AWS Pricing Calculator doesn't consider data growth rate or item size, and it doesn't consider the number of reads and writes for the base table and global secondary indexes (GSIs) separately. To use AWS Pricing Calculator, you must estimate all those aspects to assume ballpark figures for WRU, RRU, and storage size to obtain your cost estimation.

This pattern provides a mechanism and a re-usable Microsoft Excel template to estimate basic DynamoDB cost factors, such as write, read, storage, backup and recovery cost, for on-demand capacity mode. It is more granular than AWS Pricing Calculator, and it considers base table and GSIs requirements independently. It also considers the monthly item data growth rate and forecasts costs for three years.

Prerequisites and limitations

Prerequisites

- Basic knowledge of DynamoDB and DynamoDB data model design
- Basic knowledge of DynamoDB pricing, WRU, RRU, storage, and backup and recovery (for more information, see [Pricing for On-Demand Capacity](#))

- Knowledge of your data, data model, and item size in DynamoDB
- Knowledge of DynamoDB GSIs

Limitations

- The template provides you with an approximate calculation, but it isn't appropriate for all configurations. To get a more accurate estimate, you must measure the individual item size for each item in the base table and GSIs.
- For a more accurate estimate, you must consider the expected number of writes (insert, update, and delete) and reads for each item in an average month.
- This pattern supports estimating only write, read, storage, and backup and recovery costs for next few years based on fixed data growth assumptions.

Tools

AWS services

- [Amazon DynamoDB](#) is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.

Other tools

- [AWS Pricing Calculator](#) is a web-based planning tool that you can use to create estimates for your AWS use cases.

Best practices

To help keep costs low, consider the following DynamoDB design best practices.

- Partition key design – Use a high-cardinality partition key to distribute load evenly.
- Adjacency list design pattern – Use this design pattern for managing one-to-many and many-to-many relationships.
- Sparse index – Use sparse index for your GSIs. When you create a GSI, you specify a partition key and optionally a sort key. Only items in the base table that contain a corresponding GSI partition key appear in the sparse index. This helps to keep GSIs smaller.
- Index overloading – Use the same GSI for indexing various types of items.

- GSI write sharding – Shard wisely to distribute data across the partitions for efficient and faster queries.
- Large items – Store only metadata inside the table, save the blob in Amazon S3, and keep the reference in DynamoDB. Break large items into multiple items, and efficiently index by using sort keys.

For more design best practices, see the Amazon DynamoDB Developer Guide.

Epics

Extract item information from your DynamoDB data model

Task	Description	Skills required
Get item size.	1. Check how many different types of item you are going to store in your table. 2. To calculate the size of each item in kilobytes, add the Key and Value size of each attribute. 3. Calculate item size for a base table and for each GSI.	Data engineer
Estimate the write cost.	To estimate write cost in on-demand capacity mode, first you have to measure how many WRUs will be consumed in a month. For that, you need to consider the following factors: • Number of create, update, and delete operations for each item in a month.	Data engineer

Task	Description	Skills required

	• Number of available GSIs. Consider each index independently.
	  ◦ Average size of an index item
	  ◦ Number of synchroni zation times on an index
	• How many new things (for example, components or products) will be added in the table each month? The number of added things could be different every month, but you can assume an average growth rate based on your business cases.
	
	For more information, see the *Additional information* section.	

Task	Description	Skills required
Estimate the read cost.	To estimate read cost in on-demand mode, first you have to measure how many RRUs will be consumed in a month. For that, you need to consider the following factors: • Number of available GSIs. Consider each index independently. ◦ Average size of an index item • Average number of reads per product per month. • Number of total available things (components or products) in the DynamoDB table.	Data engineer, App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Estimate the storage size and cost.</td><td> First, estimate the average monthly storage requirement based on your item size in the table. Then calculate storage cost by multiplying storage size by the per GB storage price for your AWS Region. If you already entered data for estimating the write cost, you don't need to enter it again for calculating storage size. Otherwise, to estimate storage size, you need to consider the following factors: Number of data items in a module (product) based on your table design.
Average item size in kilobytes.
Number of available GSIs. Consider each index independently.
Average size of an index item
How many new products will be added in the table each month? The number of new products could be different every month, but you can assume an average growth rate based</td><td>Data engineer</td></tr>
</table>

Task	Description	Skills required
	on your business cases. This example uses an average of 10 million new products each month.	

Enter the item and object information in the Excel template

Task	Description	Skills required
Download the Excel template from the Attachments section, and adjust it for your use case table.	1. Download the Excel template. 2. Adjust the business module and GSIs, based on your table design.	Data engineer
Enter information in the Excel template.	1. Update item information in the sheet. Update data only in orange cells. 2. Adjust the object numbers: How much could be added into the table each month? 3. Update the WRU and RRU prices per million for your AWS Region. 4. Update the storage and backup prices per GB-month for your AWS Region. 5. Update the recovery price per GB for your AWS Region.	Data engineer

Task	Description	Skills required
	In the template, there are three items, or entities: information, metadata, and relationship. There are two GSIs. For your use case, if you need more items, create new rows. If you need more GSIs, copy an existing GSI block, and paste to create as many GSI blocks as you need. Then adjust the SUM and TOTAL column calculations.	

Related resources

References

- Amazon DynamoDB pricing for on-demand capacity
- AWS Pricing Calculator for DynamoDB
- Best practices for designing and architecting with DynamoDB
- Getting started with DynamoDB

Guides and patterns

- Modeling data with Amazon DynamoDB
- Estimate storage costs for an Amazon DynamoDB table

Additional information

Write cost calculation example

The DynamoDB data model design shows three items for a product, and an average item size of 4 KB. When you add a new product into the DynamoDB base table, it consumes the number of items

* (item size/1 KB write unit) = 3 * (4/1) = 12 WRU. In this example, for writing 1 KB, the product consumes 1 WRU.

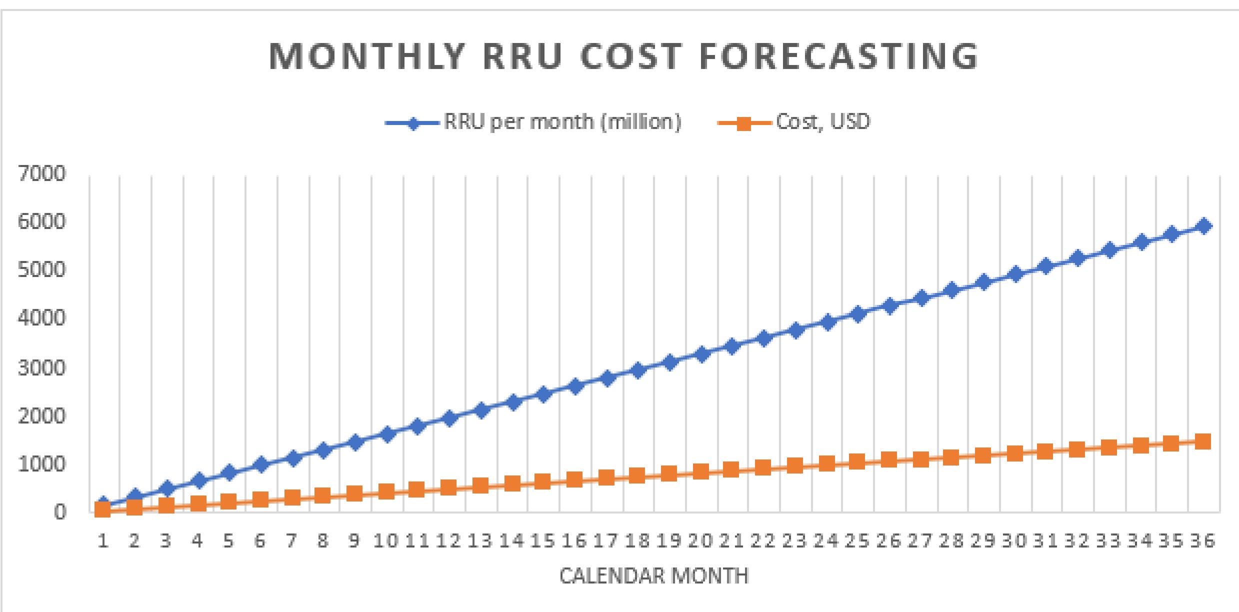
Read cost calculation example

To get the RRU estimation, consider the average of how many times each item will be read in a month. For example, the Information item will be read, on average, 10 times in a month, and the metadata item will be read two times, and the relationship item will be read five times. In the example template, total RRU for all components = number of new components created each month * RRU per component per month = 10 million * 17 RRU = 170 million RRU each month.

Every month, new things (components or products will be added, and the total number of products will grow over time. So, RRU requirements will also grow over time.

- For the first month RRU, consumption will be 170 million.
- For the second month, RRU consumption will be 2 * 170 million = 340 million.
- For the third month RRU consumption will be 3 * 170 million = 510 million.

The following graph shows monthly RRU consumption and cost forecasting.



Note that prices within the graph are for illustration only. To create accurate forecasts for your use case, check the AWS pricing page, and use those prices in the Excel sheet.

Storage, backup, and recovery cost calculation examples

DynamoDB storage, backup and restore all are connected with each other. Backup is directly connected with storage, and recovery is directly connected with backup size. As the table size increases, corresponding storage, backup, and restore costs will increase proportionally.

Storage size and cost

Storage cost will increase over time based on your data growth rate. For example, assume that the average size of a component or product in the base table and GSIs is 11 KB, and 10 million new products will be added every month into your database table. In that case, your DynamoDB table size will grow (11 KB * 10 million)/1024/1024 = 105 GB per month. At the first month, your table storage size will be 105 GB, at second month it will be 105 + 105 = 210 GBs, and so on.

- For the first month, storage cost will be 105 GB * storage price per GB for your AWS Region.
- For the second month, storage cost will be 210 GB * storage price per GB for your Region.
- For the third month, storage cost will be 315 GB * storage price per GB for your Region.

For storage size and cost for next three years, see the *Storage size and forecasting* section.

Backup cost

Backup cost will increase over time based on your data growth rate. When you turn on continuous backup with point-in-time-recovery (PITR), continuous backup charges are based on average storage GB-Month. In a calendar month, the average backup size would be the same as your table storage size, although the actual size could be a bit different. As new products will be added every month, the total storage size and the backup size will grow over time. For example, for the first month, the average backup size of 105 GB could grow to 210 GB for the second month.

- For the first month, backup cost will be 105 GB-month * continuous backup price per GB for your AWS Region.
- For the second month, backup cost will be 210 GB-month * continuous backup price per GB for your Region.
- For the third month, backup cost will be 315 GB-month * continuous backup price per GB for your Region.
- and, so on

Backup cost is included in the graph in the *Storage size and cost forecasting* section.

Recovery cost

When you are taking continuous backup with PITR enabled, recovery operation charges are based on the size of the restore. Each time that you restore, you pay based on gigabytes of restored data. If your table size is large and you perform restore multiple times in a month, it will be costly.

To estimate restore cost, this example assumes that you perform a PITR recovery one time each month at the end of the month. The example uses the monthly average backup size as the restore data size for that month. For the first month, the average backup size is 105 GB, and for the recovery at end of the month, the restore data size would be 105 GB. For the second month, it would be 210 GBs, and so on.

Recovery cost will increase over time based on your data growth rate.

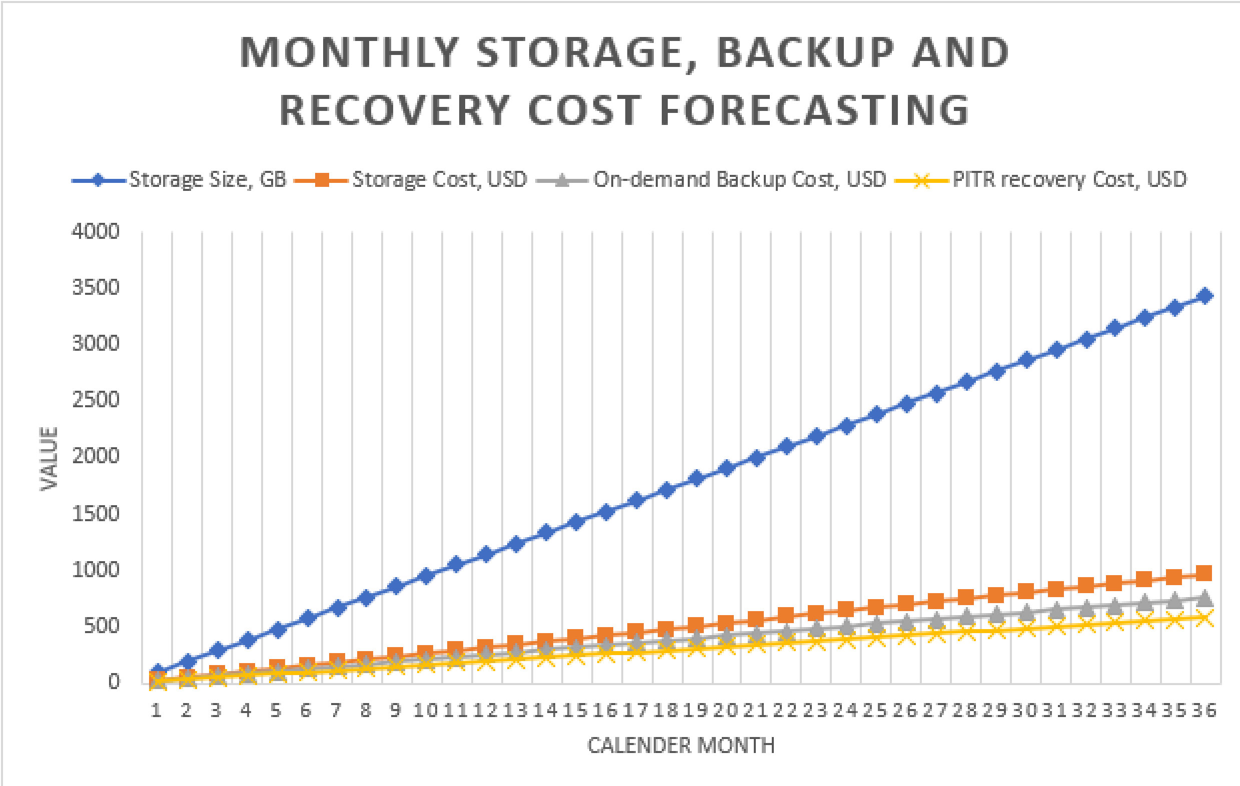
- For the first month, recovery cost will be 105 GB * restore price per GB for your AWS Region.
- For the second month, recovery cost will be 210 GB * restore price per GB for your Region.
- For the third month, recovery cost will be 315 GB * restore price per GB for your Region.

For more information, see the Storage, backup and recovery tab in the Excel template and the graph in the following section.

Storage size and cost forecasting

In the template, actual billable storage size is calculated by subtracting the free tier 25 GB per month for the Standard table class. In the sheet, you will get a forecasting graph broken out into monthly values.

The following example chart forecasts monthly storage size in GB, billable storage cost, on-demand backup cost, and recovery cost for next 36 calendar months. All costs are in USD. From the graph, it's clear that storage, backup, and recovery costs increase proportionally to increases in storage size.



Note that prices used in the graph are for illustration purposes only. To create accurate prices for your use case, check the AWS pricing page, and use those prices in the Excel template.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Estimate storage costs for an Amazon DynamoDB table

Moinul Al-Mamun, Amazon Web Services

Summary

Amazon DynamoDB is a NoSQL transactional database that provides single-digit millisecond latency even at petabytes scale. This popular serverless offering from AWS provides consistent performance and scalability. You do not need to provision storage, and your single table can grow up to petabytes.

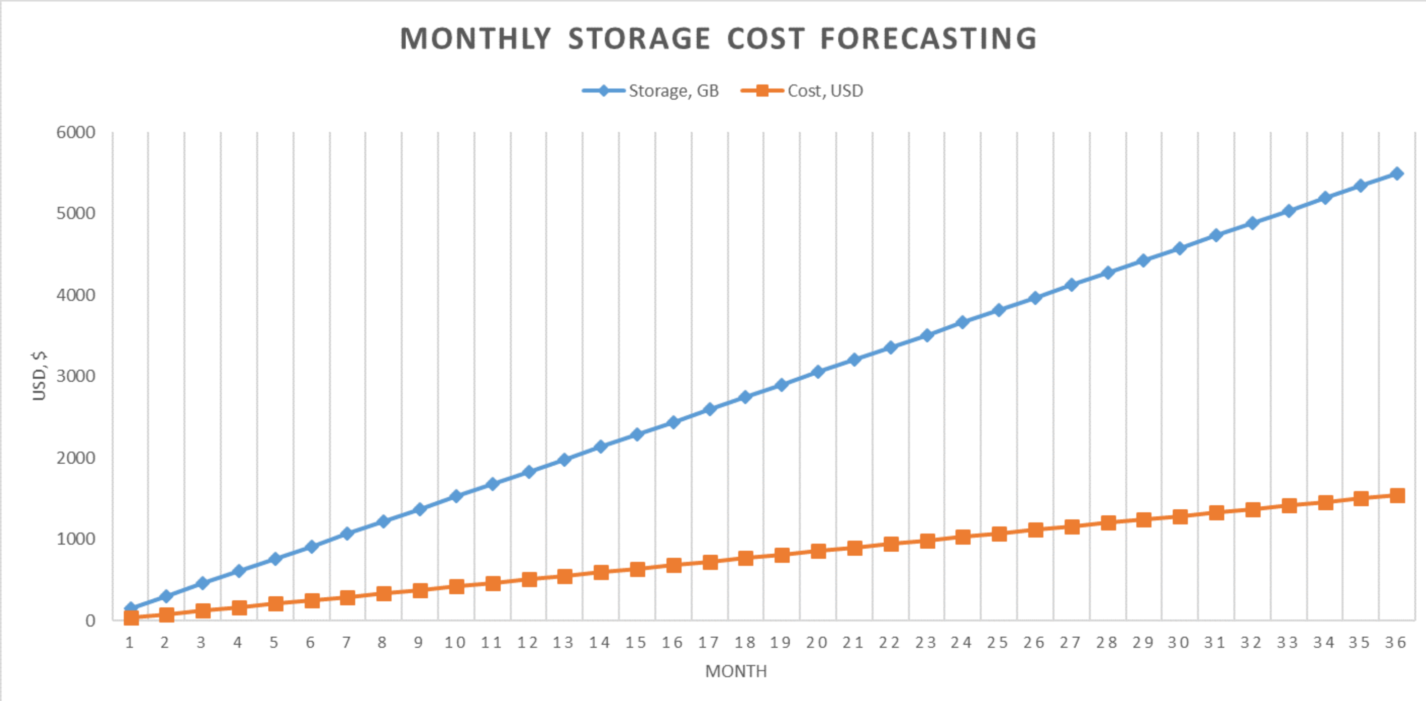
DynamoDB monitors the size of your table continuously throughout the month to determine your storage charges. AWS then charges you for the average size of storage in gigabytes. The more that your table grows over time, the more that your storage cost will grow. To calculate storage cost, you can use AWS Pricing Calculator, but you need to provide the approximate size of your table, including global secondary indexes (GSIs), which is really difficult to estimate at the beginning of the project. Also, AWS Pricing Calculator does not consider the data growth rate.

This pattern provides a mechanism and a reusable Microsoft Excel template to calculate DynamoDB storage size and cost. It considers the storage requirements for the base table and the GSIs independently. It calculates storage size by considering the size of your individual items and data growth rate over time.

To get an estimate, insert two pieces of information into the template:

- The individual item size in kilobytes for the base table and GSIs
- How many new objects or products could be added to the table, on average, in a month, (for example, 10 million)

The template generates a storage and cost forecasting graph for the next three years, which is shown in the following example.



Prerequisites and limitations

Prerequisites

- Basic knowledge of DynamoDB, including DynamoDB storage and pricing
- Knowledge of your data, data model, and item size in DynamoDB
- Knowledge of DynamoDB global secondary indexes (GSIs)

Limitations

- The template provides you with an approximate calculation, but it isn't appropriate for all configurations. To get a more accurate estimate, you must measure the individual item size for each item in the base table and GSIs.
- This pattern supports estimating only storage size and cost for the next few years based on fixed data growth assumptions.

Tools

AWS services

- [Amazon DynamoDB] is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.

Other tools

- [AWS Pricing Calculator] is a web-based planning tool that you can use to create estimates for your AWS use cases.

Epics

Extract item information from your DynamoDB data model

Task	Description	Skills required
Get item size.	1. Determine how many different item types you are going to store in your table. 2. To calculate the size of each item in kilobytes, add the `Key` and `Value` size of each attribute. 3. Calculate item size for a base table and for each GSI.	Data engineer
Get the number of objects added in a month.	Estimate how many components, or objects, will be added into the DynamoDB table, on average, in one month.	Data engineer

Enter the item and object information in the Excel template

Task	Description	Skills required
Download and adjust the Excel spreadsheet.	1. Download the Excel template from the attached document. 2. Adjust the business module and GSIs, based on your table design.	Data engineer
Enter information in the Excel template.	1. Update the item informati on in the sheet. 2. Adjust the object numbers: How much could be added into the table each month? 3. Update the storage price per GB-month for your AWS Region.	Data engineer

Related resources

- Pricing for On-Demand Capacity for Amazon DynamoDB
- AWS Pricing Calculator for DynamoDB

Additional information

Note that the attached template forecasts only storage size and cost for Standard storage table class. Based on the forecast for storage costs, and considering individual item size and product or object growth rate, you can estimate the following:

- Data export cost
- Backup and recovery cost
- Data storage requirements.

Amazon DynamoDB data storage cost

DynamoDB monitors the size of your tables continuously to determine your storage charges. DynamoDB measures the size of your billable data by adding the raw byte size of your data plus a per-item storage overhead that depends on the features you have enabled. For more information, see the DynamoDB Developer Guide.

The price for data storage depends on your table class. The first 25 GB stored each month is free if you're using the DynamoDB Standard table class. For more information about storage costs for the Standard and Standard-Infrequent Access table classes in different AWS Regions, see Pricing for On-Demand Capacity.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Estimate the Amazon RDS engine size for an Oracle database by using AWR reports

Abhishek Verma and Eduardo Valentim, Amazon Web Services

Summary

When you migrate an Oracle database to Amazon Relational Database Service (Amazon RDS) or Amazon Aurora, computing the CPU, memory, and disk I/O for the target database is a key requirement. You can estimate the required capacity of the target database by analyzing the Oracle Automatic Workload Repository (AWR) reports. This pattern explains how to use AWR reports to estimate these values.

The source Oracle database could be on premises or hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance, or it could be an Amazon RDS for Oracle DB instance. The target database could be any Amazon RDS or Aurora database.

> **Note**
>
> Capacity estimates will be more precise if your target database engine is Oracle. For other Amazon RDS databases, the engine size can vary due to differences in database architecture.

We recommend that you run the performance test before you migrate your Oracle database.

Prerequisites and limitations

Prerequisites

- An Oracle Database Enterprise Edition license and Oracle Diagnostics Pack license in order to download AWR reports.

Product versions

- All Oracle Database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c,19c.
- This pattern doesn't cover Oracle Engineered Systems or Oracle Cloud Infrastructure (OCI).

Architecture

Source technology stack

One of the following:

- An on-premises Oracle database
- An Oracle database on an EC2 instance
- An Amazon RDS for Oracle DB instance

Target technology stack

- Any Amazon RDS or Amazon Aurora database

Target architecture

For information about the full migration process, see the pattern Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT.

Automation and scale

If you have multiple Oracle databases to migrate and you want to use additional performance metrics, you can automate the process by following the steps described in the blog post Right-size Amazon RDS instances at scale based on Oracle performance metrics.

Tools

- Oracle Automatic Workload Repository (AWR) is a repository that's built into Oracle databases. It periodically gathers and stores system activity and workload data, which is then analyzed by Automatic Database Diagnostic Monitor (ADDM). AWR takes snapshots of system performance data periodically (by default, every 60 minutes) and stores the information (by default, up to 8 days). You can use AWR views and reports to analyze this data.

Best practices

- To calculate resource requirements for your target database, you can use a single AWR report, multiple AWR reports, or dynamic AWR views. We recommend that you use multiple AWR reports during the peak load period to estimate the resources required to handle those peak loads. In addition, dynamic views provide more data points that help you calculate resource requirements more precisely.

- You should estimate IOPS only for the database that you plan to migrate, not for other databases and processes that use the disk.
- To calculate how much I/O is being used by the database, don't use the information in the Load Profile section of the AWR report. Use the I/O Profile section instead, if it's available, or skip to the Instance Activity Stats section and look at the total values for physical read and write operations.
- When you estimate CPU utilization, we recommend that you use the database metrics method instead of operating system (OS) statistics, because it's based on the CPU used only by databases. (OS statistics also include CPU usage by other processes.) You should also check CPU-related recommendations in the ADDM report to improve performance after migration.
- Consider I/O throughput limits—Amazon Elastic Block Store (Amazon EBS) throughput and network throughput—for the specific instance size when you're determining the right instance type.
- Run the performance test before migration to validate the engine size.

Epics

Create an AWR report

Task	Description	Skills required
Enable the AWR report.	To enable the report, follow the instructions in the Oracle documentation.	DBA
Check the retention period.	To check the retention period of the AWR report, use the following query. `SQL> SELECT snap_interval,retention FROM dba_hist_wr_control;`	DBA

Task	Description	Skills required
Generate the snapshot.	If the AWR snapshot interval isn't granular enough to capture the spike of the peak workload, you can generate the AWR report manually. To generate the manual AWR snapshot, use the following query. `SQL> EXEC dbms_workload_repository.create_snapshot;`	DBA
Check recent snapshots.	To check recent AWR snapshots, use the following query. `SQL> SELECT snap_id, to_char(begin_interval_time,'dd/MON/yy hh24:mi') Begin_Interval, to_char(end_interval_time,'dd/MON/yy hh24:mi') End_Interval FROM dba_hist_snapshot ORDER BY 1;`	DBA

Estimate disk I/O requirements

Task	Description	Skills required
Choose a method.	IOPS is the standard measure of input and output operations per second on a storage	DBA

Task	Description	Skills required
	device, and includes both read and write operations. If you are migrating an on-premises database to AWS, you need to determine the peak disk I/O used by the database. You can use the following methods to estimate disk I/O for your target database: • Load Profile section of the AWR report • Instance Activity Stats section of the AWR report (use this section for Oracle Database 12c or later) • I/O Profile section of the AWR report (use this section for Oracle Database versions before 12c) • AWR views The following steps describe these four methods.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Option 1: Use the load profile.</td><td>The following table shows an example of the Load Profile section of the AWR report.

Important
For more accurate information, we recommend that you use option 2 (I/O profiles) or option 3 (instance activity statistics) instead of the load profile.

<table>
<tr><th></th><th>Per Seco</th><th>Per Trans on</th><th>Per Exec</th><th>Per Call</th></tr>
<tr><td>DB Time</td><td>26.6</td><td>0.2</td><td>0.00</td><td>0.02</td></tr>
<tr><td>DB CPU(</td><td>18.0</td><td>0.1</td><td>0.00</td><td>0.01</td></tr>
<tr><td>Backd CPU(</td><td>0.2</td><td>0.0</td><td>0.00</td><td>0.00</td></tr>
<tr><td>Redo size (byte</td><td>2,45 .9</td><td>17,0</td><td></td><td></td></tr>
<tr><td>Logi read</td><td>3,37 .5</td><td>23,4</td><td></td><td></td></tr>
</table>
</td><td>DBA</td></tr>
</table>

Task	Description	Skills required
	(bloc Bloc 21,6 150. chan Phys 13,5 94.4 read (bloc Phys 3,46 24.1 write (bloc Reac 3,58 24.9 IO requ Writ 574. 4.0 IO requ Reac 106. 0.7 IO (MB) Writ 27.1 0.2 IO (MB) IM 0.0 0.0 scan rows Sessi Logi Reac IM:	

Task	Description	Skills required
	<table><tr><td>**User calls**</td><td>1,24</td><td>8.7</td></tr><tr><td>**Pars (SQL**</td><td>4,62</td><td>32.2</td></tr><tr><td>**Hard pars (SQL**</td><td>8.9</td><td>0.1</td></tr><tr><td>**SQL Worl Area (MB)**</td><td>824.</td><td>5.7</td></tr><tr><td>**Logo**</td><td>1.7</td><td>0.0</td></tr><tr><td>**Exec (SQL**</td><td>136,</td><td>950.</td></tr><tr><td>**Rolll :**</td><td>22.9</td><td>0.2</td></tr><tr><td>**Tran ons:**</td><td>143.</td><td></td></tr></table> Based on this information, you can calculate IOPs and throughput as follows: *IOPS = Read I/O requests: + Write I/O requests = 3,586.8 + 574.7 = 4134.5* *Throughput = Physical read (blocks) + Physical write*	

Task	Description	Skills required
	(blocks) = 13,575.1 + 3,467.3 = 17,042.4 Because the block size in Oracle is 8 KB, you can calculate total throughput as follows: *Total throughput in MB is 17042.4 * 8 * 1024 / 1024 / 1024 = 133.2 MB* **Warning** Don't use the load profile to estimate the instance size. It isn't as precise as instance activity statistics or I/O profiles.	

Task	Description	Skills required
Option 2: Use instance activity statistics.	If you're using an Oracle Database version before 12c, you can use the Instance Activity Stats section of the AWR report to estimate IOPS and throughput. The following table shows an example of this section. <table><tr><td>Statisti</td><td>Total</td><td>per Secon</td><td>per Trans</td></tr><tr><td>**physic read total IO reque**</td><td>2,547, ,217</td><td>3,610.</td><td>25.11</td></tr><tr><td>**physic read total bytes**</td><td>80,77(6,124,</td><td>114,48 26.26</td><td>796,149 8</td></tr><tr><td>**physic write total IO reque**</td><td>534,19 08</td><td>757.1'</td><td>5.27</td></tr><tr><td>**physic write total bytes**</td><td>25,51: 8,849,</td><td>36,16! 1.84</td><td>251,508 8</td></tr></table>	DBA

Task	Description	Skills required
	Based on this information, you can calculate total IOPS and throughput as follows: *Total IOPS = 3,610.28 + 757.11 = 4367* *Total Mbps = 114,482,426.26 + 36,165,631.84 = 150648058.1 / 1024 / 1024 = 143 Mbps*	

Task	Description	Skills required
Option 3: Use I/O profiles.	In Oracle Database 12c, the AWR report includes an I/O Profiles section that presents all the information in a single table and provides more accurate data about database performance. The following table shows an example of this section.	DBA

	Read +Write Per Secon	Read per Secon	Write Per Second
Total Reque	4,367.	3,610.	757.1
Datab Reque	4,161.	3,586.	574.7
Optin Reque	0.0	0.0	0.0
Redo Reque	179.3	2.8	176.6
Total (MB):	143.7	109.2	34.5
Datab (MB):	133.1	106.1	27.1

Task	Description	Skills required
	(continued, see below)	

Optim Total (MB):	0.0	0.0	0.0
Redo (MB):	7.6	2.7	4.9
Datab (block	17,04	13,57	3,467.3
Via Buffer Cache (block	5,898.	5,360.	537.6
Direct (block	11,14	8,214.	2,929.7

This table provides the following values for throughput and total IOPS:

Throughput = 143 MBPS (from the fifth row, labeled Total, second column)

IOPS = 4,367.4 (from the first row, labeled Total Requests, second column)

Task	Description	Skills required
Option 4: Use AWR views.	You can see the same IOPS and throughput informati on by using AWR views. To get this information, use the following query: `break on report compute sum of Value on report select METRIC_NAME,avg(AVERAGE) as "Value" from dba_hist_sysmetric_summary where METRIC_NAME in ('Physical Read Total IO Requests Per Sec','Physical Write Total IO Requests Per Sec') group by metric_name;`	DBA

Estimate CPU requirements

Task	Description	Skills required
Choose a method.	You can estimate the CPU required for the target database in three ways: • By using the actual available cores of the processor • By using the utilized cores based on OS statistics	DBA

Task	Description	Skills required
	• By using the utilized cores based on database statistics If you're looking at utilized cores, we recommend that you use the database metrics method instead of OS statistics, because it's based on the CPU used only by the databases that you're planning to migrate. (OS statistics also include CPU usage by other processes.) You should also check CPU-related recommendations in the ADDM report to improve performance after migration. You can also estimate requirements based on CPU generation. If you are using different CPU generations, you can estimate the required CPU of the target database by following the instructions in the whitepaper [Demystifying the Number of vCPUs for Optimal Workload Performance]().	

Task	Description	Skills required

Option 1: Estimate requirements based on available cores.	In AWR reports:
	
	• CPUs refer to logical and virtual CPUs.
	• Cores are the number of processors in a physical CPU chipset.
	• A socket is a physical device that connects a chip to a board. Multi-core processors have sockets with several CPU cores.
	
	You can estimate available cores in two ways:
	
	• By using OS commands
	• By using the AWR report
	
	To estimate available cores by using OS commands
	
	Use the following command to count the cores in the processor.
	
	`$ cat /proc/cpuinfo | grep "cpu cores"|uniq`
	`cpu cores    : 4`
	`cat /proc/cpuinfo | egrep "core id|physical id" | tr -d "\n" | sed s/physical/\\nphysical/g | grep -v ^$ | sort | uniq | wc -l`
		DBA

Task	Description	Skills required

	Use the following command to count the sockets in the processor.
	`grep "physical id" /proc/cpuinfo | sort -u`
	`  physical id     : 0`
	`  physical id     : 1`
	Note
	We don't recommend using OS commands such as **nmon** and **sar** to extract CPU utilization. This is because those calculations include CPU utilization by other processes and might not reflect the actual CPU that is used by the database.
	To estimate available cores by using the AWR report
	You can also derive CPU utilization from the first section of the AWR report. Here's an excerpt from the report.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>
<table>
<tr><th>D N</th><th>DB Id</th><th>Inst</th><th>Inst nun</th><th>Star Tim</th><th>Rele</th><th>RA</th></tr>
<tr><td>X</td><td><DE</td><td>XXX</td><td>1</td><td>05-Sep 23:(</td><td>12.1 0</td><td>NC</td></tr>
</table>
<table>
<tr><th>Hos Nar</th><th>Plat</th><th>CPU</th><th>Cor</th><th>Soc</th><th>Mem (GB)</th></tr>
<tr><td><ho e></td><td>Linu x86 64-bit</td><td>80</td><td>80</td><td>2</td><td>441.7</td></tr>
</table>
In this example, the CPUs count is 80, which indicates that these are logical (virtual) CPUs. You can also see that this configuration has two sockets, one physical processor on each socket (for a total of two physical processors), and 40 cores for each physical processor or socket.</td><td></td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Option 2: Estimate CPU utilization by using OS statistics.</td><td>You can check the OS CPU usage statistics either directly in the OS (using sar or another host OS utility) or by reviewing the IDLE/(IDLE+BUSY) values from the Operating System Statistics section of the AWR report. You can see the seconds of CPU consumed directly from v$osstat. The AWR and Statspack reports also show this data in the Operating System Statistics section.

If there are multiple databases on the same box, they all have the same v$osstat values for BUSY_TIME.

<table>
<tr><th>Statistic</th><th>Value</th><th>End Value</th></tr>
<tr><td>FREE_MIRY_BYTE</td><td>6,810,67,248</td><td>12,280,799,232</td></tr>
<tr><td>INACTIVMEMORTES</td><td>175,627,33,632</td><td>160,380,653,568</td></tr>
<tr><td>SWAP_F_BYTES</td><td>17,145,64,336</td><td>17,145,872,384</td></tr>
</table>
</td><td>DBA</td></tr>
</table>

Task	Description	Skills required
	BUSY_TI 1,305,56 ,937 **IDLE_TII** 4,312,71 ,839 **IOWAIT_ ME** 53,417,1 4 **NICE_TII** 29,815 **SYS_TIM** 148,567, 70 **USER_TI** 1,146,91 ,783 **LOAD** 25 29 **VM_IN_E ES** 593,920 **VM_OUT TES** 327,680 **PHYSICA MEMOR' TES** 474,362, 17,152 **NUM_CP** 80 **NUM_CP ORES** 80 **NUM_CP OCKETS** 2	

Task	Description	Skills required

GLOBAL_ CEIVE_S E_MAX	4,194,30
GLOBAL_ ND_SIZE AX	2,097,15
TCP_REC VE_SIZE_ EFAULT	87,380
TCP_REC VE_SIZE_ AX	6,291,45
TCP_REC VE_SIZE_ IN	4,096
TCP_SEN SIZE_DE ULT	16,384
TCP_SEN SIZE_MA	4,194,30
TCP_SEN SIZE_MI	4,096

If there are no other major CPU consumers in the system, use the following formula to calculate the percentage of CPU utilization:

Task	Description	Skills required
	Utilization = Busy time / Total time *Busy time = requirements = v $osstat.BUSY_TIME* *C = Total time (Busy + Idle)* *C = capacity = v$ostat.B USY_TIME + v$ostat.I DLE_TIME* *Utilization = BUSY_TIME / (BUSY_TIME + IDLE_TIME)* *= -1,305,569,937 / (1,305,569,937 + 4,312,718 ,839)* *= 23% utilized*	

Task	Description	Skills required
Option 3: Estimate CPU utilization by using database metrics.	If multiple databases are running in the system, you can use the database metrics that appears at the beginning of the report. <table><tr><td></td><td>Snap Id</td><td>Snap Time</td><td>Sessi</td><td>Cursor S ession</td></tr><tr><td>**Begi Snap**</td><td>1846</td><td>28-Sep-09:00</td><td>1226</td><td>35.8</td></tr><tr><td>**End Snap**</td><td>1854</td><td>06-Oct-13:00</td><td>1876</td><td>41.1</td></tr><tr><td>**Elap**</td><td></td><td>11,7. (min</td><td></td><td></td></tr><tr><td>**DB Time**</td><td></td><td>312,0 (min</td><td></td><td></td></tr></table> To get CPU utilization metrics, use this formula: *Database CPU usage (% of CPU power available) = CPU time / NUM_CPUS / elapsed time* where CPU usage is described by *CPU time* and represents	DBA

Task	Description	Skills required
	the time spent on CPU, not the time waiting for CPU. This calculation results in: *= 312,625.40 / 11,759.64 /80 = 33% of CPU is being used* *Number of cores (33%) * 80 = 26.4 cores* *Total cores = 26.4 * (120%) = 31.68 cores* You can use the greater of these two values to calculate the CPU utilization of the Amazon RDS or Aurora DB instance. **Note** On IBM AIX, the calculated utilization doesn't match the values from the operating system or the database. These values do match on other operating systems.	

Estimate memory requirements

Task	Description	Skills required
Estimate memory requirements by using memory statistics.	You can use the AWR report to calculate the memory of the source database and match it in the target database. You should also check the performance of the existing database and reduce your memory requirements to save costs, or increase your requirements to improve performance. That requires a detailed analysis of the AWR response time and the service-level agreement (SLA) of the application. Use the sum of Oracle system global area (SGA) and program global area (PGA) usage as the estimated memory utilization for Oracle. Add an extra 20 percent for the OS to determine a target memory size requirement. For Oracle RAC, use the sum of the estimated memory utilization on all RAC nodes and reduce the total memory, because it's stored on common blocks. 1. Check for the metrics in the Instance Efficiency Percentage table. The table uses the following terms:	DBA

Task	Description	Skills required
	• *Buffer Hit %* is the percentage of times a particular block was found in the buffer cache instead of performing a physical I/O. For better performance, target 100 percent . • *Buffer Nowait %* should be close to 100 percent. • *Latch Hit %* should be close to 100 percent. • *% Non-Parse CPU* is the percentage of CPU time spent in non-parsing activities. This value should be close to 100 percent.. **Instance Efficiency Percentages (target 100%)** Buffe Nowa %: 99.99 Redo NoWa %: 100.00 **Buffe Hit %:** 99.84 **In-mem Sort %:** 100.00 **Libra Hit %:** 748.7 **Soft Parse %:** 99.81	

Task	Description	Skills required
	<table><tr><td>**Execu to Parse %:**</td><td>96.61</td><td>**Latch Hit %:**</td><td>100.00</td></tr><tr><td>**Parse CPU to Parse Elaps %:**</td><td>72.73</td><td>**% Non-Parse CPU:**</td><td>99.21</td></tr><tr><td>**Flash Cache Hit %:**</td><td>0.00</td><td></td><td></td></tr></table> In this example, all the metrics look fine, so you can use the SGA and PGA for the existing database as the capacity planning requirement. 2. Check the memory statistic s section and calculate the SGA/PGA. <table><tr><td></td><td>Begin</td><td>End</td></tr><tr><td>**Host Mem (MB):**</td><td>452,387</td><td>452,387.</td></tr></table>	

Task	Description	Skills required
	<table><tr><td>**SGA use (MB):**</td><td>220,544</td><td>220,544.0</td></tr><tr><td>**PGA use (MB):**</td><td>36,874.9</td><td>45,270.0</td></tr></table> *Total instance memory in use = SGA + PGA = 220 GB + 45 GB = 265 GB* Add 20 percent of buffer: *Total instance memory = 1.2 * 265 GB = 318 GB* Because SGA and PGA account for 70 percent of host memory, the total memory requirement is: *Total host memory = 318/0.7 = 464 GB* **Note** When you migrate to Amazon RDS for Oracle, the PGA and SGA are pre-calculated based on a predefined formula. Make sure that the pre-calculated values	

Task	Description	Skills required
	are close to your estimates.	

Determine the DB instance type of the target database

Task	Description	Skills required
Determine the DB instance type based on disk I/O, CPU, and memory estimates.	Based on the estimates in the previous steps, the capacity of the target Amazon RDS or Aurora database should be: • 68 cores of CPU • 143 MBPS of throughput • 4367 IOPS for disk I/O • 464 GB of memory In the target Amazon RDS or Aurora database, you can map these values to the db.r5.16x large instance type, which has a capacity of 32 cores, 512 GB of RAM, and 13,600 Mbps of throughput. For more information, see the AWS blog post Right-size Amazon RDS instances at scale based on Oracle performance metrics.	DBA

Related resources

- Aurora DB instance class (Amazon Aurora documentation)
- Amazon RDS DB instance storage (Amazon RDS documentation)
- AWS Miner tool (GitHub repository)

Export Amazon RDS for SQL Server tables to an S3 bucket by using AWS DMS

Subhani Shaik, Amazon Web Services

Summary

Amazon Relational Database Service (Amazon RDS) for SQL Server doesn’t support loading data onto other DB engine linked servers on the Amazon Web Services (AWS) Cloud. Instead, you can use AWS Database Migration Service (AWS DMS) to export Amazon RDS for SQL Server tables to an Amazon Simple Storage Service (Amazon S3) bucket, where the data is available to other DB engines.

AWS DMS helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.

This pattern uses AWS Secrets Manager while configuring the AWS DMS endpoints. Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. You can use the service to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to Secrets Manager, reducing the need to hardcode sensitive information. Secrets Manager offers secret rotation with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB. Also, the service is extensible to other types of secrets, including API keys and OAuth tokens. With Secrets Manager, you can control access to secrets by using fine-grained permissions and audit secret rotation centrally for resources in the AWS Cloud, third-party services, and on premises.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An S3 bucket
- A virtual private cloud (VPC)
- A DB subnet
- Amazon RDS for SQL Server

- An AWS Identity and Access Management (IAM) role with access (list, get, and put objects) to the S3 bucket on behalf of the Amazon RDS instance.
- Secrets Manager to store the RDS instance credentials.

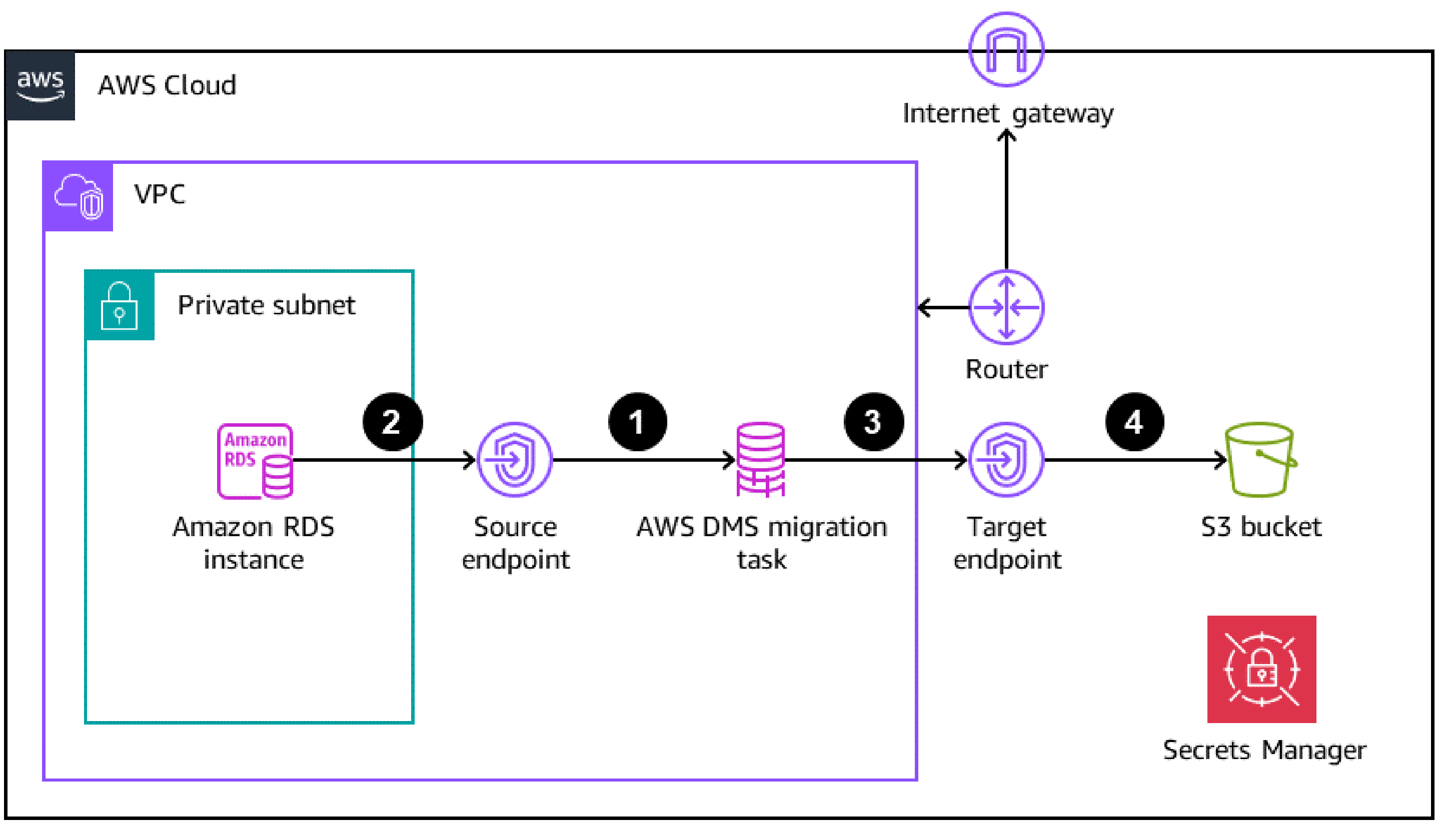
Architecture

Technology stack

- Amazon RDS for SQL Server
- AWS DMS
- Amazon S3
- AWS Secrets Manager

Target architecture

The following diagram shows the architecture for importing data from the Amazon RDS instance to the S3 bucket with the help of AWS DMS.



1. The AWS DMS migration task connecting to the source Amazon RDS instance through the source endpoint
2. Copying data from the source Amazon RDS instance
3. The AWS DMS migration task connecting to the target S3 bucket through the target endpoint
4. Exporting copied data to the S3 bucket in comma-separated values (CSV) format

Tools

AWS services

- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.

Other services

- Microsoft SQL Server Management Studio (SSMS) is a tool for managing SQL Server, including accessing, configuring, and administering SQL Server components.

Epics

Configure the Amazon RDS for SQL Server instance

Task	Description	Skills required
Create the Amazon RDS for SQL Server instance.	1. Open the AWS Managemen t Console, choose **RDS**, and use the **Standard**	DBA, DevOps engineer

Task	Description	Skills required
	create option to create an **Amazon RDS** instance with the required edition, such as SQL Server Express Edition, SQL Server Standard Edition, or SQL Server Enterprise Edition. For the version, choose 2016 or later. 2. Under **Templates**, choose **Dev/Test**.	
Set up credentials for the instance.	1. Enter a name for the instance. 2. Provide a **username** and password for the Amazon RDS instance.	DBA, DevOps engineer

Task	Description	Skills required
Configure the instance class, storage, auto scaling, and availability.	1. Select the **DB instance class** from the list: **Standard**, **Memory Optimized**, and **Burstable** classes. Choose the DB instance type that allocates the computational, network, and memory capacity required by the workloads planned for this DB instance. For more information, see the [AWS documentation](). 2. Select the Storage Type from the list: **General Purpose SSD**, **Provisioned IOPS SSD**, or **Magnetic**. Allocate the default storage size as required. 3. Choose **Enable storage autoscaling** to increase the Amazon RDS storage based on your capacity planning. 4. A Multi-AZ deployment with a replication instance is supported by AWS DMS. In the event of an outage in the Availability Zone, internal hardware, or network, AWS DMS will create a standby instance and provide high availability (HA) through automatic failover to the standby	DBA, DevOps engineer

Task	Description	Skills required
	replicas. Depending on the size of your import, select the appropriate option.	
Specify the VPC, subnet group, public access, and security group.	Select the **VPC**, **DB subnet groups**, and **VPC security group** as required to create the Amazon RDS instance. Follow the best practices, for example: • Do not enable public access to the RDS DB instance. • Do not use the CIDR 0.0.0.0/0 in the security groups. • Use only the required IP address and port details to access the RDS instance.	DBA, DevOps engineer
Configure monitoring, backup, and maintenance.	1. Specify the backup options that you want. By default, automated backups are enabled with a retention period of 7 days. 2. Choose the appropria te **auto minor version upgrade** and **Maintenance window** settings to apply the pending modificat ions or maintenance to the database by Amazon RDS. 3. Choose **Create database**.	DBA, DevOps engineer

Set up the database and example data

Task	Description	Skills required
Create a table and load the example data.	In the new database, create a table. Use the example code in the *Additional information* section to load data into the table.	DBA, DevOps engineer

Set up credentials

Task	Description	Skills required
Create the secret.	1. On the console, choose **Secrets Manager**, and choose **Store a new secret**. 2. Enter a username and password for the Amazon RDS for SQL Server database. This secret will be used for the AWS DMS source endpoint.	DBA, DevOps engineer

Set up access between the database and the S3 bucket

Task	Description	Skills required
Create an IAM role for access to Amazon RDS.	1. On the console, choose **IAM**, and create an IAM role that gives an S3 bucket read/write access to Amazon RDS.	DBA, DevOps engineer

Task	Description	Skills required
	2. Under **Feature**, select **S3 Integration**.	

Create the S3 bucket

Task	Description	Skills required
Create the S3 bucket.	To save the data from Amazon RDS for SQL Server, on the console, choose **S3**, and then choose **Create bucket**. Make sure that the S3 bucket is not publicly available.	DBA, DevOps engineer

Set up access between AWS DMS and the S3 bucket

Task	Description	Skills required
Create an IAM role for AWS DMS to access Amazon S3.	Create an IAM role that allows AWS DMS to list, get, and put objects from the S3 bucket.	DBA, DevOps engineer

Configure AWS DMS

Task	Description	Skills required
Create the AWS DMS source endpoint.	1. On the console, choose **Database Migration Service**, and choose **Endpoints**. Create the **Source endpoint**, selecting	DBA, DevOps engineer

Task	Description	Skills required
	the **Select RDS DB instance** check box. 2. For the Source engine, select **Microsoft SQL Server**. 3. Under **Access to endpoint database**, choose **AWS Secrets Manager**, and enter the secret and IAM role that you created earlier, and the database name. 4. Test the source endpoint.	
Create the AWS DMS target endpoint.	Create the **Target endpoint,** selecting Amazon S3 as the **Target engine**. Provide the S3 bucket name and folder name for the IAM role that you created previously.	DBA, DevOps engineer
Create the AWS DMS replicati on instance.	In the same VPC, subnet, and security group, create the AWS DMS replication instance. For more informati on about choosing an instance class, see the [AWS documentation]().	DBA, DevOps engineer

Task	Description	Skills required
Create the AWS DMS migration task.	To export the data from Amazon RDS for SQL Server to the S3 bucket, create a database migration task. For the migration type, choose **Migrate existing data**. Select the AWS DMS endpoints and replication instance that you created.	DBA, DevOps engineer

Export the data to the S3 bucket

Task	Description	Skills required
Run the database migration task.	To export the SQL Server table data, start the database migration task. The task will export the data from Amazon RDS for SQL Server to the S3 bucket in CSV format.	DBA, DevOps engineer

Clean up resources

Task	Description	Skills required
Delete the resources.	To avoid incurring extra costs, use the console to delete the resources in the following order: 1. Migration task 2. Replication instance 3. Endpoints	DBA, DevOps engineer

Task	Description	Skills required
	4. S3 bucket 5. Database instance	

Related resources

- AWS DMS
- Amazon S3
- Amazon RDS for SQL Server
- Amazon S3 integration

Additional information

To create the database and table, and to load the example data, use the following code.

```
--Step1: Database creation in RDS SQL Server
CREATE DATABASE [Test_DB]
 ON  PRIMARY
( NAME = N'Test_DB', FILENAME = N'D:\rdsdbdata\DATA\Test_DB.mdf' , SIZE = 5120KB ,
 FILEGROWTH = 10%)
 LOG ON
( NAME = N'Test_DB_log', FILENAME = N'D:\rdsdbdata\DATA\Test_DB_log.ldf' , SIZE =
 1024KB , FILEGROWTH = 10%)
GO

--Step2: Create Table
USE Test_DB
GO
Create Table Test_Table(ID int, Company Varchar(30), Location Varchar(20))

--Step3: Load sample data.
USE Test_DB
GO
Insert into Test_Table values(1,'AnyCompany','India')
Insert into Test_Table values(2,'AnyCompany','USA')
Insert into Test_Table values(3,'AnyCompany','UK')
Insert into Test_Table values(4,'AnyCompany','Hyderabad')
Insert into Test_Table values(5,'AnyCompany','Banglore')
```

Handle anonymous blocks in Dynamic SQL statements in Aurora PostgreSQL

anuradha chintha, Amazon Web Services

Summary

This pattern shows you how to avoid the error that you get when handling anonymous blocks in Dynamic SQL statements. You receive an error message when you use the AWS Schema Conversion Tool to convert an Oracle database to an Aurora PostgreSQL-Compatible Edition database. To avoid the error, you must know the value of an `OUT` bind variable, but you can't know the value of an `OUT` bind variable until after you run the SQL statement. The error results from the AWS Schema Conversion Tool (AWS SCT) not understanding the logic inside the Dynamic SQL statement. AWS SCT can't convert the dynamic SQL statement in PL/SQL code (that is, functions, procedures, and packages).

Prerequisites and limitations

Prerequisites

- Active AWS account
- Aurora PostgreSQL database (DB) instance
- Amazon Relational Database Service (Amazon RDS) for Oracle DB instance
- PostgreSQL interactive terminal (psql)
- SQL *Plus
- `AWS_ORACLE_EXT` schema (part of the AWS SCT extension pack) in your target database
- Latest version of AWS Schema Conversion Tool (AWS SCT) and its required drivers

Architecture

Source technology stack

- On-premises Oracle Database 10g and later version

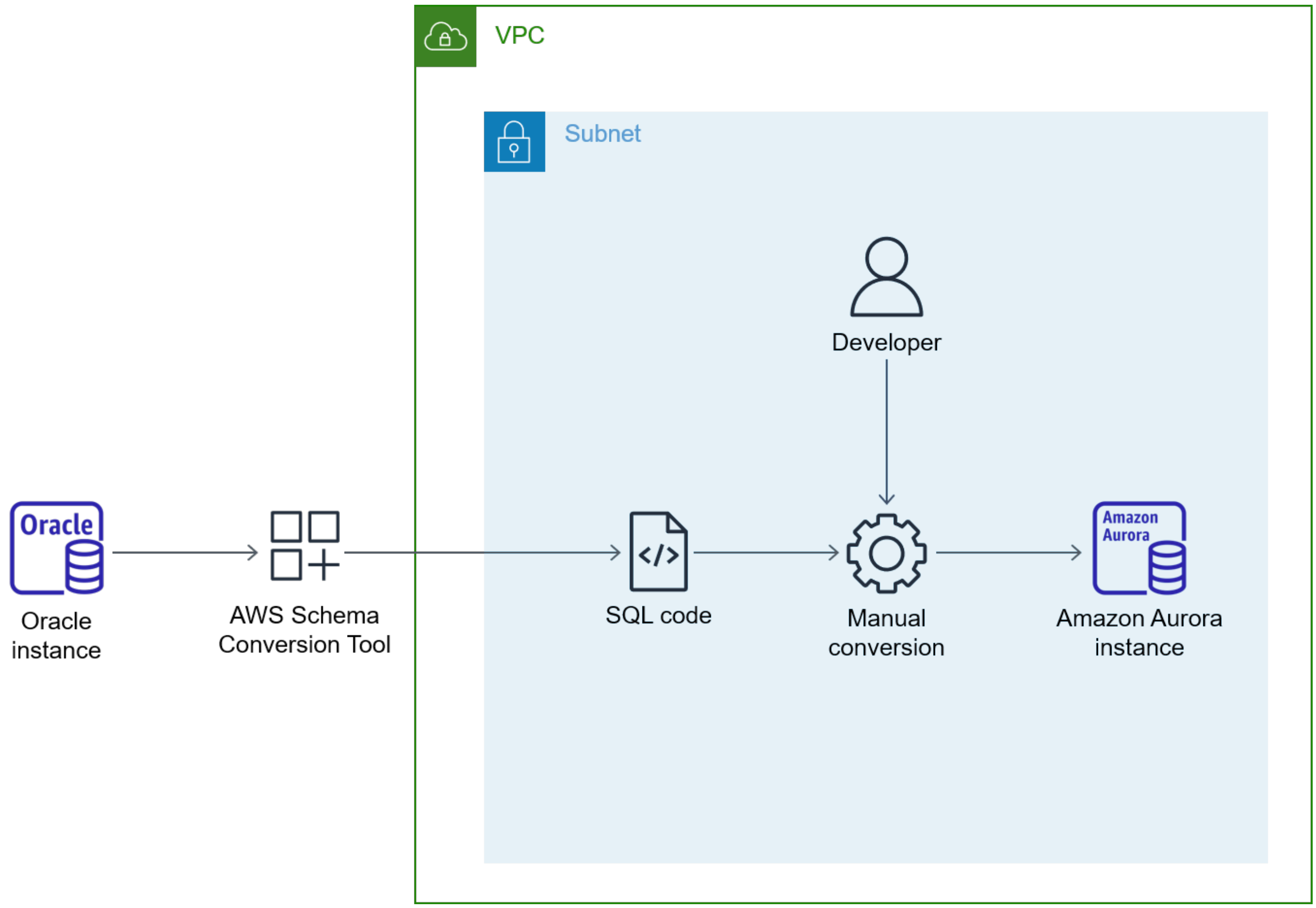
Target technology stack

- Amazon Aurora PostgreSQL

- Amazon RDS for PostgreSQL
- AWS Schema Conversion Tool (AWS SCT)

Migration architecture

The following diagram shows how to use AWS SCT and Oracle OUT bind variables to scan your application code for embedded SQL statements and convert the code to a compatible format that an Aurora database can use.



The diagram shows the following workflow:

1. Generate an AWS SCT report for the source database by using Aurora PostgreSQL as the target database.
2. Identify the anonymous block in the Dynamic SQL code block (for which AWS SCT raised the error).
3. Convert the code block manually and deploy the code on a target database.

Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- Amazon Relational Database Service (Amazon RDS) for Oracle helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.
- AWS Schema Conversion Tool (AWS SCT) helps you make heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects to a format compatible with the target database.

Other tools

- pgAdmin enables you to connect to and interact with your database server.
- Oracle SQL Developer is an integrated development environment that you can use to develop and manage databases in Oracle Database. You can use either SQL *Plus or Oracle SQL Developer for this pattern.

Epics

Configure the Oracle source database

Task	Description	Skills required
Create an Oracle instance on Amazon RDS or Amazon EC2.	To create an Oracle DB instance on Amazon RDS, see Creating an Oracle DB instance and connecting to a database on an Oracle DB instance in the Amazon RDS documentation. To create an Oracle DB instance on Amazon Elastic Compute Cloud (Amazon EC2), see Amazon EC2 for	DBA

Task	Description	Skills required
	Oracle in the AWS Prescriptive Guidance documentation.	
Create a database schema and objects for migration.	You can use Amazon Cloud Directory to create a database schema. For more information, see Create a Schema in the Cloud Directory documentation.	DBA
Configure inbound and outbound security groups.	To create and configure security groups, see Controlling access with security groups in the Amazon RDS documentation.	DBA
Confirm that the database is running.	To check the status of your database, see Viewing Amazon RDS events in the Amazon RDS documentation.	DBA

Configure the target Aurora PostgreSQL database

Task	Description	Skills required
Create an Aurora PostgreSQL instance in Amazon RDS.	To create an Aurora PostgreSQL instance, see Creating a DB cluster and connecting to a database on an Aurora PostgreSQL DB cluster in the Amazon RDS documentation.	DBA
Configure an inbound and outbound security group.	To create and configure security groups, see Provide	DBA

Task	Description	Skills required
	access to the DB cluster in the VPC by creating a security group in the Aurora documentation.	
Confirm that the Aurora PostgreSQL database is running.	To check the status of your database, see Viewing Amazon RDS events in the Aurora documentation.	DBA

Set up AWS SCT

Task	Description	Skills required
Connect AWS SCT to the source database.	To connect AWS SCT to your source database, see Connecting to PostgreSQL as a source in the AWS SCT documentation.	DBA
Connect AWS SCT to the target database.	To connect AWS SCT to your target database, see the What is the AWS Schema Conversio n Tool? in the AWS Schema Conversion Tool User Guide.	DBA
Convert the database schema in AWS SCT and save the automated converted code as a SQL file.	To save AWS SCT converted files, see Saving and applying your converted schema in AWS SCT in the AWS Schema Conversion Tool User Guide.	DBA

Migrate the code

Task	Description	Skills required
Get the SQL file for manual conversion.	In the AWS SCT converted file, pull the SQL file that requires manual conversion.	DBA
Update the script.	Manually update the SQL file.	DBA

Related resources

- Amazon RDS
- Amazon Aurora Features

Additional information

The following example code shows how to configure the Oracle source database:

```
CREATE or replace PROCEDURE calc_stats_new1 (
  a NUMBER,
  b NUMBER,
  result out NUMBER)
IS
BEGIN
result:=a+b;
END;
/
```

```
set serveroutput on ;

DECLARE
  a NUMBER := 4;
  b NUMBER := 7;
  plsql_block VARCHAR2(100);
  output number;
BEGIN
  plsql_block := 'BEGIN calc_stats_new1(:a, :b,:output); END;';
  EXECUTE IMMEDIATE plsql_block USING a, b,out output;
```

```
  DBMS_OUTPUT.PUT_LINE('output:'||output);

END;
```

The following example code shows how to configure the target Aurora PostgreSQL database:

```
 w integer,
 x integer)
RETURNS integer
AS
$BODY$
DECLARE
begin
return w + x ;
end;
$BODY$
LANGUAGE  plpgsql;


CREATE OR REPLACE FUNCTION test_pg.init()
RETURNS void
AS
$BODY$
BEGIN
if aws_oracle_ext.is_package_initialized
      ('test_pg' ) then
      return;
    end if;
    perform aws_oracle_ext.set_package_initialized
      ('test_pg' );

PERFORM aws_oracle_ext.set_package_variable('test_pg', 'v_output', NULL::INTEGER);
PERFORM aws_oracle_ext.set_package_variable('test_pg', 'v_status', NULL::text);
END;
$BODY$
LANGUAGE  plpgsql;


DO $$
declare
v_sql text;
v_output_loc int;
a integer :=1;
```

```
b integer :=2;
BEGIN
perform  test_pg.init();
--raise notice 'v_sql %',v_sql;
execute 'do $a$ declare v_output_l int; begin select * from test_pg.calc_stats_new1('||
a||','||b||') into v_output_l;
PERFORM aws_oracle_ext.set_package_variable(''test_pg'', ''v_output'', v_output_l) ;
 end; $a$'  ;
v_output_loc := aws_oracle_ext.get_package_variable('test_pg', 'v_output');
raise notice 'v_output_loc %',v_output_loc;
END ;
$$
```

Handle overloaded Oracle functions in Aurora PostgreSQL-Compatible

Sumana Yanamandra, Amazon Web Services

Summary

The code you migrate from an on-premises Oracle database to Amazon Aurora PostgreSQL-Compatible Edition might include overloaded functions. These functions have the same definition —that is, the same function name and the same number and data type of input (`IN`) parameters— but the data type or the number of output (`OUT`) parameters might differ.

These parameter mismatches can cause problems in PostgreSQL, because it's difficult to determine which function to run. This pattern illustrates how to handle overloaded functions when you migrate your database code to Aurora PostgreSQL-Compatible.

Prerequisites and limitations

Prerequisites

- An Oracle database instance as your source database
- An Aurora PostgreSQL-Compatible DB instance as your target database (see instructions in the Aurora documentation)

Product versions

- Oracle Database 9i or later
- Oracle SQL Developer version 18.4.0.376
- pgAdmin 4 client
- Aurora PostgreSQL-Compatible version 11 or later (see Identifying versions of Amazon Aurora PostgreSQL in the Aurora documentation)

Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.

Other tools

- [Oracle SQL Developer](#) is a free, integrated development environment for working with SQL in Oracle databases in both traditional and cloud deployments.
- [pgAdmin](#) is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.

Epics

Create a simple function

Task	Description	Skills required
Create a function in PostgreSQL that has one input parameter and one output parameter.	The following example illustrates a function named `test_overloading` in Aurora PostgreSQL-Compatible. This function has two parameters: one input text parameter and one output text parameter. ```CREATE OR REPLACE FUNCTION public.test_overloading(str1 text, OUT str2 text) LANGUAGE 'plpgsql' COST 100 VOLATILE AS $BODY$ DECLARE BEGIN str2 := 'Success'; RETURN ; EXCEPTION WHEN others THEN RETURN ; END;```	Data engineer, Aurora PostgreSQL-Compatible

Task	Description	Skills required
	`$BODY$;`	

Run the function in PostgreSQL.	Run the function that you created in the previous step.
	`select public.test_overloading('Test');`
	It should display the following output.
	`Success`
		Data engineer, Aurora PostgreSQL-Compatible

Overload the function

Task	Description	Skills required

Use the same function name to create an overloaded function in PostgreSQL.	Create an overloaded function in Aurora PostgreSQL-Compatible that uses the same function name as your previous function. The following example is also named `test_overloading` , but it has three parameters: one input text parameter, one output text parameter, and one output integer parameter.
	`CREATE OR REPLACE FUNCTION public.test_overloading(`
	`str1 text,`
		Data engineer, Aurora PostgreSQL-Compatible

Task	Description	Skills required

	<pre> OUT str2 text,
	        OUT num1
	 integer)
	    LANGUAGE 'plpgsql'
	
	    COST 100
	    VOLATILE
	AS $BODY$
	DECLARE
	str3 text;
	
	BEGIN
	
	        str2 :=
	 'Success';
	        num1 := 100;
	
	    RETURN ;
	    EXCEPTION
	        WHEN others THEN
	            RETURN ;
	END;
	$BODY$;</pre>	

Task	Description	Skills required

Run the function in PostgreSQL.	When you run this function, it fails with the following error message.
	`ERROR: cannot change return type of existing function`
	`HINT: Use DROP FUNCTION test_overloading(text) first.`
	This happens because Aurora PostgreSQL-Compatible doesn't support function overloading directly. It can't identify which function to run, because the number of output parameters is different in the second version of the function, although the input parameters are the same.
		Data engineer, Aurora PostgreSQL-Compatible

Apply the workaround

Task	Description	Skills required

Add INOUT to the first output parameter.	As a workaround, modify the function code by representing the first output parameter as `INOUT`.
	`CREATE OR REPLACE FUNCTION public.test_overloading(`
	`str1 text,`
		Data engineer, Aurora PostgreSQL-Compatible

Task	Description	Skills required

	<pre> INOUT str2
	 text,
	        OUT num1
	 integer)
	    LANGUAGE 'plpgsql'
	
	    COST 100
	    VOLATILE
	AS $BODY$
	DECLARE
	str3 text;
	BEGIN
	
	        str2 :=
	 'Success';
	        num1 := 100;
	
	    RETURN ;
	    EXCEPTION
	        WHEN others THEN
	            RETURN ;
	END;
	$BODY$;</pre>	

Task	Description	Skills required

Run the revised function.	Run the function that you updated by using the following query. You pass a null value as the second argument of this function, because you declared this parameter as `INOUT` to avoid the error.
	`select public.test_overloading('Test', null);`
	The function is now created successfully.
	`Success, 100`
		Data engineer, Aurora PostgreSQL-Compatible

Validate the results.	Verify that the code with the overloaded function was converted successfully.	Data engineer, Aurora PostgreSQL-Compatible

Related resources

- Working with Amazon Aurora PostgreSQL (Aurora documentation)
- Function overloading in Oracle (Oracle documentation)
- Function overloading in PostgreSQL (PostgreSQL documentation)

Help enforce DynamoDB tagging

Mansi Suratwala, Amazon Web Services

Summary

This pattern sets up automatic notifications when a predefined Amazon DynamoDB tag is missing or removed from a DynamoDB resource on the Amazon Web Services (AWS) Cloud.

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database. When you use DynamoDB, you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.

The pattern uses an AWS CloudFormation template, which creates an Amazon CloudWatch Events event and an AWS Lambda function. The event watches for any new or existing DynamoDB tagging information by using AWS CloudTrail. If a predefined tag is missing or removed, CloudWatch triggers a Lambda function, which sends you an Amazon Simple Notification Service (Amazon SNS) notification informing you of the violation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Simple Storage Service (Amazon S3) bucket for the Lambda .zip file that contains the Python script for running the Lambda function

Limitations

- The solution works only when the `TagResource` or `UntagResource` CloudTrail events occur. It does not create notifications for any other events.

Architecture

Target technology stack

- Amazon DynamoDB
- AWS CloudTrail

- Amazon CloudWatch
- AWS Lambda
- Amazon S3
- Amazon SNS

Target architecture

Automation and scale

You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run the template only once in each Region or account.

Tools

Tools

- Amazon DynamoDB – DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with scalability.
- AWS CloudTrail – CloudTrail is an AWS service that helps you with governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near-real time stream of system events that describe changes in AWS resources.
- AWS Lambda – Lambda is a compute service that supports running code without needing to provision or manage servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

- [Amazon SNS](#) – Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud.

Code

- A .zip file of the project is available as an attachment.

Epics

Define the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	On the Amazon S3 console, choose or create an S3 bucket with a unique name that does not contain leading slashes. This S3 bucket will host the Lambda code .zip file. Your S3 bucket must be in the same AWS Region as the DynamoDB resource that is being monitored.	Cloud Architect

Upload the Lambda code to the S3 bucket

Task	Description	Skills required
Upload the Lambda code to the S3 bucket.	Upload the Lambda code .zip file provided in the *Attachments* section to the S3 bucket. The S3 bucket must be in the same Region as the DynamoDB resource that is being monitored.	Cloud Architect

Deploy the AWS CloudFormation template

Task	Description	Skills required
Deploy the AWS CloudFormation template.	On the AWS CloudFormation console, deploy the AWS CloudFormation template that's provided in the *Attachments* section. In the next epic, provide values for the parameters.	Cloud Architect

Complete the parameters in the AWS CloudFormation template

Task	Description	Skills required
Name the S3 bucket.	Enter the name of the S3 bucket that you created or chose in the first epic.	Cloud Architect
Provide the Amazon S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `<folder>/<file-name>.zip`).	Cloud Architect
Provide an email address	Provide an active email address to receive Amazon SNS notifications.	Cloud Architect
Define the logging level.	Define the logging level and frequency for your Lambda function. `Info` designates detailed informational messages on the application's progress. `Error` designate	Cloud Architect

Task	Description	Skills required
	s error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	
Enter the required DynamoDB tag keys.	Be sure that the tags are separated by commas, with no spaces between them (for example, `ApplicationId,CreatedBy,Environment,Organization`). The CloudWatch Events event searches for these tags and sends a notification if they are not found.	Cloud Architect

Confirm the subscription.

Task	Description	Skills required
Confirm the subscription.	When the template successfully deploys, it sends a subscription email to the email address that you provided. To receive violation notifications, you must confirm this email subscription.	Cloud Architect

Related resources

- [Creating an S3 bucket]

- Uploading files to an S3 bucket
- Tagging resources in DynamoDB
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Implement cross-Region disaster recovery with AWS DMS and Amazon Aurora

Mark Hudson, Amazon Web Services

Summary

Natural or human-induced disasters can occur at any time and can impact the availability of services and workloads running in a given AWS Region. To mitigate the risks, you must develop a disaster recovery (DR) plan that incorporates the built-in cross-Region capabilities of AWS services. For AWS services that do not inherently provide cross-Region functionality, the DR plan must also provide a solution to handle their failover across AWS Regions.

This pattern guides you through a disaster recovery setup involving two Amazon Aurora MySQL-Compatible Edition database clusters in a single Region. To meet DR requirements, the database clusters are configured to use the Amazon Aurora global database feature, with a single database spanning multiple AWS Regions. An AWS Database Migration Service (AWS DMS) task replicates data between the clusters in the local Region. AWS DMS, however, currently doesn't support task failover between Regions. This pattern includes the steps required to work around that limitation and independently configure AWS DMS in both Regions.

Prerequisites and limitations

Prerequisites

- Selected primary and secondary AWS Regions that support Amazon Aurora global databases.
- Two independent Amazon Aurora MySQL-Compatible Edition database clusters in a single account in the primary Region.
- Database instance class db.r5 or higher (recommended).
- An AWS DMS task in the primary Region performing ongoing replication between the existing database clusters.
- DR Region resources in place to meet requirements for creating database instances. For more information, see Working with a DB instance in a VPC.

Limitations

- For the full list of Amazon Aurora global database limitations, see Limitations of Amazon Aurora global databases.

Product versions

- Amazon Aurora MySQL-Compatible Edition 5.7 or 8.0. For more information, see Amazon Aurora versions.

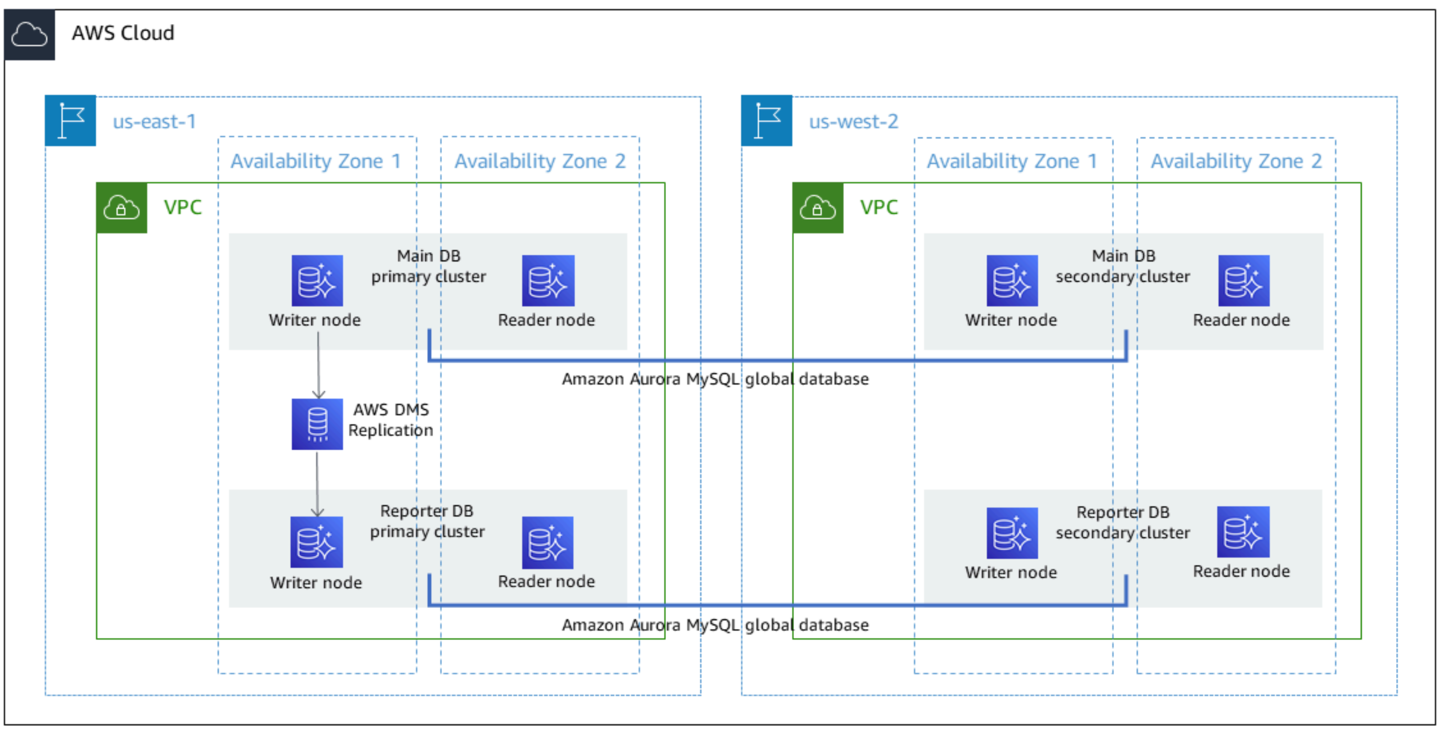
Architecture

Target technology stack

- Amazon Aurora MySQL-Compatible Edition global database cluster
- AWS DMS

Target architecture

The following diagram shows a global database for two AWS Regions, one with the primary main and reporter databases and AWS DMS replication, and one with the secondary main and reporter databases.



Automation and scale

You can use AWS CloudFormation to create the prerequisite infrastructure in the secondary Region, such as the virtual private cloud (VPC), subnets, and parameter groups. You can also use

AWS CloudFormation to create the secondary clusters in the DR Region and add them to the global database. If you used CloudFormation templates to create the database clusters in the primary Region, you can update or augment them with an additional template to create the global database resource. For more information, see Creating an Amazon Aurora DB cluster with two DB instances and Creating a global database cluster for Aurora MySQL.

Finally, you can create the AWS DMS tasks in the primary and secondary Regions using CloudFormation after failover and failback events occur. For more information, see AWS::DMS::ReplicationTask.

Tools

- Amazon Aurora is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. This pattern uses Amazon Aurora MySQL-Compatible Edition.
- Amazon Aurora global databases are designed for globally distributed applications. A single Amazon Aurora global database can span multiple AWS Regions. It replicates your data with no impact on database performance. It also enables fast local reads with low latency in each Region, and it provides disaster recovery from Region-wide outages.
- AWS DMS provides one-time migration or on-going replication. An on-going replication task keeps your source and target databases in sync. After it is set up, the on-going replication task continuously applies source changes to the target with minimal latency. All AWS DMS features, such as data validation and transformations, are available for any replication task.

Epics

Prepare the existing database clusters in the primary Region

Task	Description	Skills required
Modify the database cluster parameter group.	In the existing database cluster parameter group, activate row-level binary logging by setting the `binlog_format` parameter to a value of **row**. AWS DMS requires row-level binary logging for MYSQL-	AWS administrator

Task	Description	Skills required
	compatible databases when performing ongoing replication or change data capture (CDC). For more information, see Using an AWS managed MySQL-compatible database as a source for AWS DMS.	

Task	Description	Skills required

Update the database binary log retention period.	Using a MySQL client installed on your end-user device or an Amazon Elastic Compute Cloud (Amazon EC2) instance, run the following stored procedure provided by Amazon Relational Database Service (Amazon RDS) on the main database cluster's writer node, where XX is the number of hours to retain the logs.
	
	`call mysql.rds_set_configuration('binlog retention hours', XX)`
	
	Confirm the setting by running the following command.
	
	`call mysql.rds_show_configuration;`
	
	MySQL-compatible databases managed by AWS purge the binary logs as soon as possible. Therefore, the retention period must be long enough to ensure that the logs are not purged before the AWS DMS task runs. A value of 24 hours is usually sufficient, but the value should be based on the time
		DBA

Task	Description	Skills required
	required to set up the AWS DMS task in the DR Region.	

Update the existing AWS DMS task in the primary Region

Task	Description	Skills required

Record the AWS DMS task ARN.	Use the Amazon Resource Name (ARN) to obtain the AWS DMS task name for later use. To retrieve the AWS DMS task ARN, view the task in the console or run the following command.
	`aws dms describe-replication-tasks`
	An ARN looks like the following.
	`arn:aws:dms:us-east-1:<accountid>:task:AN6HFFMPM246XOZVEUHCNSOVF7MQCLTOZUIRAMY`
	The characters after the last colon correspond to the task name used in a later step.
		AWS administrator

Modify the existing AWS DMS task to record the checkpoint.	AWS DMS creates checkpoints that contain information so that the replication engine knows the recovery point	AWS administrator

Task	Description	Skills required
	for the change stream. To record checkpoint informati on, perform the following steps in the console: 1. Stop the AWS DMS task. 2. Use the JSON editor in the task to set the TaskRecoveryTableEnabled parameter to **true**. 3. Start the AWS DMS task.	
Validate checkpoint informati on.	Using a MySQL client connected to the writer endpoint for the cluster, query the new metadata table in the reporter database cluster to verify that it exists and contains the replication state information. Run the following command. `select * from awsdms_control.awsdms_txn_state;` The task name from the ARN should be found in this table in the `Task_Name` column.	DBA

Expand both Amazon Aurora clusters to a DR Region

Task	Description	Skills required
Create base infrastructure in the DR Region.	Create the base component s required for the creation of and access to the Amazon Aurora clusters: • Virtual private cloud (VPC) • Subnets • Security group • Network access control lists • Subnet group • DB parameter group • DB cluster parameter group Ensure that the configuration of both parameter groups matches the configuration in the primary Region.	AWS administrator
Add the DR Region to both Amazon Aurora clusters.	Add a secondary Region (the DR Region) to the main and reporter Amazon Aurora clusters. For more informati on, see Adding an AWS Region to an Amazon Aurora global database.	AWS administrator

Perform failover

Task	Description	Skills required
Stop the AWS DMS task.	The AWS DMS task in the primary Region will not function properly after failover occurs and should be stopped to avoid errors.	AWS administrator
Perform a managed failover.	Perform a managed failover of the main database cluster to the DR Region. For instructi ons, see Performing managed planned failovers for Amazon Aurora global databases. After failover on the main database cluster is complete, perform the same activity on the reporter database cluster.	AWS administrator, DBA
Load data into the main database.	Insert test data into writer node of the main database in the DR database cluster. This data will be used to validate that replication is functioning properly.	DBA
Create the AWS DMS replicati on instance.	To create the AWS DMS replication instance in the DR Region, see Creating a replication instance.	AWS administrator, DBA
Create the AWS DMS source and target endpoints.	To create the AWS DMS source and target endpoints in the DR Region, see Creating source and target endpoints . The source should point to	AWS administrator, DBA

Task	Description	Skills required
	the writer instance of the main database cluster. The target should point to the writer instance of the reporter database cluster.	
Obtain the replication checkpoint.	To obtain the replication checkpoint, use a MySQL client to query the metadata table by running the following against the writer node in the reporter database cluster in the DR Region. `select * from awsdms_control.awsdms_txn_state;` In the table, find the task_name value that corresponds to the AWS DMS task's ARN that exists in the primary Region that you obtained in the second epic.	DBA

Task	Description	Skills required
Create an AWS DMS task.	Using the console, create an AWS DMS task in the DR Region. In the task, specify a migration method of **Replicate data changes only**. For more information, see [Creating a task](). 1. In the task settings, use the wizard to specify the following: • **CDC start mode for source transactions** – Enable custom CDC start mode • **Custom CDC start point for source transactions** – Specify a recovery checkpoint 2. In the **Recovery checkpoint** box, enter the replication checkpoint value previously obtained through the database query on the `awsdms_txn_state` table. 3. In the task settings section, select the JSON editor, and set the **TaskRecoveryTableEnabled** parameter to **true**.	AWS administrator, DBA

Task	Description	Skills required
	Set the AWS DMS task **Start migration task** setting to **Automatically on create**.	
Record the AWS DMS task ARN.	Use the ARN to obtain the AWS DMS task name for later use. To retrieve the AWS DMS task ARN, run the following command. `aws dms describe-replication-tasks`	AWS administrator, DBA
Validate the replicated data.	Query the reporter database cluster in the DR Region to confirm that the test data that you loaded into the main database cluster has been replicated.	DBA

Perform failback

Task	Description	Skills required
Stop the AWS DMS task.	The AWS DMS task in the DR Region will not function properly after failback occurs and should be stopped to avoid errors.	AWS administrator
Perform a managed failback.	Fail back the main database cluster to the primary Region. For instructions, see Performing managed planned failovers for Amazon	AWS administrator, DBA

Task	Description	Skills required
	Aurora global databases. After the failback on the main database cluster is complete, perform the same activity on the reporter database cluster.	
Obtain the replication checkpoint.	To obtain the replication checkpoint, use a MySQL client to query the metadata table by running the following against the writer node in the reporter database cluster in the DR Region. `select * from awsdms_control.awsdms_txn_state;` In the table, find the `task_name` value that corresponds to the AWS DMS task's ARN that exists in the DR Region that you obtained in the fourth epic.	DBA

Task	Description	Skills required
Update the AWS DMS source and target endpoints.	After the database clusters have failed back, check the clusters in the primary Region to determine which nodes are the writer instances . Then verify the existing AWS DMS source and target endpoints in the primary Region are pointing to the writer instances. If not, update the endpoints with the writer instance Domain Name System (DNS) names.	AWS administrator

Task	Description	Skills required
Create an AWS DMS task.	Using the console, create an AWS DMS task in the primary Region. In the task, specify a migration method of **Replicate data changes only**. For more information, see [Creating a task](#). 1. In the task settings, use the wizard and specify the following: • **CDC start mode for source transactions** – Enable custom CDC start mode • **Custom CDC start point for source transacti ons** – Specify a recovery checkpoint 2. In the **Recovery checkpoin t** box, enter the replication checkpoint value previousl y obtained through the database query on the `awsdms_txn_state` table. 3. Also within the task settings section, select the JSON editor and set the **TaskRecoveryTableE nabled** parameter to **true**. 4. Finally, set the AWS DMS task **Start migration task**	AWS administrator, DBA

Task	Description	Skills required
	setting to **Automatically on create**.	
Record the AWS DMS task Amazon Resource Name (ARN).	Use the ARN to obtain the AWS DMS task name for later use. To retrieve the AWS DMS task ARN, run the following command: `aws dms describe-replication-tasks` The task name will be needed when performing another managed failover or during a DR scenario.	AWS administrator, DBA
Delete AWS DMS tasks.	Delete the original (currentl y stopped) AWS DMS task in the primary Region and the existing AWS DMS task (currently stopped) in the secondary Region.	AWS administrator

Related resources

- Configuring your Amazon Aurora DB cluster
- Using Amazon Aurora global databases
- Working with Amazon Aurora MySQL
- Working with an AWS DMS replication instance
- Working with AWS DMS endpoints
- Working with AWS DMS tasks
- What is AWS CloudFormation?

Additional information

Amazon Aurora global databases are used in this example for DR because they provide an effective recovery time objective (RTO) of 1 second and a recovery point objective (RPO) of less than 1 minute, both lower than traditional replicated solutions and ideal for DR scenarios.

Amazon Aurora global databases offer many other advantages, including the following:

- Global reads with local latency – Global consumers can access information in a local Region, with local latency.
- Scalable secondary Amazon Aurora DB clusters – Secondary clusters can be scaled independently, adding up to 16 read-only replicas.
- Fast replication from primary to secondary Amazon Aurora DB clusters – Replication has little performance impact on the primary cluster. It occurs at the storage layer, with typical cross-Region replication latencies of less than 1 second.

This pattern also uses AWS DMS for replication. Amazon Aurora databases provide the ability to create read replicas, which can simplify the replication process and the DR setup. However, AWS DMS is often used to replicate when data transformations are required or when the target database requires additional indexes that the source database does not have.

Migrate Oracle functions and procedures that have more than 100 arguments to PostgreSQL

Srinivas Potlachervoo, Amazon Web Services

Summary

This pattern shows how to migrate Oracle Database functions and procedures that have more than 100 arguments to PostgreSQL. For example, you can use this pattern to migrate Oracle functions and procedures to one of the following PostgreSQL-compatible AWS database services:

- Amazon Relational Database Service (Amazon RDS) for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

PostgreSQL doesn’t support functions or procedures that have more than 100 arguments. As a workaround, you can define a new data type that has type fields that match the source function’s arguments. Then, you can create and run a PL/pgSQL function that uses the custom data type as an argument.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS Oracle database (DB) instance
- An Amazon RDS for PostgreSQL DB instance or an Aurora PostgreSQL-Compatible DB instance

Product versions

- Amazon RDS Oracle DB instance versions 10.2 and later
- Amazon RDS PostgreSQL DB instance versions 9.4 and later, or Aurora PostgreSQL-Compatible DB instance versions 9.4 and later
- Oracle SQL Developer version 18 and later
- pgAdmin version 4 and later

Architecture

Source technology stack

- Amazon RDS Oracle DB instance versions 10.2 and later

Target technology stack

- Amazon RDS PostgreSQL DB instance versions 9.4 and later, or Aurora PostgreSQL-Compatible DB instance versions 9.4 and later

Tools

AWS services

- Amazon Relational Database Service (Amazon RDS) for PostgreSQL helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.
- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.

Other services

- Oracle SQL Developer is an integrated development environment that simplifies the development and management of Oracle databases in both traditional and cloud-based deployments.
- pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.

Best practices

Make sure that the data type that you create matches the type fields that are included in the source Oracle function or procedure.

Epics

Run an Oracle function or procedure that has more than 100 arguments

Task	Description	Skills required
Create or identify an existing Oracle/PLSQL function or procedure that has more than 100 arguments.	Create an Oracle/PLSQL function or procedure that has more than 100 arguments . -or- Identify an existing Oracle/ PLSQL function or procedure that has more than 100 arguments. For more information, see sections 14.7 CREATE FUNCTION Statement and 14.11 CREATE PROCEDURE Statement in the Oracle Database documentation.	Oracle/PLSQL knowledge
Compile the Oracle/PLSQL function or procedure.	Compile the Oracle/PLSQL function or procedure. For more information, see Compiling a function in the Oracle Database documenta tion.	Oracle/PLSQL knowledge
Run the Oracle/PLSQL function.	Run the Oracle/PLSQL function or procedure. Then, save the output.	Oracle/PLSQL knowledge

Define a new data type that matches the source function's or procedure's arguments

Task	Description	Skills required
Define a new data type in PostgreSQL.	Define a new data type in PostgreSQL that includes all of the same fields that appear in the source Oracle function's or procedure's arguments. For more information, see [CREATE TYPE] in the PostgreSQL documentation.	PostgreSQL PL/pgSQL knowledge

Create a PostgreSQL function that includes the new TYPE argument

Task	Description	Skills required
Create a PostgreSQL function that includes the new data type.	Create a PostgreSQL function that includes the new TYPE argument. To review an example function, see the **Additional information** section of this pattern.	PostgreSQL PL/pgSQL knowledge
Compile the PostgreSQL function.	Compile the function in PostgreSQL. If the new data type fields match the source function's or procedure's arguments, then the function successfully compiles.	PostgreSQL PL/pgSQL knowledge
Run the PostgreSQL function.	Run the PostgreSQL function.	PostgreSQL PL/pgSQL knowledge

Troubleshooting

Issue	Solution
The function returns the following error: **ERROR: syntax error near "<statement>"**	Make sure that all of the function's statements end with a semicolon (`;`).
The function returns the following error: **ERROR: "<variable>" is not a known variable**	Make sure that the variable that's used in the function body is listed within the function's `DECLARE` section.

Related resources

- [Working with Amazon Aurora PostgreSQL](#) (*Amazon Aurora User Guide for Aurora*)
- [CREATE TYPE](#) (PostgreSQL documentation)

Additional information

Example PostgreSQL function that includes a TYPE argument

```
CREATE OR REPLACE FUNCTION test_proc_new
(
    IN p_rec type_test_proc_args
)
RETURNS void
AS
$BODY$
BEGIN

    /*
    **************
    The body would contain code to process the input values.
    For our testing, we will display couple of values.
    ***************
    */
    RAISE NOTICE USING MESSAGE = CONCAT_WS('', 'p_acct_id: ', p_rec.p_acct_id);
    RAISE NOTICE USING MESSAGE = CONCAT_WS('', 'p_ord_id: ', p_rec.p_ord_id);
    RAISE NOTICE USING MESSAGE = CONCAT_WS('', 'p_ord_date: ', p_rec.p_ord_date);
```

```
END;
$BODY$
LANGUAGE plpgsql
COST 100;
```

Migrate SAP HANA to AWS using SAP HSR with the same hostname

Pradeep Puliyampatta, Amazon Web Services

Summary

SAP HANA migrations to Amazon Web Services (AWS) can be performed using multiple options, including backup and restore, export and import, and SAP HANA System Replication (HSR). The selection of a particular option depends on the network connectivity between source and target SAP HANA databases, the size of the source database, downtime considerations, and other factors.

The SAP HSR option for migrating SAP HANA workloads to AWS works well when there is a stable network between the source and target systems and the entire database (SAP HANA DB replication snapshot) can be completely replicated within 1 day, as stipulated by SAP for network throughput requirements for SAP HSR. The downtime requirements with this approach are limited to performing the takeover on the target AWS environment, SAP HANA DB backup, and post-migration tasks.

SAP HSR supports the use of different hostnames (hostnames mapped to different IP addresses) for replication traffic between the primary, or source, and secondary, or target, systems. You can do this by defining those specific sets of hostnames under the `[system_replication_hostname_resolution]` section in `global.ini`. In this section, all hosts of the primary and the secondary sites must be defined on each host. For detailed configuration steps, see the [SAP documentation](#).

One key takeaway from this setup is that the hostnames in the primary system must be different from the hostnames in the secondary system. Otherwise, the following errors can be observed.

- `"each site must have a unique set of logical hostnames"`
- `"remoteHost does not match with any host of the source site. All hosts of source and target site must be able to resolve all hostnames of both sites correctly"`

However, the number of post-migration steps can be reduced by using the same SAP HANA DB hostname on the target AWS environment.

This pattern provides a workaround for using the same hostname on source and target environments when using the SAP HSR option. With this pattern, you can use the SAP HANA Hostname Rename option. You assign a temporary hostname to the target SAP HANA DB to

facilitate hostname uniqueness for SAP HSR. After the migration completes the takeover milestone on the target SAP HANA environment, you can revert the target system hostname back to the hostname of the source system.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A virtual private cloud (VPC) with a virtual private network (VPN) endpoint or a router.
- AWS Client VPN or AWS Direct Connect configured to transfer files from the source to the target.
- SAP HANA databases in both the source and the target environment. The target SAP HANA DB patch level should be equal to or higher than the source SAP HANA DB patch level, within the same SAP HANA Platform edition. For example, replication cannot be set up between HANA 1.0 and HANA 2.0 systems. For more information, see question 15 in SAP Note: 1999880 – FAQ: SAP HANA System Replication.
- SAP application servers in the target environment.
- Amazon Elastic Block Store (Amazon EBS) volumes in the target environment.

Limitations

The following list of SAP documents covers known issues that are related to this workaround, including constraints regarding SAP HANA dynamic tiering and scale-out migrations:

- 2956397 – Renaming of SAP HANA Database System failed
- 2222694 – When trying to rename the HANA system, the following error appears "Source files are not owned by the original sidadm user (uid = xxxx)"
- 2607227 – hdblcm: register_rename_system: Renaming SAP HANA instance failed
- 2630562 – HANA Hostname Rename failed and HANA does not start up
- 2935639 – sr_register is not using the hostname that is specified under system_replication_hostname_resolution in the global.ini section
- 2710211 – Error: source system and target system have overlapping logical hostnames
- 2693441 – Failed to rename an SAP HANA System due to error
- 2519672 – HANA Primary and Secondary has different system PKI SSFS data and key or unable to check

- 2457129 – SAP HANA System Host Rename is not Permitted when Dynamic Tiering is Part of Landscape
- 2473002 – Using HANA System Replication to migrate scale out system (There are no restrictions provided by SAP in using this hostname rename approach for scale-out SAP HANA systems. However, the procedure must be repeated on each individual host. Other scale-out migration limitations also apply to this approach.)

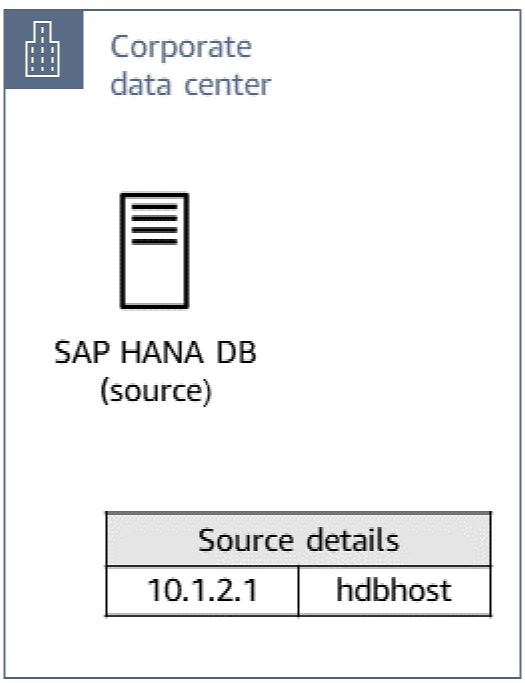
Product versions

- This solution applies to SAP HANA DB platform edition 1.0 and 2.0.

Architecture

Source setup

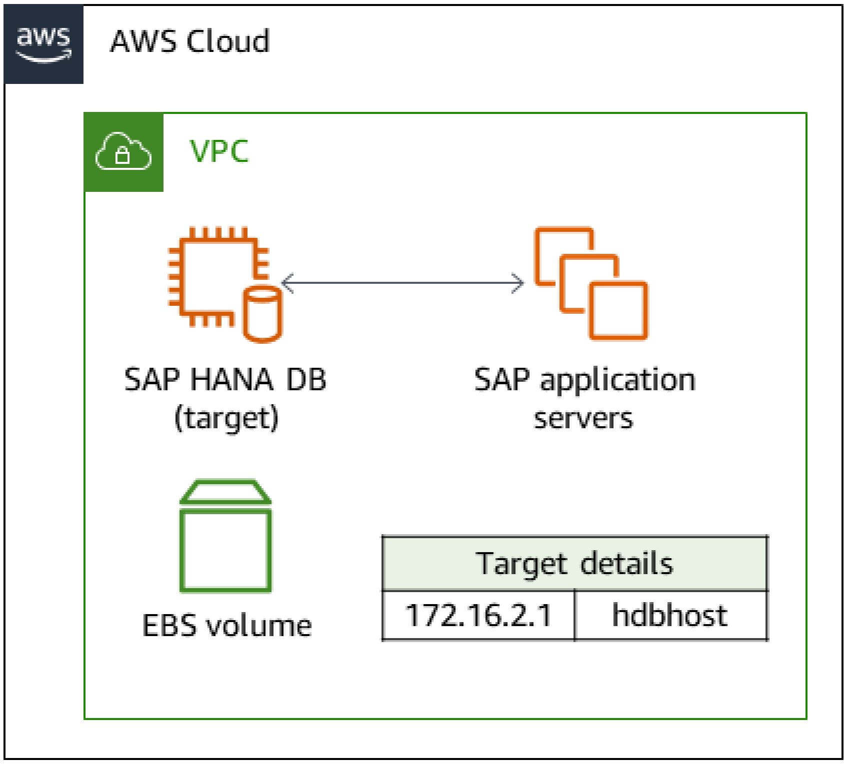
An SAP HANA database is installed on the source environment. All the SAP application server connections and DB interfaces use the same hostname for client connections. The following diagram shows the example source hostname `hdbhost` and its corresponding IP address.



Target setup

The AWS Cloud target environment uses the same hostname to run an SAP HANA database. The target environment on AWS includes the following:

- SAP HANA database
- SAP application servers
- EBS volumes

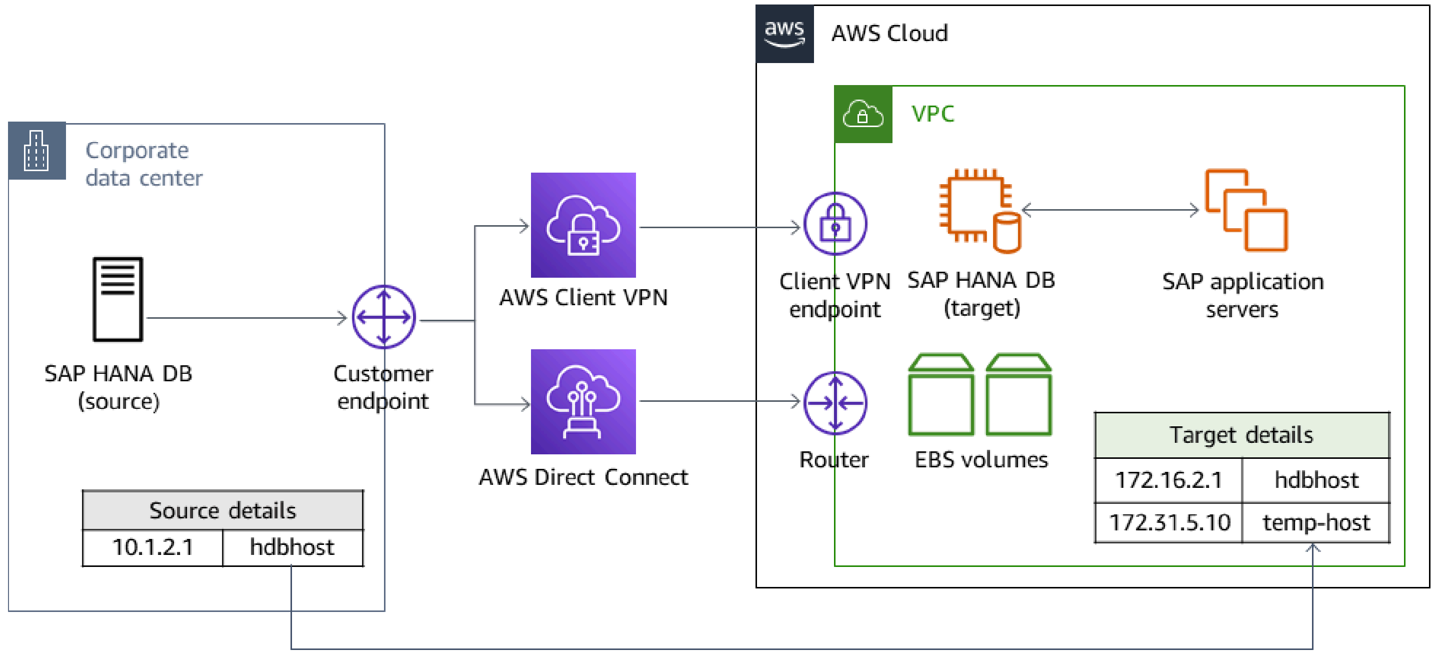


Intermediate configuration

In the following diagram, the hostname on the AWS target environment is temporarily renamed as `temp-host` so that the hostnames on the source and target are unique. After the migration completes the takeover milestone on the target environment, the target system virtual hostname is renamed using the original name, `hdbhost`.

The intermediate configuration includes one of the following options:

- AWS Client VPN with a Client VPN endpoint
- AWS Direct Connect connecting to a router



SAP application servers on the AWS target environment can be installed either before replication setup or after the takeover. However, installing the application servers before replication setup can help with reduction of downtime during installation, configuration of high availability, and backups.

Tools

AWS services

- AWS Client VPN is a managed client-based VPN service that enables you to securely access AWS resources and resources in your on-premises network.
- AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services, bypassing internet service providers in your network path.
- Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances.

Other tools

- SAP application servers – SAP application servers provide programmers with a way to express business logic. The SAP application server performs the data processing based on the business logic. The actual data is stored in a database, which is a separate component.
- SAP HANA cockpit and SAP HANA Studio – Both SAP HANA cockpit and SAP HANA Studio provide an administrative interface to the SAP HANA database. In SAP HANA Studio, the SAP HANA Administration console is the system view that provides relevant content for SAP HANA database administration.
- SAP HANA System Replication – SAP HANA System Replication (SAP HSR) is the standard procedure provided by SAP for replicating SAP HANA databases. The required executables for SAP HSR are part of the SAP HANA server kernel itself.

Epics

Prepare the source and target environments

Task	Description	Skills required
Install and configure the SAP HANA databases.	In the source and target environments, ensure that the SAP HANA DB is installed and configured according to SAP HANA on best practices. For more information, see SAP HANA on AWS.	SAP Basis administration
Map the IP address.	In the target environment, ensure that the temporary hostname is assigned to an internal IP address. 1. Assign a secondary IPv4 address to the EC2 instance on the AWS Managemen t Console by navigating to **EC2**, **Instance**, **Actions**, **Networking**, **Manage IP**	AWS administration

Task	Description	Skills required
	address, **Assign new IP address**. 2. To assign the same address to the EC2 network adaptor (NIC), from the operating system, as root user, run the command `ip addr add <IP>/32 dev eth0`, replacing `<IP>` with the IP address from step 1.	
Resolve target hostnames.	On the secondary SAP HANA DB, confirm that both hostnames (`hdbhost` and `temp-host`) are resolved for the SAP HANA replicati on networks by updating the relevant hostnames in the `/etc/hosts` file.	Linux administration
Back up the source and target SAP HANA databases.	Use SAP HANA Studio or the SAP HANA cockpit to perform backups on the SAP HANA databases.	SAP Basis administration

Task	Description	Skills required
Exchange system PKI certifica tes.	(Applies only to SAP HANA 2.0 and later) Exchange certificates in the system public key infrastructure (PKI) secure store in the file system (SSFS) store between the primary and secondary databases. For more informati on, see SAP Note 2369981 – Required configuration steps for authentication with SAP HANA System Replication.	SAP Basis administration

Rename the target SAP HANA DB

Task	Description	Skills required
Stop target client connectio ns.	In the target environment, shut down the SAP applicati on servers and other client connections.	SAP Basis administration

Rename the target SAP HANA DB to the temporary hostname.	1. As root user, rename the target SAP HANA DB hostname to the temporary hostname by using resident `hdblcm`.
	`root $> cd /hana/shared/<SID/hdblcm`
	`root $> ./hdblcm`
	2. Choose option 9 | `rename_system` |
		SAP Basis administration

Task	Description	Skills required
	`Rename the SAP HANA Database System`. 3. Provide the new name: `temp-host` . 4. You can validate other options as needed. However, be sure that you don't mix up the host rename with a SID change (SAP Note 2598814 – hdblcm: SID rename fails). The SAP HANA DB stop and start will be controlled by `hdblcm`.	
Assign replication networks.	In the `global.ini` file of the source system, under the `[system_replication_hostname_resolution]` header, provide the source and target replication network details. Then copy the entries to the `global.ini` file on the target system.	SAP Basis administration
Enable replication on primary.	To enable replication on the source SAP HANA DB, run the following command. `hdbnsutil -sr_enable --name=siteA`	SAP Basis administration

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Register the target SAP HANA DB as a secondary system.</td><td>To register the target SAP HANA DB as a secondary system to source for SAP HSR, choose async replication.
<pre><code>(sid)adm $> HDB stop
(sid)adm $> hdbnsutil -
sr_register -name=sit
eB -remotehost=hdbhos
t /
--remoteInstance=00 -
replicationMode=async
 -operationMode=log
replay
(sid)adm $> HDB start</code></pre>Alternatively, you can choose the <code>–online</code> option to register. In that case, you don't need to stop and start the SAP HANA DB.</td><td>SAP Basis administration</td></tr>
<tr><td>Validate synchronization.</td><td>On the source SAP HANA DB, verify that all the logs are applied on the target system (because it is async replicati on).

To verify the replication, on the source, run the following commands.
<pre><code>(sid)adm $> cdpy
(sidadm $> python
 systemReplicationS
tatus.py</code></pre></td><td>SAP Basis administration</td></tr>
</table>

Task	Description	Skills required
Shut down the source SAP application and SAP HANA DB.	During the migration cutover, perform a shutdown of the source system (the SAP application and SAP HANA database.	SAP Basis administration
Perform a takeover at the target.	To perform a takeover at the target on AWS, run the command `hdbnsutil -sr_takeover` .	SAP Basis administration
On the target SAP HANA DB, turn off replication.	To clear the replication metadata, stop replication on the target system by running the command `hdbnsutil -sr_disable` . **Note** This is in accordanc e with SAP Note 2693441 – Failed to rename an SAP HANA System due to error.	SAP Basis administration
Back up the target SAP HANA DB.	After the takeover is successfu l, we recommend performing a full SAP HANA DB backup.	SAP Basis administration

Revert to the original hostname in the target system

Task	Description	Skills required

Revert the target SAP HANA DB hostname to the original.	1. To revert the target SAP HANA DB hostname to the original virtual hostname, use resident `hdblcm`.
	`root $> cd /hana/shared/<SID>/hdblcm`
	`root $> ./hdblcm`
	2. Choose option `9 | rename_system | Rename the SAP HANA Database System`.
	3. Provide the new name: `hdbhost`.
	
	You can validate other options as needed. However, be sure that you don't mix up the host rename with a SID change (SAP Note 2598814 – hdblcm: SID rename fails).
		SAP Basis administration

Adjust hdbuserstore.	Adapt the `hdbuserstore` details pointing to the source `schema/user` details. For detailed steps, see the SAP documentation. To validate this step, run the command `R3trans -d`. The result should reflect a	SAP Basis administration

Task	Description	Skills required
	successful connection to the SAP HANA database.	
Start up client connections.	In the target environment, start up the SAP applicati on servers and other client connections.	SAP Basis administration

Related resources

SAP references

SAP documentation references are frequently updated by SAP. To stay up to date, see SAP Note 2407186 – How-To Guides & Whitepapers For SAP HANA High Availability.

Additional SAP notes

- 2550327 – How-To Rename an SAP HANA System
- 1999880 – FAQ: SAP HANA System Replication
- 2078425 – Troubleshooting note for SAP HANA platform lifecycle management tool hdblcm
- 2592227 – FQDN suffix change in HANA systems
- 2048681 – Performing SAP HANA platform lifecycle management administration tasks on multiple-host systems without SSH or root credentials

SAP documents

- System Replication Network Connection
- Host Name Resolution for System Replication

AWS references

- Migrating SAP HANA from Other Platforms to AWS

Additional information

The changes performed by `hdblcm` as part of the hostname rename activity are consolidated in the following verbose log.

```
All server processes stopped on host 'temp-host' (worker).
  Stopping sapstartsrv service...
  Removing sapservices entry...
  Updating system configuration files...
  Renaming instance...
  Configuring environment scripts...
  Creating sapservices entry...
  Starting service (sapstartsrv)...
  Starting instance <SID> (HDB00) on host 'hdbhost'...
    Starting 7 processes on host 'hdbhost' (worker):
......
All server processes started on host 'hdbhost' (worker).
  Registering Instance Service...
  Updating Component List...
  Updating SAP HANA Database Instance Integration on Local Host...
  Regenerating SSL certificates...
  Deploying SAP Host Agent configurations...
SAP HANA Database System renamed
```

Migrate Microsoft SQL Server Always On availability group using AWS Application Migration Service

Sreenivas Nettem, Bharath Kumar Pammi Ramesh, Anantharaman Seshadri, and Gireesh Sreekantan, Amazon Web Services

Summary

AWS Application Migration Service (AWS MGN) is a preferred tool for rehosting existing environments in the AWS Cloud, which allows customers to move away from on-premises data centers. This pattern outlines the process of using AWS MGN to migrate Windows clusters with Microsoft SQL Server Always On availability groups.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An AWS Identity and Access Management (IAM) role for AWS MGN orchestration.
- Access to source database servers (SQL Server Always On availability group).
- Active Directory in an AWS landing zone to retain DNS names.
- A staging subnet with closed network communication to Active Directory.
- Target subnets that can communicate with Active Directory.
- Two reserved IP addresses for the Windows cluster in a target subnet (one in each Availability Zone).
- Two reserved IP addresses for the SQL Always On listener in a target subnet (one in each Availability Zone).

Product versions

- Windows Server 2012 or later
- SQL Server 2012 or later

Architecture

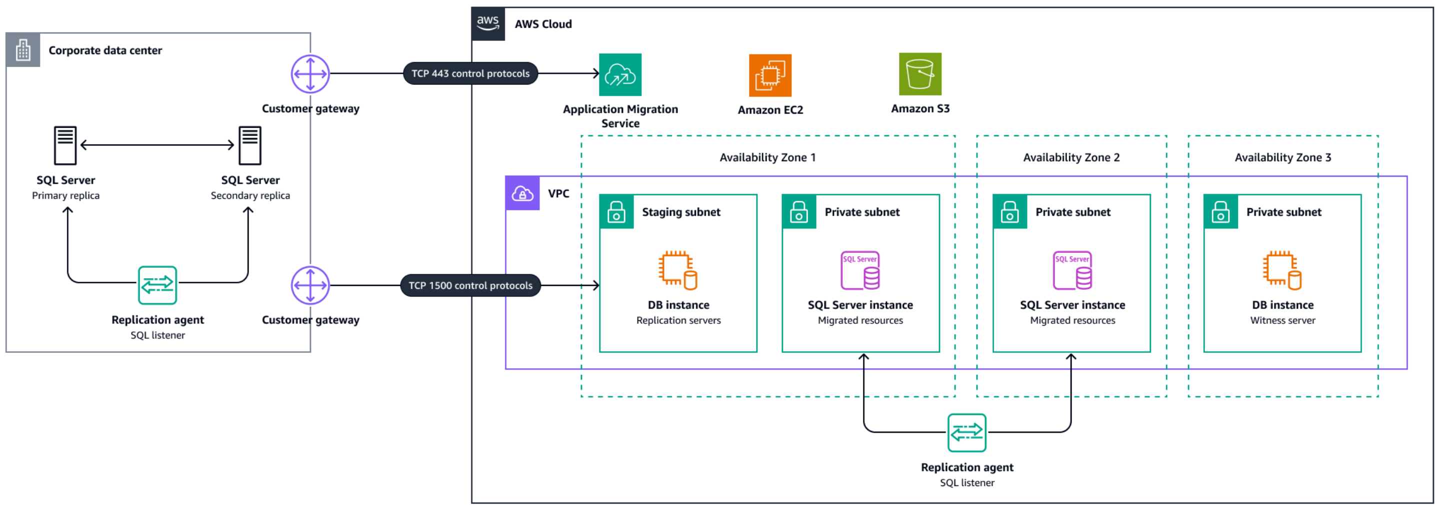
Source technology stack

Microsoft Windows cluster (on-premises physical or virtual machine) Microsoft SQL Server Always On availability group

Target technology stack

Amazon EC2 Windows instance

Target architecture



Tools

AWS services

- [Amazon Elastic Compute Cloud (Amazon EC2)] provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- [AWS Application Migration Service] helps you rehost (lift and shift) applications to the AWS Cloud without change and with minimal downtime.
- [AWS Identity and Access Management (IAM)] helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Other tools

- [Microsoft SQL Server Management Studio (SSMS)] is a tool for managing SQL Server, including accessing, configuring, and administering SQL Server components.

Best practices

For AWS MGN, see Best practices for AWS Application Migration Service.

Epics

Prepare the target account

Task	Description	Skills required
Initialize AWS MGN.	Initialize AWS MGN in the target AWS Region. This creates the required IAM roles and policies. For more information, see Initializing Application Migration Service with the console.	Cloud administrator
Create replication and launch templates.	Configure the replication and launch templates for use with AWS MGN. For more information, see Configuring the templates in the AWS documentation.	Cloud administrator
Allow communication ports.	To enable network communications for AWS MGN, allow traffic over TCP ports 443 and 1500. For more information, see Network requirements for Application Migration Service in the AWS documentation.	Cloud administrator, Network administrator

Prepare the source server

Task	Description	Skills required
Verify AWS MGN prerequis ites.	Verify that the source servers meet the prerequisites for the AWS MGN agent installat ion. For more information, see [Installation requirements](#) in the AWS documentation.	Migration engineer
Install AWS MGN agent.	Install the AWS MGN agent on the source servers. During the installation, choose the AWS Region where the servers will be migrated. After installat ion, the agent communica tes with the service and starts replication. For more information, see [Installing the AWS Replication Agent on Windows servers](#).	Migration engineer
Check source server status.	In the AWS MGN console, check the status of source servers. The servers display **Ready for testing** when replication begins. If you encounter any errors, see [Troubleshooting communication errors](#) in the AWS MGN documentation.	Cloud administrator, Migration engineer
Optimize replication settings.	SQL Always On clusters use high I/O synchronous replicati on from the primary server to the secondary server. To	Cloud administrator, Migration engineer

Task	Description	Skills required
	optimize replication and avoid lag, use a dedicated replication server for each SQL Always On server. If the database is greater than 5 TB, consider choosing a larger replication server instance size such as **m5.large** instead of the default **t3.small**.	
Update launch template.	Update the launch settings and choose subnets for the SQL Always On servers. The SQL Always On cluster servers are spread across different AWS Availability Zones for high availability.	Migration engineer, Migration lead
Update launch settings.	Based on your size and performance requirements, update the instance type and input/output operations per second (IOPS) in the launch settings. (Optional) Choose an existing elastic network interface in the launch settings.	Migration engineer, Migration lead

Test cutover

Task	Description	Skills required
Verify source servers.	In the AWS MGN console, verify that the source server status is **Ready for testing**.	Cloud administrator, Migration engineer
Launch the test instances.	1. Launch the test instances and verify in the Amazon EC2 console that the automated checks pass. 2. Select and launch test instances for the witness server. 3. Verify that you can sign in to the servers through the AWS MGN console. 4. Select the SQL Always On cluster servers and launch the test instances together.	Cloud administrator, Migration engineer
Test for connectivity and database integrity.	Test the connectivity and database integrity of the test instances. Then mark the source servers as **Ready for cutover** in the AWS MGN console.	Cloud administrator, Migration engineer

Pre-migration tasks

Task	Description	Skills required
Test the database integrity.	This helps to ensure there are no database integrity issues at the source prior	Data engineer, DBA

Task	Description	Skills required
	to migration. Run `DBCC CHECKDB` and specify `WITH_PHYSICAL_ONLY` . Running this check without `WITH_PHYSICAL_ONLY` can cause performance issues at the source. To maintain database integrity, run a weekly full check of the database. These commands check the database's logical and physical integrity by detecting potential corruption issues. The check verifies the database's structure, such as pages, rows, indexes, and system tables.	
Test the connections to linked servers.	Test the connections among all existing servers, and document their status. This helps to ensure that linked servers work as intended after the migration.	Data engineer, DBA
Verify the backups.	Confirm the integrity of source backups.	Data engineer, DBA

AWS MGN cutover

Task	Description	Skills required
Stop the SQL Server and cluster services.	Stop the SQL Server and Microsoft cluster services on all SQL cluster nodes.	DBA, Migration engineer
Verify the servers.	In the AWS MGN console, verify that the status of the source servers is **Ready for cutover** and that the data replication status is **Healthy**.	Migration engineer
Launch the cutover.	1. Launch the AWS MGN cutover for the witness server. 2. Launch the AWS MGN cutover for the SQL Always On cluster instances. 3. Verify that the status changes to **Cutover in progress**. For more information, see [Launching a cutover instance](#) in the AWS MGN documenta tion.	Migration engineer
Test the launched servers.	Log in to the launched Amazon EC2 instances and validate the cluster’s health. Verify that the servers are in the correct subnets, the instance size and IOPS	DBA, Migration engineer

Task	Description	Skills required
	settings are correct, and the witness server is accessible.	

Database post-cutover tasks

Task	Description	Skills required
Update the cluster IP address.	Update the cluster IP address for the Windows cluster using the two reserved IP addresses in the target subnets. For more information, see [Change the IP Address of a Failover Cluster Instance]().	DBA, Migration engineer
Update the Always On availability group listener IPs.	1. Open the failover cluster manager. 2. Choose the Always On availability group role. 3. Expand the availability group listener name. 4. In the context (right-click) menu, choose **IP address properties**. 5. Update the IP address using the address reserved for the listener in the target subnet. 6. Connect to the SQL Server primary instance using SSMS, and verify that the Always On listener is using both subnet IPs.	DBA, Migration engineer

Task	Description	Skills required
Verify the connection.	Using SSMS, connect to the Always On availability group listener, and confirm that the connection succeeds.	DBA, Migration engineer
Check the health of Always On availability group.	1. Navigate to the **Availability groups** folder, open the context (right-click) menu, and choose **Show dashboard**. 2. For all replicas, verify that the synchronization state is **Synchronized**.	DBA, Migration engineer
Check the error log.	Open the error log, and verify any errors reported for the SQL Server instance. Ensure that the recovery is complete for all databases.	DBA, Migration engineer
Test the linked servers.	Test the connectivity of any linked servers. In case of any connectivity issues, ensure that the target server and port are accessible.	DBA, Migration engineer

Finalize the cutover

Task	Description	Skills required
Finalize the cutover.	After you validate the target SQL Always On clusters, finalize the cutover through the AWS MGN console. This stops data replication	Cloud administrator, Migration engineer

Task	Description	Skills required
	from the source servers and discards the data from the replication server. It also removes the replication server and its associated resources.	

Troubleshooting

Issue	Solution
AWS MGN troubleshooting	For common issues and resolutions, see the Troubleshooting and FAQ sections in the AWS MGN documentation.

Related resources

AWS resources

- Option-1 Rehost - AWS Application Migration Service (AWS)
- What Is AWS Application Migration Service?

SQL Server resources

- What is SQL Server Management Studio (SSMS)?

Additional information

For standard security requirements for migrating workloads to the AWS Cloud, see the Best Practices for Security, Identity, and Compliance on the AWS website.

Migrate SQL Server to AWS using distributed availability groups

Praveen Marthala, Amazon Web Services

Summary

Microsoft SQL Server Always On availability groups provide a high availability (HA) and disaster recovery (DR) solution for SQL Server. An availability group consists of a primary replica that accepts read/write traffic, and up to eight secondary replicas that accept read traffic. An availability group is configured on a Windows Server Failover Cluster (WSFC) with two or more nodes.

Microsoft SQL Server Always On distributed availability groups provide a solution to configure two separate availability groups between two independent WFSCs. The availability groups that are part of the distributed availability group don't have to be in the same data center. One availability group can be on premises, and the other availability group can be on the Amazon Web Services (AWS) Cloud on Amazon Elastic Compute Cloud (Amazon EC2) instances in a different domain.

This pattern outlines steps for using a distributed availability group to migrate on-premises SQL Server databases that are part of an existing availability group to SQL Server with availability groups set up on Amazon EC2. By following this pattern, you can migrate the databases to the AWS Cloud with minimal downtime during cutover. The databases are highly available on AWS immediately after the cutover. You can also use this pattern to change the underlying operating system from on-premises to AWS while keeping the same version of SQL Server.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Direct Connect or AWS Site-to-Site VPN
- The same version of SQL Server installed on-premises and on the two nodes on AWS

Product versions

- SQL Server version 2016 and later
- SQL Server Enterprise Edition

Architecture

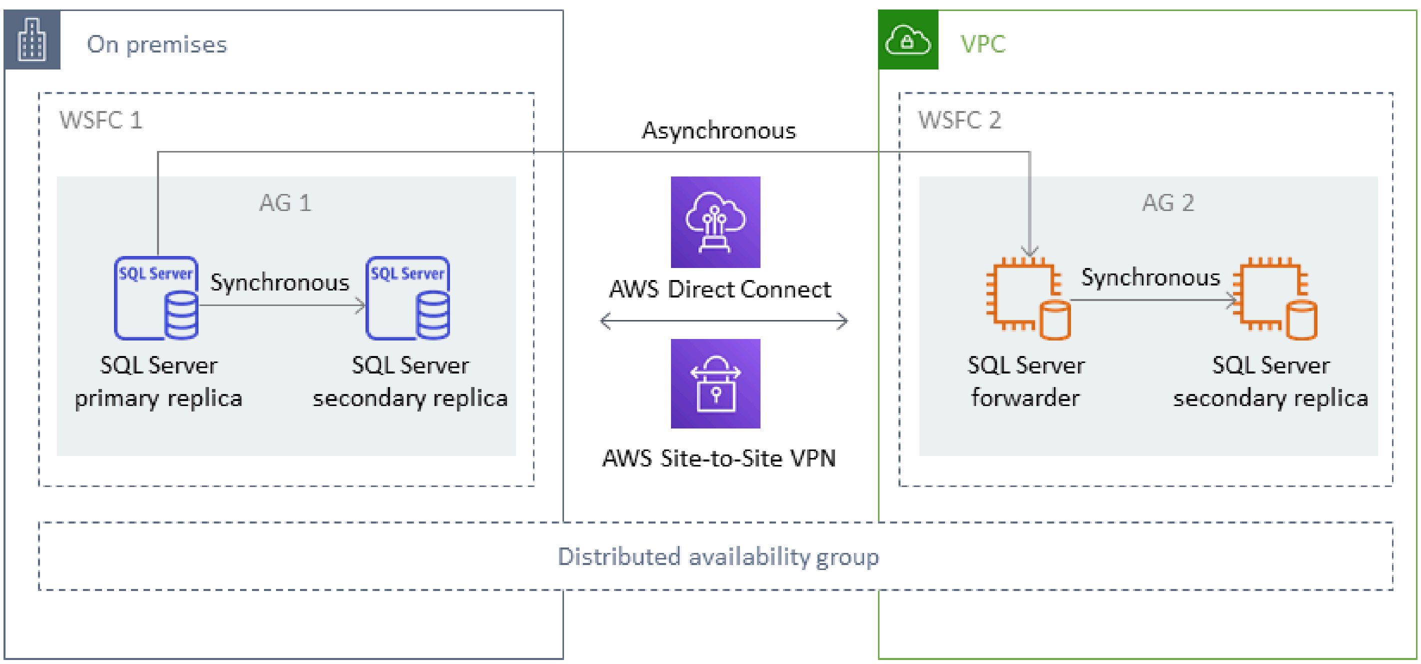
Source technology stack

- Microsoft SQL Server database with Always On availability groups on premises

Target technology stack

- Microsoft SQL Server database with Always On availability groups on Amazon EC2 on the AWS Cloud

Migration architecture



Terminology

- WSFC 1 – WSFC on premises
- WSFC 2 – WSFC on the AWS Cloud
- AG 1 – First availability group, which is in WSFC 1
- AG 2 – Second availability group, which is in WSFC 2
- SQL Server primary replica – Node in AG 1 that is considered the global primary for all writes
- SQL Server forwarder – Node in AG 2 that receives data asynchronously from the SQL Server primary replica

- SQL Server secondary replica – Nodes in AG 1 or AG 2 that receive data synchronously from the primary replica or the forwarder

Tools

- [AWS Direct Connect](#) – AWS Direct Connect links your internal network to an AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create *virtual interfaces* directly to public AWS services, bypassing internet service providers in your network path.
- [Amazon EC2](#) – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- [AWS Site-to-Site VPN](#) – AWS Site-to-Site VPN supports creating a site-to-site virtual private network (VPN). You can configure the VPN to pass traffic between instances that you launch on AWS and your own remote network.
- [Microsoft SQL Server Management Studio](#) – Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

Epics

Set up a second availability group on AWS

Task	Description	Skills required
Create a WSFC on AWS.	Create WSFC 2 on Amazon EC2 instances with two nodes for HA. You will use this failover cluster to create the second availability group (AG 2) on AWS.	Systems administrator, SysOps administrator
Create the second availability group on WSFC 2.	Using SSMS, create AG 2 on two nodes in WSFC 2. The first node in WSFC 2 will act as the forwarder. The second	DBA, Developer

Task	Description	Skills required
	node in WSFC 2 will act as the secondary replica of AG 2. At this stage, no databases are available in AG 2. This is the starting point for setting up the distributed availability group.	
Create databases with no recovery option on AG 2.	Back up databases on the on-premises availability group (AG 1). Restore the databases to both the forwarder and the secondary replica of AG 2 with no recovery option. While restoring the databases , specify a location with enough disk space for the database data files and the log files. At this stage, the databases are in the restoring state. They are not part of AG 2 or the distributed availabil ity group, and they are not synchronizing.	DBA, Developer

Configure the distributed availability group

Task	Description	Skills required
Create the distributed availability group on AG 1.	To create the distributed availability group on AG 1,	DBA, Developer

Task	Description	Skills required
	use the `CREATE AVAILABILITY GROUP` with the `DISTRIBUTED` option. 1. Use `LISTENER_URL` endpoint addresses for AG 1 and AG 2. 2. For `AVAILABILITY-MODE`, use `ASYNCHRONOUS_COMMIT` to avoid network latency, if any. This will not impact the performance of the database. 3. For `FAILOVER_MODE`, use `MANUAL`. It is the only availability mode that works with distributed availability groups. 4. To restore the databases manually on AG 2 and have more control on larger databases, use `MANUAL` for `SEEDING_MODE`.	

Task	Description	Skills required
Create the distributed availability group on AG 2.	To create the distributed availability group on AG 2, use `ALTER AVAILABILITY GROUP` with the `DISTRIBUTED` option. 1. Use `LISTENER_URL` endpoint addresses for AG 1 and AG 2. 2. For `AVAILABILITY-MODE`, use `ASYNCHRONOUS_COMMIT` to avoid network latency, if any. This will not impact the performance of the database. 3. For `FAILOVER_MODE`, use `MANUAL`. It is the only availability mode that works with distributed availability groups. 4. To restore the databases manually on AG 2 and have more control on larger databases, use `MANUAL` for `SEEDING_MODE`. The distributed availability group is created between AG 1 and AG 2. The databases in AG 2 are not yet configured to take part in	DBA, Developer

Task	Description	Skills required
	the data flow from AG 1 to AG 2.	
Add databases to the forwarder and secondary replica on AG 2.	Add the databases to the distributed availability group by using `ALTER DATABASE` with the `SET HADR AVAILABILITY GROUP` option in both the forwarder and the secondary replica on AG 2. This starts asynchronous data flow between databases on AG 1 and AG 2. The global primary takes writes, sends data synchrono usly to the secondary replica on AG 1, and sends data asynchronously to the forwarder on AG 2. The forwarder on AG 2 sends data synchronously to the secondary replica on AG 2.	DBA, Developer

Monitor asynchronous data flow between AG 1 and AG 2

Task	Description	Skills required
Use DMVs and SQL Server logs.	Monitor the status of the data flow between two availabil ity groups by using dynamic management views (DMVs) and SQL Server logs.	DBA, Developer

Task	Description	Skills required
	DMVs that are of interest for monitoring include `sys.dm_hadr_availability_replica_states` and `sys.dm_hadr_automatic_seeding` . For the status of forwarder synchronization, monitor the *synchronized state* in the SQL Server log on the forwarder.	

Perform cutover activities for final migration

Task	Description	Skills required
Stop all traffic to the primary replica.	Stop incoming traffic to the primary replica in AG 1 so that no write activity occurs on the databases and the databases are ready for migration.	App owner, Developer
Change the availability mode of the distributed availability group on AG 1.	On the primary replica, set the availability mode of the distributed availability group to synchronous. After you change the availability mode to synchronous, the data are sent synchronously from the primary replica in AG 1 to the forwarder in AG 2.	DBA, Developer

Task	Description	Skills required
Check the LSNs in both availability groups.	Check the last Log Sequence Numbers (LSNs) in both AG 1 and AG 2. Because no writes are happening in the primary replica in AG 1, the data are synchronized, and last LSNs for both availability groups should match.	DBA, Developer
Update AG 1 to the secondary role.	When you update AG 1 to the secondary role, AG 1 loses the primary replica role and doesn't accept writes, and the data flow between two availability groups stops.	DBA, Developer

Fail over to the second availability group

Task	Description	Skills required
Manually fail over to AG 2.	On the forwarder in AG 2, alter the distributed availability group to allow data loss. Because you already checked and confirmed that the last LSNs on AG 1 and AG 2 match, data loss is not a concern. When you allow data loss on the forwarder in AG 2, the roles of AG 1 and AG 2 change:	DBA, Developer

Task	Description	Skills required
	• AG 2 becomes the availability group with the primary replica and secondary replica. • AG 1 becomes the availability group with the forwarder and secondary replica.	
Change the availability mode of the distributed availability group on AG 2.	On the primary replica in AG 2, change the availability mode to asynchronous. This changes the data movement from AG 2 to AG 1, from synchronous to asynchronous. This step is required to avoid network latency between AG 2 and AG 1, if any, and will not impact the performance of the database.	DBA, Developer
Start sending traffic to the new primary replica.	Update the connection string to use the listener URL endpoint on AG 2 for sending traffic to the databases. AG 2 now accepts writes and sends data to the forwarder in AG 1, along with sending data to its own secondary replica in AG 2. Data moves asynchronously from AG 2 to AG 1.	App owner, Developer

Perform post-cutover activities

Task	Description	Skills required
Drop the distributed availabil ity group on AG 2.	Monitor the migration for the planned amount of time. Then drop the distributed availability group on AG 2 to remove distributed availabil ity group setup between AG 2 and AG 1. This removes the distributed availability group configuration, and data flow from AG 2 to AG 1 stops. At this point, AG 2 is highly available on AWS, with a primary replica that takes writes and a secondary replica in the same availability group.	DBA, Developer
Decommission the on-premis es servers.	Decommission the on-premis es servers in WSFC 1 that are part of AG 1.	Systems administrator, SysOps administrator

Related resources

- [Distributed availability groups]
- [SQL Docs: Distributed availability groups]
- [SQL Docs: Always On availability groups: a high-availability and disaster-recovery solution]

Migrate from Oracle 8i or 9i to Amazon RDS for Oracle using SharePlex and AWS DMS

Ramu Jagini, Amazon Web Services

Summary

This pattern describes how to migrate an on-premises Oracle 8i or 9i database to an Amazon Relational Database Service (Amazon RDS) for Oracle database. You can use this pattern to complete your migration with reduced downtime by using Quest SharePlex for synchronous replication.

You must use an intermediate Oracle database instance for your migration because AWS Database Migration Service (AWS DMS) doesn't support Oracle 8i or 9i as a source environment. You can use SharePlex 7.6.3 to replicate from previous Oracle database versions to later Oracle database versions. The intermediate Oracle database instance is compatible as a target for SharePlex 7.6.3 and supported as a source for AWS DMS or newer releases of SharePlex. This support enables onward replication of data to the Amazon RDS for Oracle target environment.

Consider that several deprecated data types and features can impact a migration from Oracle 8i or 9i to the latest version of Oracle Database. To mitigate this impact, this pattern uses Oracle 11.2.0.4 as an intermediate database version to help optimize the schema code prior to migrating to the Amazon RDS for Oracle target environment.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle 8i or 9i database in an on-premises environment
- Oracle Database 12c Release 2 (12CR2) for staging on Amazon Elastic Compute Cloud (Amazon EC2)
- Quest SharePlex 7.6.3 (commercial grade)

Limitations

- RDS for Oracle limitations

Product versions

- Oracle 8i or 9i for the source database
- Oracle 12CR2 for the staging database (must match the Amazon RDS for Oracle version)
- Oracle 12CR2 or later for the target database (Amazon RDS for Oracle)

Architecture

Source technology stack

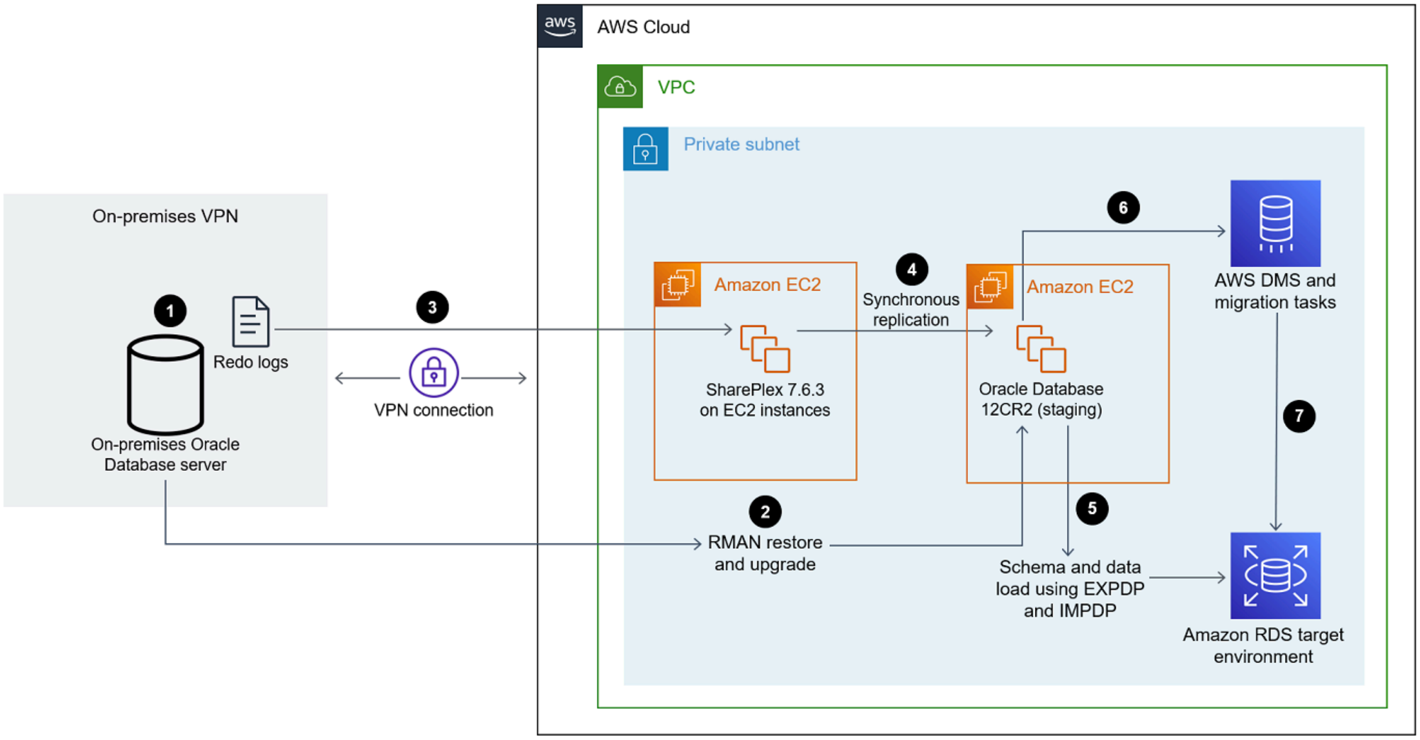
- Oracle 8i or 9i database
- SharePlex

Target technology stack

- Amazon RDS for Oracle

Migration architecture

The following diagram shows how to migrate an Oracle 8i or 9i database from an on-premises environment to an Amazon RDS for Oracle DB instance in the AWS Cloud.



The diagram shows the following workflow:

1. Enable the Oracle source database with archive log mode, force logging, and supplemental logging.
2. Restore the Oracle staging database from the Oracle source database by using Recovery Manager (RMAN) point-in-time recovery and FLASHBACK_SCN.
3. Configure SharePlex to read redo logs from the Oracle source database by using FLASHBACK_SCN (used in RMAN).
4. Start SharePlex replication to synchronize data from the Oracle source database to the Oracle staging database.
5. Restore the Amazon RDS for Oracle target database by using EXPDP and IMPDP with FLASHBACK_SCN.
6. Configure AWS DMS and its source tasks as the Oracle staging database and Amazon RDS for Oracle as the target database by using FLASHBACK_SCN (used in EXPDP).
7. Start AWS DMS tasks to synchronize data from the Oracle staging database to the Oracle target database.

Tools

- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Quest SharePlex is an Oracle-to-Oracle data replication tool for moving data with minimal downtime and no data loss.
- Recovery Manager (RMAN) is an Oracle Database client that performs backup and recovery tasks on your databases. It greatly simplifies backing up, restoring, and recovering database files.
- Data Pump Export helps you upload data and metadata into a set of operating system files called a dump file set. The dump file set can be imported only by the Data Pump Import utility or DBMS_DATAPUMP package.

Epics

Set up SharePlex and the Oracle staging database on Amazon EC2

Task	Description	Skills required
Create an EC2 instance.	1. Create an EC2 instance. 2. Install Oracle 12CR2 on the EC2 instance to serve as the Oracle staging database.	Oracle administration
Prepare the staging database.	Prepare the Oracle staging database for restore as an upgrade on Oracle 12CR2 by taking the RMAN backup from the Oracle 8i or 9i database source environment. For more information, see Oracle 9i Recovery Manager User's Guide and Database	Oracle administration

Task	Description	Skills required
	[Backup and Recovery User's Guide] in the Oracle documentation.	
Configure SharePlex.	Configure the SharePlex source as an on-premises Oracle 8i or 9i database, and configure the target as the Oracle 12CR2 staging database hosted on Amazon EC2.	SharePlex, Oracle administr ation

Set up Amazon RDS for Oracle as your target environment

Task	Description	Skills required
Create an Oracle DB instance.	Create an Amazon RDS for Oracle database, and then connect Oracle 12CR2 to the database. For more information, see [Creating an Oracle DB instance and connecting to a database on an Oracle DB instance] in the Amazon RDS documentation.	DBA
Restore Amazon RDS for Oracle from the staging database.	1. Take an EXPDP backup from the Oracle staging database server by using `FLASHBACK_SCN` . 2. Restore Amazon RDS for Oracle from the staging database.	DBA

Task	Description	Skills required
	For more information, see 54 DBMS_DATAPUMP in the Oracle documentation.	

Set up AWS DMS

Task	Description	Skills required
Create endpoints for the databases.	Create a source endpoint for the Oracle staging database and a target endpoint for the Amazon RDS for Oracle database. For more information, see How do I create source or target endpoints using AWS DMS? in the AWS Knowledge Center.	DBA
Create a replication instance.	Use AWS DMS to launch a replication instance for the Oracle staging database to the Amazon RDS for Oracle database. For more information, see How do I create an AWS DMS replication instance? in the AWS Knowledge Center.	DBA
Create and start replication tasks.	Create AWS DMS replication tasks for change data capture (CDC) by using `FLASHBACK_SCN` from EXPDP (since the	DBA

Task	Description	Skills required
	full load already happened through EXPDP). For more information, see Creating a task in the AWS DMS documentation.	

Cut over to Amazon RDS for Oracle

Task	Description	Skills required
Stop the application workload.	Stop the application servers and its applications during the planned cutover window.	App developer, DBA
Validate the synching of the on-premises Oracle staging database with the EC2 instance.	Confirm that all messages have been posted for replicati on tasks from the SharePlex replication instance to the Oracle staging database on Amazon EC2 by performing a few log switches on the on-premises source database. For more information, see 6.4.2 Switching a Log File in the Oracle documentation.	DBA
Validate the synching of the Oracle staging database with the Amazon RDS for Oracle database.	Confirm that all your AWS DMS tasks have no lag and no errors, and then check the validation state of the tasks.	DBA
Stop the replication of SharePlex and Amazon RDS.	If both the SharePlex and AWS DMS replications are not	DBA

Task	Description	Skills required
	showing any errors, then stop both replications.	
Remap the application to Amazon RDS.	Share the Amazon RDS for Oracle endpoint details with the application server and its applications, and then start the application to resume business operations.	App developer, DBA

Test the AWS target environment

Task	Description	Skills required
Test the Oracle staging database environment on AWS.	1. Test the SharePlex replicati on and verify that there are no sync gaps or replicati on errors on the Oracle staging database. 2. Verify that the applicati on behaves as expected through benchmarks defined in the on-premises environment.	SharePlex, Oracle administr ation
Test the Amazon RDS environment.	1. Verify that all data propagated to Amazon RDS after replication is error free. 2. Point another applicati on to the Amazon RDS DB instance, and then run performance tests to verify expected behavior.	Oracle administration

Task	Description	Skills required
	For more information, see Amazon RDS for Oracle in the Amazon RDS documentation.	

Related resources

- Migrate with confidence
- Amazon EC2
- Amazon RDS for Oracle
- AWS Database Migration Service
- Debugging Your AWS DMS Migrations: What to Do When Things Go Wrong (Part 1)
- Debugging Your AWS DMS Migrations: What to Do When Things Go Wrong (Part 2)
- Debugging Your AWS DMS Migrations: What to Do When Things Go Wrong? (Part 3)
- SharePlex for Database Replication
- SharePlex: database replication for any environment

Monitor Amazon Aurora for instances without encryption

Mansi Suratwala, Amazon Web Services

Summary

This pattern provides an Amazon Web Services (AWS) CloudFormation template that you can deploy to set up automatic notifications when an Amazon Aurora instance is created without encryption turned on.

Aurora is a fully managed relational database engine that's compatible with MySQL and PostgreSQL. With some workloads, Aurora can deliver up to five times the throughput of MySQL and up to three times the throughput of PostgreSQL without requiring changes to most of your existing applications.

The CloudFormation template creates an Amazon CloudWatch Events event and an AWS Lambda function. The event uses AWS CloudTrail to monitor for any Aurora instance creation or a point in time restoration of an existing instance. The Cloudwatch Events event initiates the Lambda function, which checks whether encryption is enabled. If encryption is not turned on, the Lambda function sends an Amazon Simple Notification Service (Amazon SNS) notification informing you of the violation.

Prerequisites and limitations

Prerequisites

- An active AWS account

Limitations

- This service control works with Amazon Aurora instances only. It does not support other Amazon Relational Database Service (Amazon RDS) instances.
- The CloudFormation template must be deployed for `CreateDBInstance` and `RestoreDBClusterToPointInTime` only.

Product versions

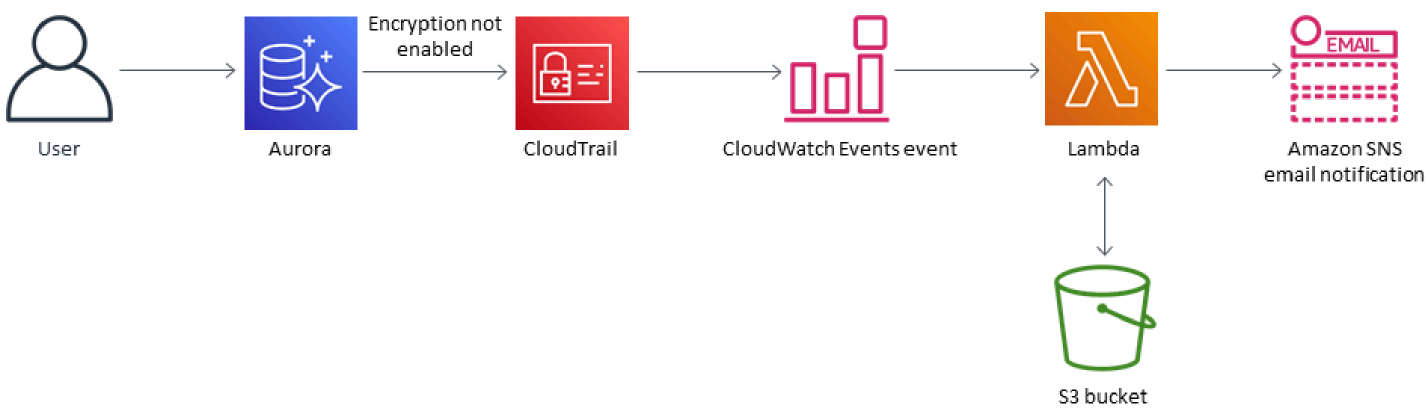
- PostgreSQL versions that are supported in Amazon Aurora
- MySQL versions that are supported in Amazon Aurora

Architecture

Target technology stack

- Amazon Aurora
- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon SNS

Target architecture



Automation and scale

You can use the CloudFormation template multiple times for different Regions and accounts. You need to run it only once in each Region or account.

Tools

Tools

- [Amazon Aurora](#) – Amazon Aurora is a fully managed relational database engine that's compatible with MySQL and PostgreSQL.
- [AWS CloudTrail](#) – AWS CloudTrail helps you manage governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, a role, or an AWS service are recorded as events in CloudTrail.

- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in AWS resources.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery using Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

Code

A .zip file of the project is available as an attachment.

Epics

Create the S3 bucket for the Lambda script

Task	Description	Skills required
Define the S3 bucket.	Open the Amazon S3 console, and choose or create an S3 bucket. This S3 bucket will host the Lambda code .zip file. Your S3 bucket needs to be in the same Region as Aurora. The S3 bucket name cannot contain leading slashes.	Cloud architect

Upload the Lambda code to the S3 bucket

Task	Description	Skills required
Upload the Lambda code.	Upload the Lambda code .zip file provided in the *Attachments* section to the S3 bucket that you defined.	Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Deploy the CloudFormation template.	On the CloudFormation console, deploy the `RDS_Aurora_Encryption_At_Rest.yml` CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide values for the template parameters.	Cloud architect

Complete the parameters in the CloudFormation template

Task	Description	Skills required
Provide the S3 bucket name.	Enter the name of the S3 bucket that you created or chose in the first epic.	Cloud architect
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example,	Cloud architect

Task	Description	Skills required
	`<directory>/<file-name>.zip`).	
Provide an email address.	Provide an active email address to receive Amazon SNS notifications.	Cloud architect
Define the logging level.	Define the logging level and frequency for your Lambda function. `Info` designates detailed informational messages on the application's progress. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the template successfully deploys, it sends a subscription email message to the email address provided. To receive notifications, you must confirm this email subscription.	Cloud architect

Related resources

- [Creating an S3 bucket]

- Uploading files to an S3 bucket
- Creating an Amazon Aurora DB cluster
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Monitor Oracle GoldenGate logs by using Amazon CloudWatch

Chithra Krishnamurthy, Amazon Web Services

Summary

Oracle GoldenGate provides real-time replication between Amazon Relational Database Service (Amazon RDS) for Oracle databases, or between Oracle databases hosted on Amazon Elastic Compute Cloud (Amazon EC2). It supports both unidirectional and bidirectional replication.

When you use GoldenGate for replication, monitoring is critical to verify that the GoldenGate process is up and running, to make sure that the source and target databases are in sync.

This pattern explains the steps to implement Amazon CloudWatch monitoring for a GoldenGate error log, and how to set alarms to send notifications for specific events such as `STOP` or `ABEND` so you can take appropriate actions to resume replication quickly.

Prerequisites and limitations

Prerequisites

- GoldenGate installed and configured on an EC2 instance, so you can set up CloudWatch monitoring on those EC2 instances. If you want to monitor GoldenGate across AWS Regions for bidirectional replication, you must install the CloudWatch agent in each EC2 instance where the GoldenGate process is running.

Limitations

- This pattern explains how to monitor the GoldenGate process by using CloudWatch. CloudWatch doesn't monitor replication lag or data synchronization issues during replication. You must run separate SQL queries to monitor replication lag or data-related errors, as explained in the [GoldenGate documentation](https://docs.oracle.com/en/middleware/goldengate/).

Product versions

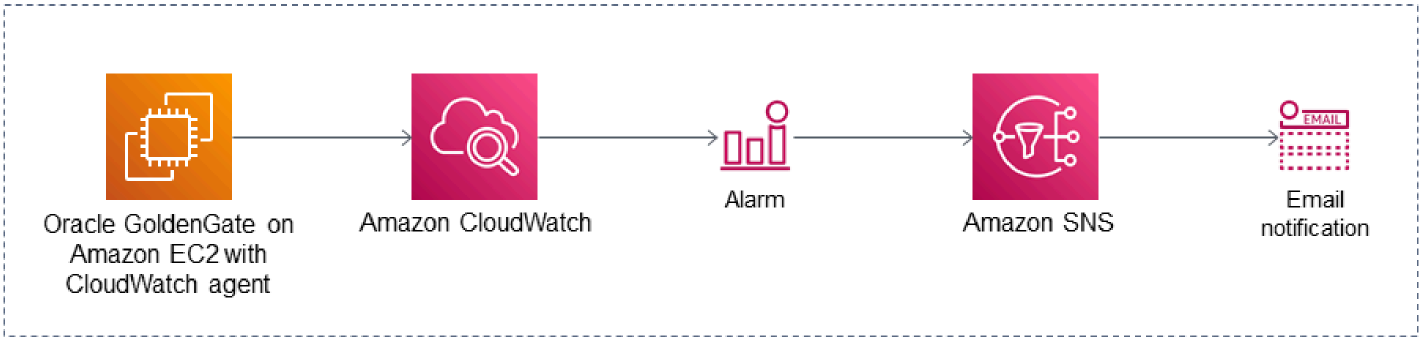
- This document is based on the implementation of Oracle GoldenGate 19.1.0.0.4 for Oracle on Linux x86-64. However, this solution is applicable to all major versions of GoldenGate.

Architecture

Target technology stack

- GoldenGate binaries for Oracle installed on an EC2 instance
- Amazon CloudWatch
- Amazon Simple Notification Service (Amazon SNS)

Target architecture



Tools

AWS services

- [Amazon CloudWatch](#) is a monitoring service that is used in this pattern to monitor GoldenGate error logs.
- [Amazon SNS](#) is a message notification service that is used in this pattern to send email notifications.

Other tools

- [Oracle GoldenGate](#) is a data replication tool that you can use for Amazon RDS for Oracle databases or Oracle databases that are hosted on Amazon EC2.

High-level implementation steps

1. Create an AWS Identity and Access Management (IAM) role for the CloudWatch agent.
2. Attach the IAM role to the EC2 instance where GoldenGate error logs are generated.
3. Install the CloudWatch agent on the EC2 instance.

4. Configure the CloudWatch agent configuration files: `awscli.conf` and `awslogs.conf`.
5. Start the CloudWatch agent.
6. Create metric filters in the log group.
7. Set up Amazon SNS.
8. Create an alarm for the metric filters. Amazon SNS sends email alerts when those filters capture events.

For detailed instructions, see the next section.

Epics

Step 1. Create an IAM role for the CloudWatch agent

Task	Description	Skills required
Create the IAM role.	Access to AWS resources requires permissions, so you create IAM roles to include the permissions necessary for each server to run the CloudWatch agent. To create the IAM role: 1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/. 2. In the navigation pane, choose **Roles**, and then choose **Create role**. 3. For **Trusted entity type**, choose **AWS service**.	AWS general

Task	Description	Skills required
	4. For **Common use cases**, choose **EC2**, and then choose **Next**. 5. In the list of policies, select the check box next to **CloudWatchAgentServerPolicy**. If necessary, use the search box to find the policy. 6. Choose **Next**. 7. For **Role name**, enter a name for your new role, such as `goldengate-cw-monitoring-role` or another name that you prefer. 8. (Optional) For **Role description**, enter a description. 9. Confirm that **CloudWatchAgentServerPolicy** appears under **Policy name**. 10. (Optional) Add one or more tags (key-value pairs) to organize, track, or control access for this role, and then choose **Create role**.	

Step 2. Attach the IAM role to the GoldenGate EC2 instance

Task	Description	Skills required
Attach the IAM role to the EC2 instance where GoldenGate error logs are generated.	The error logs generated by GoldenGate have to be populated to CloudWatch and monitored, so you need to attach the IAM role you created in step 1 to the EC2 instance where GoldenGate is running. To attach an IAM role to an instance: 1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2. In the navigation pane, choose **Instances**, and then find the instance where GoldenGate is running. 3. Select the instance, and then choose **Actions**, **Security**, **Modify IAM role**. 4. Select the IAM role created in the first step to attach to your instance, and then choose **Save**.	AWS general

Steps 3-5. Install and configure the CloudWatch agent on the Goldengate EC2 instance

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install the CloudWatch agent on the GoldenGate EC2 instance.</td><td>To install the agent, run the command:<pre><code>sudo yum install -y
 awslogs</code></pre></td><td>AWS general</td></tr>
<tr><td>Edit the agent configuration files.</td><td>1. Run the following command.<pre><code>sudo su -</code></pre>2. Edit this file to update the AWS Region as necessary.<pre><code>cat /etc/awslogs/conf
[plugins]
cwlogs = cwlogs
[default]
region = us-east-1</code></pre>3. Edit the <code>/etc/awslogs/awslogs.conf</code> file to update the file name, log group name, and the date/time format. You must specify the date/time to match the date format in <code>ggserror.log</code> ; otherwise, the log stream won't flow into CloudWatch. For example:<pre><code>datetime_format = %Y-
%m-%dT%H:%M:%S%z</code></pre></td><td>AWS general</td></tr>
</table>

Task	Description	Skills required
	`file = /u03/oracle/oragg/ggserr.log log_group_name = goldengate_monitor`	
Start the CloudWatch agent.	To start the agent, use the following command. `$ sudo service awslogsd start` After you start the agent, you can view the log group in the CloudWatch console. The log stream will have the contents of the file.	AWS general

Step 6. Create metric filters for the log group

Task	Description	Skills required
Create metric filters for the keywords ABEND and STOPPED.	When you create metric filters for the log group, whenever the filters are identified in the error log, it starts an alarm and sends an email notificat ion based on the Amazon SNS configuration. To create metric filters: 1. Open the CloudWatc h console at https://console.aws.amazon.com/cloudwatch/.	CloudWatch

Task	Description	Skills required
	2. Choose the name of the log group. 3. Choose **Actions**, and then choose **Create metric filter**. 4. For the **Filter pattern**, specify a pattern such as `ABEND`. 5. Choose **Next**, and then enter a name for your metric filter. 6. Under **Metric details**, for **Metric namespace**, enter a name for the CloudWatch namespace where the metric will be published. If the namespace doesn't already exist, make sure that **Create new** is selected. 7. For **Metric value**, enter 1, because your metric filter is counting occurrences of the keywords in the filter. 8. Set **Unit** to **None**. 9. Choose **Create metric filter**. You can find the metric filter that you created from the navigation pane. 10. Create another metric filter for the `STOPPED` pattern. Within one log group, you	

Task	Description	Skills required
	can create multiple metric filters and set alarms individually.	

Step 7. Set up Amazon SNS

Task	Description	Skills required
Create an SNS topic.	In this step, you configure Amazon SNS to create alarms for the metric filters. To create an SNS topic: 1. Sign in to the Amazon SNS console at https://console.aws.amazon.com/sns/home. 2. In the **Create topic** box, enter a topic name such as `goldengate-alert` , and then choose **Next step**. 3. For **Type**, choose **Standard**. 4. Scroll to the end of the form and choose **Create topic**. The console opens the new topic's **Details** page.	Amazon SNS
Create a subscription.	To create a subscription to the topic: 1. In the left navigation pane, choose **Subscriptions**.	Amazon SNS

Task	Description	Skills required
	2. On the **Subscriptions** page, choose **Create subscription**. 3. On the **Create subscription** page, choose the **Topic ARN** field to see a list of the topics in your AWS account. 4. Choose the topic that you created in the previous step. 5. For **Protocol**, choose **Email**. 6. For **Endpoint**, enter an email address that can receive notifications. 7. Choose **Create subscription**. The console opens the new subscription's **Details** page. 8. Check your email inbox for a message from AWS Notifications, and then choose **Confirm subscription** in the email. Amazon SNS opens your web browser and displays a subscription confirmation with your subscription ID.	

Step 8. Create an alarm to send notifications for the metric filters

Task	Description	Skills required
Create an alarm for the SNS topic.	To create an alarm based on a log group-metric filter: 1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/. 2. From the navigation pane, choose **Logs**, and then choose **Log groups**. 3. Choose the log group that includes your metric filter. 4. Choose **Metric filters**. 5. In the **Metric filters** tab, select the check box for the metric filter that you want to base your alarm on. 6. Choose **Create alarm**. 7. For **Conditions**, specify the following in each section: • For **Threshold type**, choose **Static** . • For **Whenever <metric-name> is . . .**, choose **Greater**. • For **than . . .**, specify **0**. 8. Choose **Next**. 9. Under **Notification**: • For **Alarm state trigger**, choose **In alarm**.	CloudWatch

Task	Description	Skills required
	• For **Send notification to following SNS topic**, choose **Select an existing topic**. • In the email box, select the Amazon SNS topic that you created in the previous step. 10.Choose **Next**. 11.For **Name and description**, enter a name and description for your alarm. **Note** For the description, you can specify the instance name so that the notification email is descriptive. 12.For **Preview and create**, check that your configuration is correct, and then choose **Create alarm**. After these steps, whenever these patterns are detected in the GoldenGate error log file (`ggserr.log`) that you are monitoring, you will get an email notification.	

Troubleshooting

Issue	Solution
The log stream from the GoldenGate error log doesn't flow into CloudWatch.	Check the `/etc/awslogs/awslogs.conf` file to verify the file name, log group name, and the date/time format. You must specify the date/time to match the date format in `ggserror.log` . Otherwise, the log stream won't flow into CloudWatch.

Related resources

- Amazon CloudWatch documentation
- Collecting metrics and logs with the CloudWatch agent
- Amazon SNS documentation

Replatform Oracle Database Enterprise Edition to Standard Edition 2 on Amazon RDS for Oracle

Lanre (Lan-Ray) showunmi and Tarun Chawla, Amazon Web Services

Summary

Oracle Database Enterprise Edition (EE) is a popular choice for running applications in many enterprises. In some cases, however, applications use few or no Oracle Database EE features, so there is a lack of justification for incurring huge licensing costs. You can achieve cost savings by downgrading such databases to Oracle Database Standard Edition 2 (SE2) when you migrate to Amazon RDS.

This pattern describes how to downgrade from Oracle Database EE to Oracle Database SE2 when migrating from on premises to Amazon RDS for Oracle. The steps presented in this pattern also apply if your EE Oracle database is already running on Amazon RDS or on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

For more information, see the AWS Prescriptive Guidance guide on how to Evaluate downgrading Oracle databases to Standard Edition 2 on AWS.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Oracle Database Enterprise Edition
- A client tool, such as Oracle SQL Developer or SQL*Plus, for connecting to and running SQL commands on Oracle database
- Database user for performing the assessment; for example, one of the following:
  - User with sufficient privileges for running AWS Schema Conversion Tool (AWS SCT) assessment
  - User with sufficient privileges to run SQL queries on Oracle database dictionary tables
- Database user for performing database migration; for example, one of the following:
  - User with sufficient privileges for running AWS Database Migration Service (AWS DMS)
  - User with sufficient privileges for performing Oracle Data Pump export and import
  - User with sufficient privileges for running Oracle GoldenGate

Limitations

- Amazon RDS for Oracle has a maximum database size. For more information, see Amazon RDS DB instance storage.

Product versions

The general logic described in this document applies to Oracle versions from 9i and later. For supported versions of self-managed and Amazon RDS for Oracle databases, see the AWS DMS documentation.

To identify feature usage in cases where AWS SCT is not supported, run SQL queries on the source database. To migrate from earlier versions of Oracle where AWS DMS and Oracle Data Pump are not supported, use Oracle Export and Import utilities.

For a current list of supported versions and editions, see Oracle on Amazon RDS in the AWS documentation. For details on pricing and supported instance classes, see Amazon RDS for Oracle pricing.

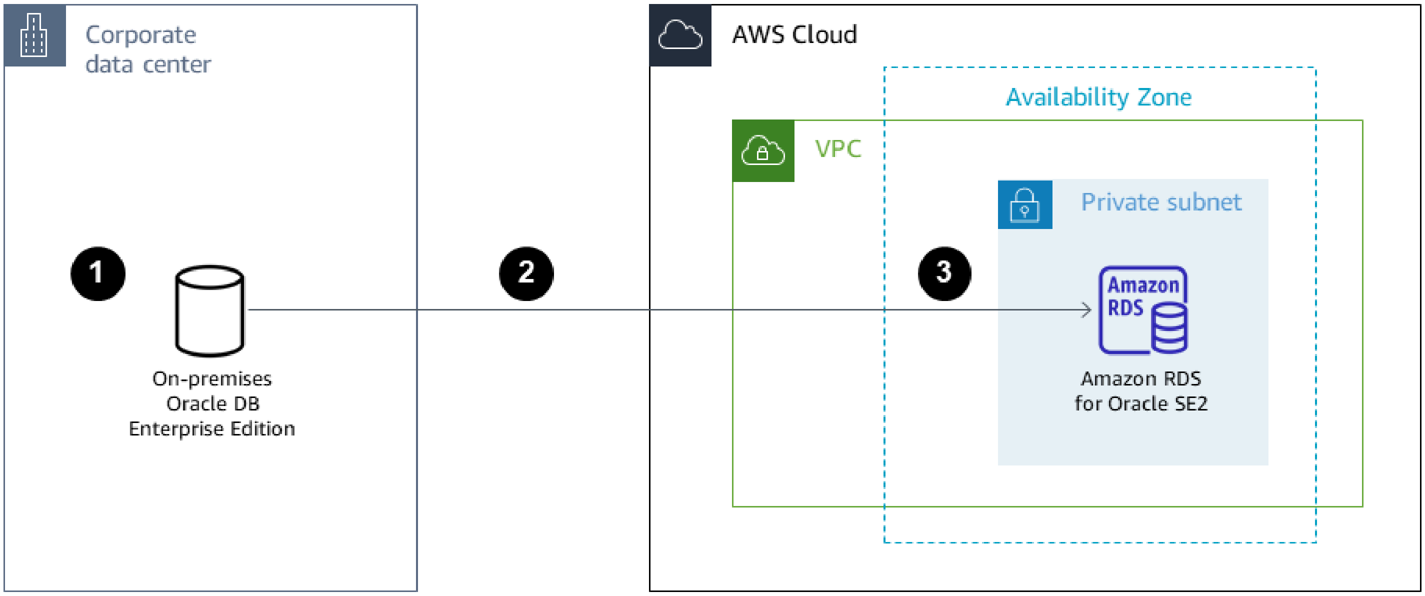
Architecture

Source technology stack

- Oracle Database Enterprise Edition running on premises or on Amazon EC2

Target technology stack using native Oracle tools

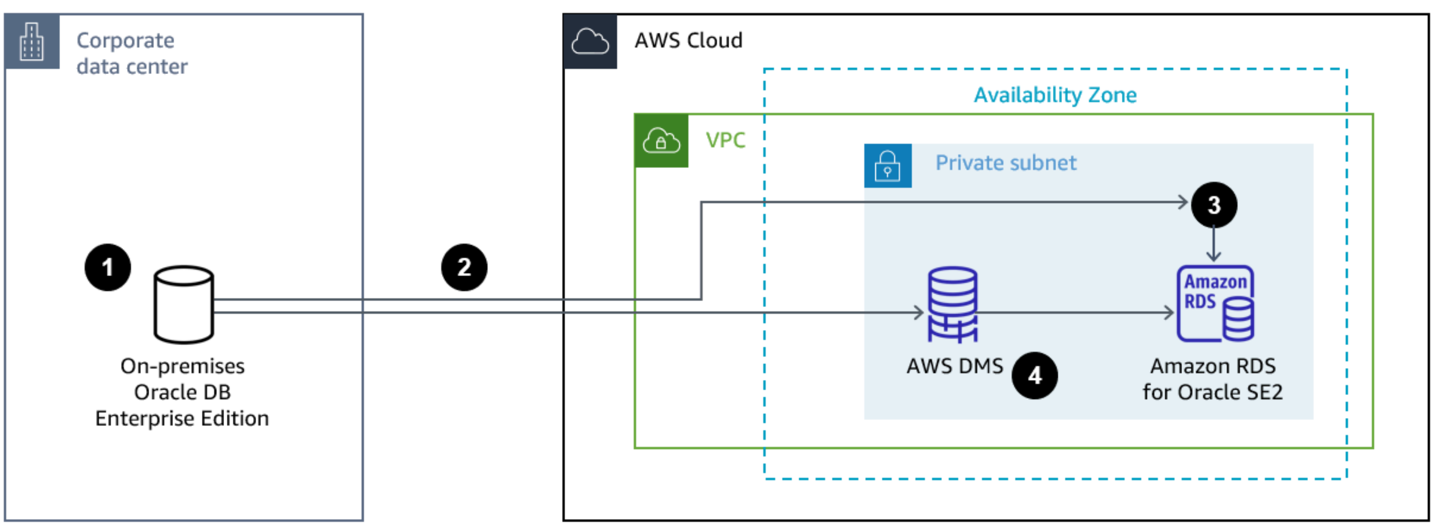
- Amazon RDS for Oracle running Oracle Database SE2



1. Export data by using Oracle Data Pump.
2. Copy dump files to Amazon RDS through a database link.
3. Import dump files to Amazon RDS by using Oracle Data Pump.

Target technology stack using AWS DMS

- Amazon RDS for Oracle running Oracle Database SE2
- AWS DMS



1. Export data by using Oracle Data Pump with FLASHBACK_SCN.
2. Copy dump files to Amazon RDS through a database link.
3. Import dump files to Amazon RDS by using Oracle Data Pump.
4. Use AWS DMS change data capture (CDC).

Tools

AWS services

- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud. This pattern uses Amazon RDS for Oracle.
- AWS SCT provides a project-based user interface to automatically assess, convert, and copy the database schema of your source Oracle database into a format compatible with Amazon RDS for Oracle. AWS SCT enables you to analyze potential cost savings that can be achieved by changing your license type from Enterprise to Standard Edition of Oracle. The **License Evaluation and Cloud Support** section of the AWS SCT report provides detailed information about Oracle features in use so you can make an informed decision while migrating to Amazon RDS for Oracle.

Other tools

- Native Oracle import and export utilities support moving Oracle data in and out of Oracle databases. Oracle offers two types of database import and export utilities: Original Export and Import (for earlier releases) and Oracle Data Pump Export and Import (available in Oracle Database 10g release 1 and later).
- Oracle GoldenGate offers real-time replication capabilities so that you can synchronize your target database after an initial load. This option can help reduce application downtime during go-live.

Epics

Make a pre-migration assessment

Task	Description	Skills required
Validate database requirements for your applications.	Ensure that your applications are certified to run on Oracle Database SE2. Check directly with the software vendor, developer, or application documentation.	App developer, DBA, App owner
Investigate use of EE features directly in the database.	To determine EE feature use, do one of the following: • Generate an AWS SCT assessment report for your Oracle EE database. The report tells you which features from your current EE database should be removed if you want to change license types. • If you have an Oracle Support account, obtain and run the script `options_packs_usage_statistics.sql` in Support document 1317265.1 to generate a report of options and features that are being used on your Oracle database. • Query DBA_FEATURE_USAGE_STATISTICS	App owner, DBA, App developer

Task	Description	Skills required
	to display details of all features that are in use.	
Identify use of EE features for operational activities.	Database or application administrators sometimes rely on EE-only features for operational activities. Common examples include online maintenance activities (index rebuild, table move) and use of parallelism by batch jobs. These dependencies can be mitigated by modifying your operations where possible. Identify the use of these features and make a decision based on cost compared with benefits. Use the [Comparing Oracle Database EE and SE2 features](#) table as a guide to identify features that are available in Oracle Database SE2.	App developer, DBA, App owner

Task	Description	Skills required
Review workload patterns of the EE Oracle database.	Oracle Database SE2 automatically restricts usage to a maximum of 16 CPU threads at any time. If your Oracle EE database is licensed to use the Oracle Diagnostic Pack, use the Automatic Workload Repository (AWR) tool, or DBA_HIST_* views, to analyze database workload patterns to determine whether the maximum limit of 16 CPU threads will negatively impact service levels when you downgrade to SE2. Ensure that your assessmen t covers periods of peak activity, such as end of day, month, or year processing.	App owner, DBA, App developer

Prepare the target infrastructure on AWS

Task	Description	Skills required
Deploy and configure networking infrastructure.	Create a [virtual private cloud (VPC) and subnets](), [security groups](), and [network access control lists]().	AWS administrator, Cloud architect, Network administr ator, DevOps engineer
Provision the Amazon RDS for Oracle SE2 database.	Provision the target [Amazon RDS for Oracle]() SE2 database to meet your applications'	Cloud administrator, Cloud architect, DBA, DevOps engineer, AWS administrator

Task	Description	Skills required
	performance, availability, and security requirements. We recommend Multi-AZ configuration for production workloads. However, to improve migration performance, you can defer enabling Multi-AZ until after data migration.	
Customize the Amazon RDS environment.	Configure custom parameters and options, and enable additional monitoring. For more information, see Best practices for migrating to Amazon RDS for Oracle.	AWS administrator, AWS systems administrator, Cloud administrator, DBA, Cloud architect

Perform the migration dry run and application testing

Task	Description	Skills required
Migrate the data (dry run).	Migrate data from the source Oracle EE database to the Amazon RDS for Oracle SE2 database instance using the approach best suited to your specific environment. Select a migration strategy based on factors such as size, complexity, and the available downtime window. Use one or a combination of the following:	DBA

Task	Description	Skills required
	• Native Oracle tools such as Oracle Data Pump (recommended), Oracle Import-Export utilities, and Oracle GoldenGate. • AWS DMS, using the full load with continuous replication through CDC.	
Validate the target database.	Perform post-migration validation of database storage and code objects. Review migration logs, and fix any identified issues. For more information, see the guide Migrating Oracle databases to the AWS Cloud.	DBA
Test the applications.	Application and database administrators should conduct functional, performance, and operation al tests as appropriate. For more information, see Best practices for migrating to Amazon RDS for Oracle. Lastly, obtain sign-offs on test-results from stakehold ers.	App developer, App owner, DBA, Migration engineer, Migration lead

Cut over

Task	Description	Skills required
Refresh data from Oracle Database EE.	Select a data refresh approach based on the application availability requirement. For more information, see the migration methods in [Strategies for Migrating Oracle Databases to AWS](). For example, you can achieve near-zero downtime by using tools such as Oracle GoldenGate or AWS DMS with ongoing replication. If the downtime window permits, you can perform the final data cutover using offline methods such as Oracle Data Pump or Original Export-Im port utilities.	App owner, Cutover lead, DBA, Migration engineer, Migration lead
Point applications to the target database instance.	Update connection parameter s in applications and other clients to point to the Amazon RDS for Oracle SE2 database.	App developer, App owner, Migration engineer, Migration lead, Cutover lead
Perform post-migration activities.	Perform post data migration tasks such as enabling Multi-AZ, data validation, and other checks.	DBA, Migration engineer

Task	Description	Skills required
Perform post-cutover monitoring.	Use tools such as Amazon CloudWatch and Amazon RDS Performance Insights to monitor the Amazon RDS for Oracle SE2 database.	App developer, App owner, AWS administrator, DBA, Migration engineer

Related resources

AWS Prescriptive Guidance

- Migrating Oracle databases to the AWS Cloud (guide)
- Evaluate downgrading Oracle databases to Standard Edition 2 on AWS (guide)
- Migrate an on-premises Oracle database to Amazon RDS for Oracle (pattern)
- Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump (pattern)

Blog posts

- Migrating Oracle databases with near-zero downtime using AWS DMS
- Analyzing performance management in Oracle SE using Amazon RDS for Oracle
- Managing your SQL plan in Oracle SE with Amazon RDS for Oracle
- Implementing table partitioning in Oracle Standard Edition: Part 1

Replicate mainframe databases to AWS by using Precisely Connect

Lucio Pereira, Sayantan Giri, and Balaji Mohan, Amazon Web Services

Summary

This pattern outlines steps for replicating data from mainframe databases to Amazon data stores in near real time by using Precisely Connect. It implements an event-based architecture with Amazon Managed Streaming for Apache Kafka (Amazon MSK) and custom database connectors in the cloud to improve scalability, resilience, and performance.

Precisely Connect is a replication tool that captures data from legacy mainframe systems and integrates it into cloud environments. Data is replicated from mainframes to AWS through change data capture (CDC) by using near real-time message flows with low-latency and high-throughput heterogeneous data pipelines.

This pattern also covers a disaster recovery strategy for resilient data pipelines with multi-Region data replication and failover routing.

Prerequisites and limitations

Prerequisites

- An existing mainframe database—for example, IBM DB2, IBM Information Management System (IMS), or Virtual Storage Access Method (VSAM)—that you want to replicate to the AWS Cloud
- An active [AWS account](https://aws.amazon.com/account/)
- [AWS Direct Connect](https://aws.amazon.com/directconnect/) or [AWS Virtual Private Network (AWS VPN](https://aws.amazon.com/vpn/)) from your corporate environment to AWS
- A [virtual private cloud](https://aws.amazon.com/vpc/) with a subnet that is reachable by your legacy platform

Architecture

Source technology stack

A mainframe environment that includes at least one of the following databases:

- IBM IMS database
- IBM DB2 database

- VSAM files

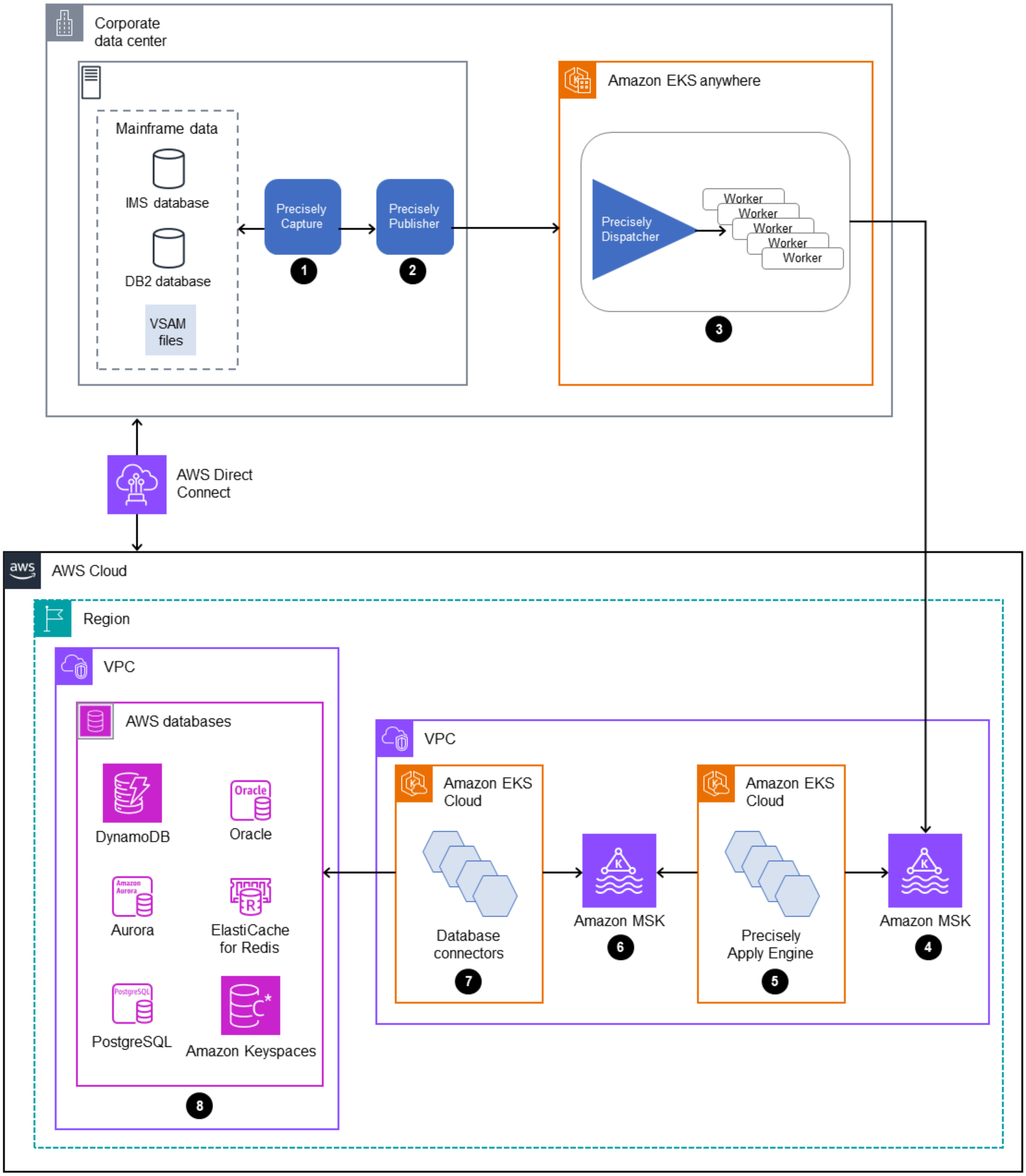
Target technology stack

- Amazon MSK
- Amazon Elastic Kubernetes Service (Amazon EKS) and Amazon EKS Anywhere
- Docker
- An AWS relational or NoSQL database such as the following:
  - Amazon DynamoDB
  - Amazon Relational Database Service (Amazon RDS) for Oracle, Amazon RDS for PostgreSQL, or Amazon Aurora
  - Amazon ElastiCache for Redis
  - Amazon Keyspaces (for Apache Cassandra)

Target architecture

Replicating mainframe data to AWS databases

The following diagram illustrates the replication of mainframe data to an AWS database such as DynamoDB, Amazon RDS, Amazon ElastiCache, or Amazon Keyspaces. The replication occurs in near real time by using Precisely Capture and Publisher in your on-premises mainframe environment, Precisely Dispatcher on Amazon EKS Anywhere in your on-premises distributed environment, and Precisely Apply Engine and database connectors in the AWS Cloud.



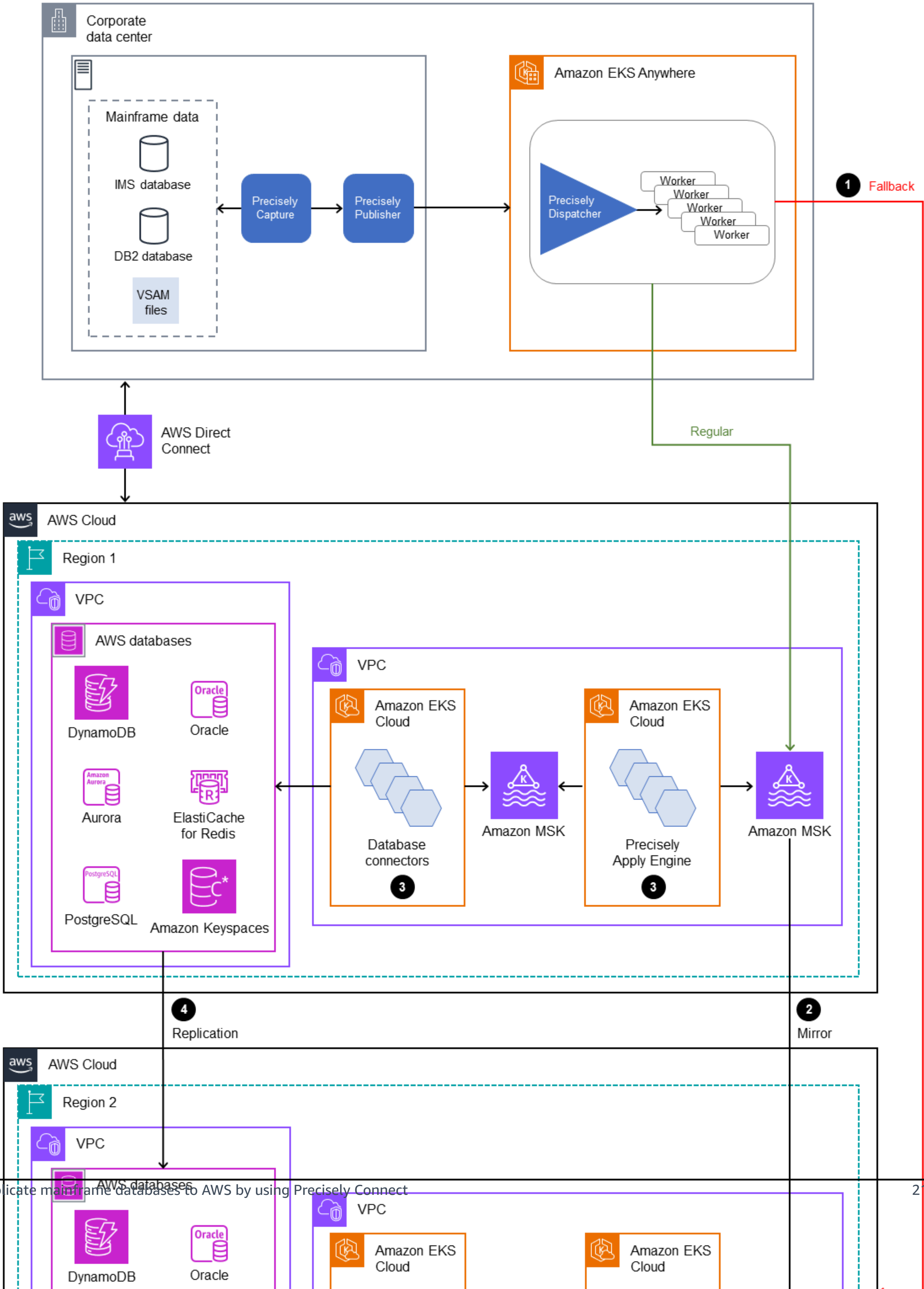
The diagram shows the following workflow:

1. Precisely Capture gets mainframe data from CDC logs and maintains the data in internal transient storage.
2. Precisely Publisher listens for changes in the internal data storage and sends CDC records to Precisely Dispatcher through a TCP/IP connection.
3. Precisely Dispatcher receives the CDC records from Publisher and sends them to Amazon MSK. Dispatcher creates Kafka keys based on the user configuration and multiple worker tasks to push data in parallel. Dispatcher sends an acknowledgment back to Publisher when records have been stored in Amazon MSK.
4. Amazon MSK holds the CDC records in the cloud environment. The partition size of topics depends on your transaction processing system (TPS) requirements for throughput. The Kafka key is mandatory for further transformation and transaction ordering.
5. The Precisely Apply Engine listens to the CDC records from Amazon MSK and transforms the data (for example, by filtering or mapping) based on target database requirements. You can add customized logic to the Precisely SQD scripts. (SQD is Precisely's proprietary language.) The Precisely Apply Engine transforms each CDC record to Apache Avro or JSON format and distributes it to different topics based on your requirements.
6. The target Kafka topics hold CDC records in multiple topics based on the target database, and Kafka facilitates transaction ordering based on the defined Kafka key. The partition keys align with the corresponding partitions to support a sequential process.
7. Database connectors (customized Java applications) listen to the CDC records from Amazon MSK and store them in the target database.
8. You can select a target database based on your requirements. This pattern supports both NoSQL and relational databases.

Disaster recovery

Business continuity is key to your organization's success. The AWS Cloud provides capabilities for high availability (HA) and disaster recovery (DR), and supports your organization's failover and fallback plans. This pattern follows an active/passive DR strategy and provides high-level guidance for implementing a DR strategy that meets your RTO and RPO requirements.

The following diagram illustrates the DR workflow.

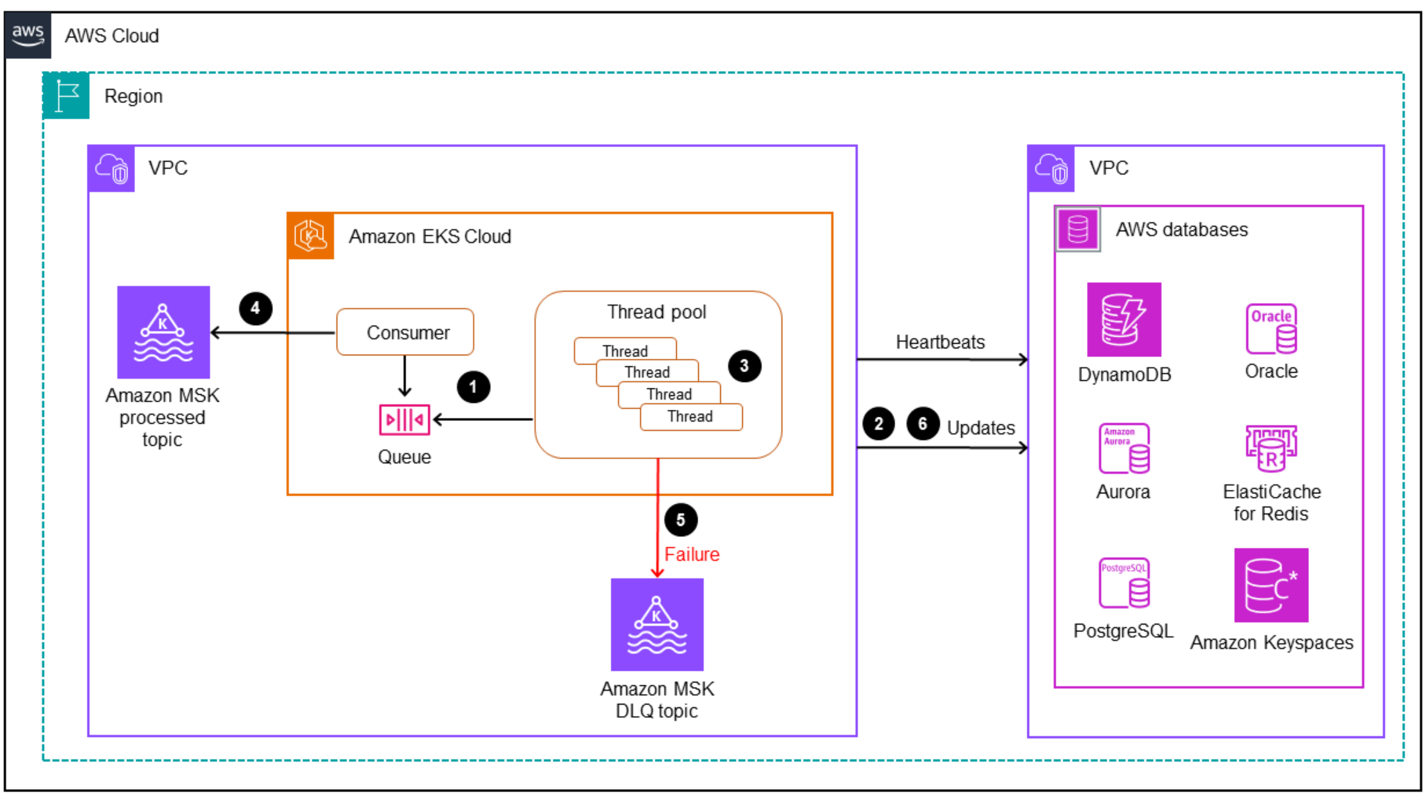

Corporate data center
Mainframe data
IMS database
DB2 database
VSAM files
Precisely Capture
Precisely Publisher
Amazon EKS Anywhere
Precisely Dispatcher
Worker
Worker
Worker
Worker
Worker
1
Fallback
Regular
AWS Direct Connect
AWS Cloud
Region 1
VPC
AWS databases
DynamoDB
Oracle
Aurora
ElastiCache for Redis
PostgreSQL
Amazon Keyspaces
VPC
Amazon EKS Cloud
Database connectors
3
Amazon MSK
Amazon EKS Cloud
Precisely Apply Engine
3
Amazon MSK
4
Replication
2
Mirror
AWS Cloud
Region 2
VPC
AWS databases
DynamoDB
Oracle
VPC
Amazon EKS Cloud
Amazon EKS Cloud

The diagram shows the following:

1. A semi-automated failover is required if any failure happens in AWS Region 1. In the case of failure in Region 1, the system must initiate routing changes to connect Precisely Dispatcher to Region 2.
2. Amazon MSK replicates data through mirroring between Regions, For this reason, during failover, the Amazon MSK cluster in Region 2 has to be promoted as the primary leader.
3. The Precisely Apply Engine and database connectors are stateless applications that can work in any Region.
4. Database synchronization depends on the target database. For example, DynamoDB can use global tables, and ElastiCache can use global datastores.

Low-latency and high-throughput processing through database connectors

Database connectors are critical components in this pattern. Connectors follow a listener-based approach to collect data from Amazon MSK and send transactions to the database through high-throughput and low-latency processing for mission-critical applications (tiers 0 and 1). The following diagram illustrates this process.



This pattern supports the development of a customized application with single-threaded consumption through a multithreaded processing engine.

1. The connector main thread consumes CDC records from Amazon MSK and sends them to the thread pool for processing.
2. Threads from the thread pool process CDC records and send them to the target database.
3. If all threads are busy, the CDC records are kept on hold by the thread queue.
4. The main thread waits to get all the records cleared from the thread queue and commits offsets into Amazon MSK.
5. The child threads handle failures. If failures happen during processing, the failed messages are sent to the DLQ (dead letter queue) topic.
6. The child threads initiate conditional updates (see Condition expressions in the DynamoDB documentation), based on the mainframe timestamp, to avoid any duplication or out-of-order updates in the database.

For information about how to implement a Kafka consumer application with multi-threading capabilities, see the blog post Multi-Threaded Message Consumption with the Apache Kafka Consumer on the Confluent website.

Tools

AWS services

- Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that helps you build and run applications that use Apache Kafka to process streaming data.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without having to install or maintain your own Kubernetes control plane or nodes.
- Amazon EKS Anywhere helps you deploy, use, and manage Kubernetes clusters that run in your own data centers.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon ElastiCache helps you set up, manage, and scale distributed in-memory cache environments in the AWS Cloud.

- [Amazon Keyspaces (for Apache Cassandra)](#) is a managed database service that helps you migrate, run, and scale your Cassandra workloads in the AWS Cloud.

Other tools

- [Precisely Connect](#) integrates data from legacy mainframe systems such as VSAM datasets or IBM mainframe databases into next-generation cloud and data platforms.

Best practices

- Find the best combination of Kafka partitions and multi-threaded connectors to balance optimal performance and cost. Multiple Precisely Capture and Dispatcher instances can increase cost because of higher MIPS (million instructions per second) consumption.
- Avoid adding data manipulation and transformation logic to the database connectors. For this purpose, use the Precisely Apply Engine, which provides processing times in microseconds.
- Create periodic request or health check calls to the database (*heartbeats*) in database connectors to warm up the connection frequently and reduce latency.
- Implement thread pool validation logic to understand the pending tasks in the thread queue and wait for all threads to be completed before the next Kafka polling. This helps avoid data loss if a node, container, or process crashes.
- Expose latency metrics through health endpoints to enhance observability capabilities through dashboards and tracing mechanisms.

Epics

Prepare the source environment (on premises)

Task	Description	Skills required
Set up the mainframe process (batch or online utility) to start the CDC process from mainframe databases.	1. Identify the mainframe environment. 2. Identify the mainframe databases that will be involved in the CDC process.	Mainframe engineer

Task	Description	Skills required
	3. In the mainframe environment, develop a process that launches the CDC tool to capture changes in the source database. For instructions, see your mainframe documentation. 4. Document the CDC process, including the configuration. 5. Deploy the process in both test and production environments.	
Activate mainframe database log streams.	1. Configure log streams in the mainframe environment to capture CDC logs. For instructions, see your mainframe documentation. 2. Test the log streams to ensure that they capture the necessary data. 3. Deploy the log streams in test and production environments.	Mainframe DB specialist

Task	Description	Skills required
Use the Capture component to capture CDC records.	1. Install and configure the Precisely Capture component in the mainframe environment. For instructions, see the [Precisely documentation](). 2. Test the configuration to ensure that the Capture component works correctly . 3. Set up a replication process to replicate the captured CDC records through the Capture component. 4. Document the Capture configuration for each source database. 5. Develop a monitoring system to ensure that the Capture component collects logs properly over time. 6. Deploy the installation and configurations in the test and production environme nts.	Mainframe engineer, Precisely Connect SME

Task	Description	Skills required
Configure the Publisher component to listen to the Capture component.	1. Install and configure the Precisely Publisher component in the mainframe environment. For instructions, see the [Precisely documentation](). 2. Test the configuration to ensure that the Publisher component works correctly . 3. Set up a replication process to publish the CDC records to the Precisely Dispatcher component from Publisher. 4. Document the Publisher configuration. 5. Develop a monitoring system to ensure that the Publisher component works properly over time. 6. Deploy the installation and configurations in the test and production environme nts.	Mainframe engineer, Precisely Connect SME

Task	Description	Skills required
Provision Amazon EKS Anywhere in the on-premises distributed environment.	1. Install Amazon EKS Anywhere on the on-premises infrastructure and ensure that it is properly configured. For instructi ons, see the [Amazon EKS Anywhere](https://anywhere.eks.amazonaws.com/docs/) documentation. 2. Set up a secure network environment for the Kubernetes cluster, including firewalls. 3. Implement and test the sample application deployment to the Amazon EKS Anywhere cluster. 4. Implement automatic scaling capabilities for the cluster. 5. Develop and implement backup and disaster recovery procedures.	DevOps engineer

Task	Description	Skills required
Deploy and configure the Dispatcher component in the distributed environment to publish the topics in the AWS Cloud.	1. Configure and container ize the Precisely Dispatche r component. For instructi ons, see the [Precisely documentation](). 2. Deploy the Dispatcher Docker image into the on-premises Amazon EKS Anywhere environment. 3. Set up a secure connectio n between the AWS Cloud and Dispatcher. 4. Develop a monitoring system to ensure that the Dispatcher component works properly over time. 5. Deploy the installation and configurations in the test and production environme nts.	DevOps engineer, Precisely Connect SME

Prepare the target environment (AWS)

Task	Description	Skills required
Provision an Amazon EKS cluster in the designated AWS Region.	1. Sign in to your AWS account and configure it to ensure that the necessary permissions are in place to create and manage the Amazon EKS cluster. 2. Create a virtual private cloud (VPC) and subnets in	DevOps engineer, Network administrator

Task	Description	Skills required
	the selected AWS Region. For instructions, see the Amazon EKS documentation. 3. Create and configure the necessary network security groups to allow communications between the Amazon EKS cluster and other resources in the VPC. For more information, see the Amazon EKS documentation. 4. Create the Amazon EKS cluster and configure it with the correct node group size and instance types. 5. Validate the Amazon EKS cluster by deploying a sample application.	

Task	Description	Skills required
Provision an MSK cluster and configure applicable Kafka topics.	1. Configure your AWS account to ensure that the necessary permissions are in place to create and manage the MSK cluster. 2. Create and configure the necessary network security groups to allow communications between the MSK cluster and other resources in the VPC. For more information, see the [Amazon VPC documentation](#). 3. Create the MSK cluster and configure it to include the Kafka topics that will be used by the application. For more information, see the [Amazon MSK documentation](#).	DevOps engineer, Network administrator

Task	Description	Skills required
Configure the Apply Engine component to listen to the replicated Kafka topics.	1. Configure and containerize the Precisely Apply Engine component. 2. Deploy the Apply Engine Docker image into the Amazon EKS cluster in your AWS account. 3. Set up the Apply Engine to listen to MSK topics. 4. Develop and configure a SQD script in the Apply Engine to handle filtering and transformation. For more information, see the Precisely documentation. 5. Deploy the Apply Engine in test and production environments.	Precisely Connect SME

Task	Description	Skills required
Provision DB instances in the AWS Cloud.	1. Configure your AWS account to ensure that the necessary permissions are in place to create and manage DB clusters and tables. For instructions, see the AWS documentation for the AWS database service you want to use. (See the Resources section for links.) 2. Create a VPC and subnets in the selected AWS Region. 3. Create and configure the necessary network security groups to allow communications between the DB instances and other resources in the VPC. 4. Create the databases and configure them to include the tables that the application will use. 5. Design and validate the database schemas.	Data engineer, DevOps engineer

Task	Description	Skills required
Configure and deploy database connectors to listen to the topics published by the Apply Engine.	1. Design database connector s to connect the Kafka topics with the AWS databases that you created in the earlier steps. 2. Develop the connector s based on the target database. 3. Configure the connectors to listen to the Kafka topics that are published by the Apply Engine. 4. Deploy the connectors into the Amazon EKS cluster.	App developer, Cloud architect, Data engineer

Set up business continuity and disaster recovery

Task	Description	Skills required
Define disaster recovery goals for your business applications.	1. Define the RPO and RTO goals for CDC pipelines based on your business needs and impact analysis. 2. Define the communica tion and notification procedures to ensure that all stakeholders are aware of the disaster recovery plan. 3. Determine the budget and resources needed to implement the disaster recovery plan.	Cloud architect, Data engineer, App owner

Task	Description	Skills required
	4. Document the disaster recovery goals, including the RPO and RTO goals.	
Design disaster recovery strategies based on defined RTO/RPO.	1. Determine the most appropriate disaster recovery strategies for CDC pipelines based on your criticality and recovery requirements. 2. Define the disaster recovery architecture and topology. 3. Define the failover and failback procedures for CDC pipelines to ensure that they can be quickly and seamlessly switched to the backup Region. 4. Document the disaster recovery strategies and procedures and ensure that all stakeholders have a clear understanding of the design.	Cloud architect, Data engineer

Task	Description	Skills required
Provision disaster recovery clusters and configurations.	1. Provision a secondary AWS Region for disaster recovery. 2. In the secondary AWS Region, create an environment that's identical to the primary AWS Region. 3. Configure Apache Kafka MirrorMaker between the primary and secondary Regions. For more information, see the [Amazon MSK documentation](#). 4. Configure standby applications in the secondary Region. 5. Configure database replications between the primary and secondary Regions.	DevOps engineer, Network administrator, Cloud architect

Task	Description	Skills required
Test the CDC pipeline for disaster recovery.	1. Define the scope and objectives of the disaster recovery test for the CDC pipeline, including the test scenarios and RTO to be achieved. 2. Identify the test environment and infrastructure for conducting the disaster recovery test. 3. Prepare the test datasets and script to simulate failure scenarios. 4. Verify data integrity and consistency to ensure that there is no data loss.	App owner, Data engineer, Cloud architect

Related resources

AWS resources

- Amazon DynamoDB
- Condition expressions with Amazon DynamoDB
- Amazon EKS
- Amazon EKS Anywhere
- Amazon ElastiCache
- Amazon Keyspaces
- Amazon MSK
- Amazon RDS and Amazon Aurora
- Amazon VPC

Precisely Connect resources

- Precisely Connect Overview
- Change Data Capture with Precisely Connect

Confluent resources

- Multi-Threaded Message Consumption with the Apache Kafka Consumer

Schedule jobs for Amazon RDS for PostgreSQL and Aurora PostgreSQL by using Lambda and Secrets Manager

Yaser Raja, Amazon Web Services

Summary

For on-premises databases and databases that are hosted on Amazon Elastic Compute Cloud (Amazon EC2) instances, database administrators often use the **cron** utility to schedule jobs.

For example, a job for data extraction or a job for data purging can easily be scheduled using **cron**. For these jobs, database credentials are typically either hard-coded or stored in a properties file. However, when you migrate to Amazon Relational Database Service (Amazon RDS) or Amazon Aurora PostgreSQL-Compatible Edition, you lose the ability to log in to the host instance to schedule **cron** jobs.

This pattern describes how to use AWS Lambda and AWS Secrets Manager to schedule jobs for Amazon RDS for PostgreSQL and Aurora PostgreSQL-Compatible databases after migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database

Limitations

- A job must complete within 15 minutes, which is the Lambda function timeout limit. For other limits, see the AWS Lambda documentation.
- Job code must be written in a language supported by Lambda.

Architecture

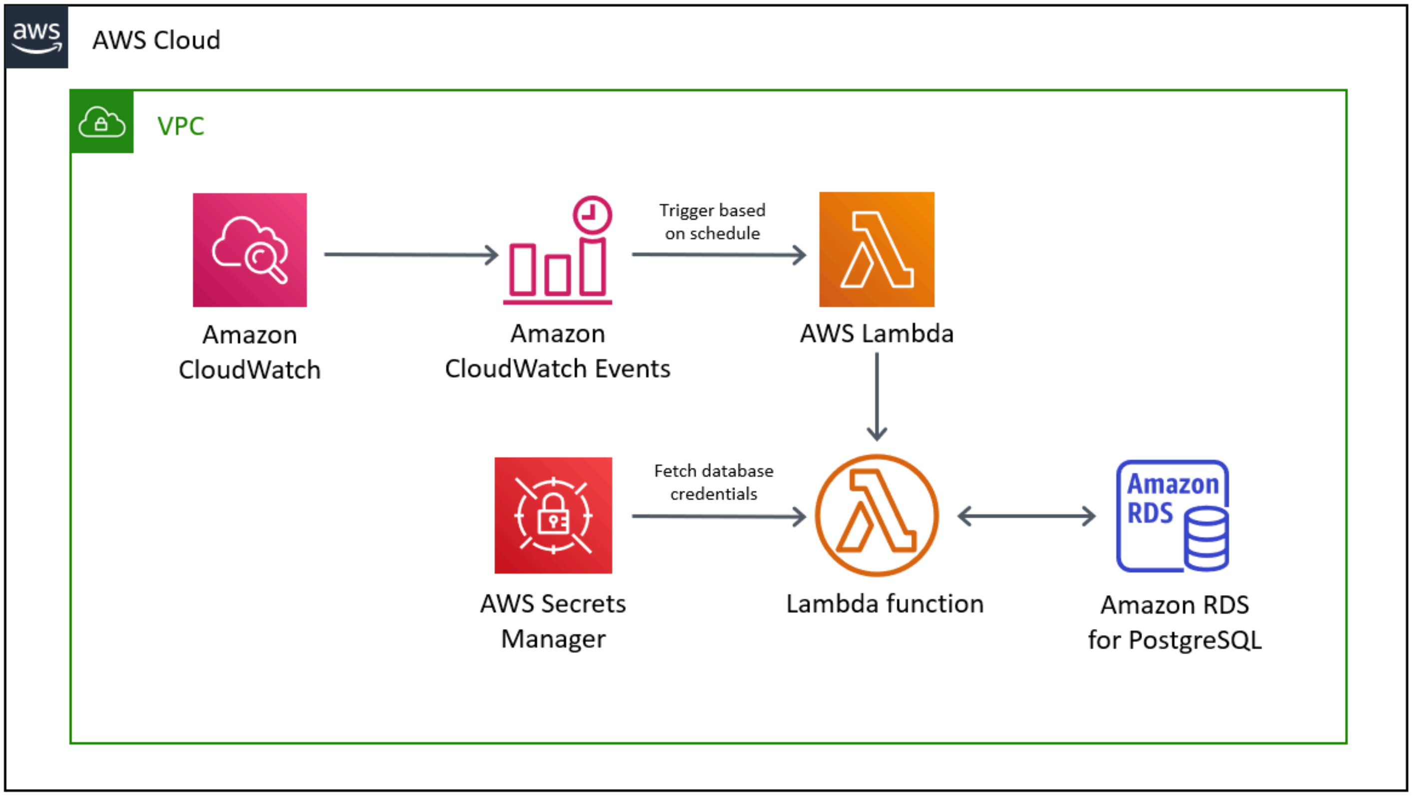
Source technology stack

This stack features jobs written in languages such as Bash, Python, and Java. Database credentials are stored in the properties file, and the job is scheduled using Linux **cron**.

Target technology stack

This stack has a Lambda function that uses the credentials stored in Secrets Manager to connect to the database and to perform the activity. The Lambda function is initiated at the scheduled interval by using Amazon CloudWatch Events.

Target architecture



Tools

- [Amazon CloudWatch Events](https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html) delivers a near real-time stream of system events that describe changes in AWS resources. Using simple rules that you can quickly set up, you can match events and route them to one or more target functions or streams. CloudWatch Events becomes aware of operational changes as they occur. It responds to these operational changes and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information. You can also use CloudWatch Events to schedule automated actions that self-initiate at certain times using **cron** or **rate** expressions.
- [AWS Lambda](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests

per day to thousands per second. You pay only for the compute time you consume; there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service with zero administration. Lambda runs your code on a high-availability compute infrastructure and manages all the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring, and logging. All you need to do is provide your code in one of the languages that Lambda supports.

- AWS Secrets Manager helps you protect secrets for accessing your applications, services, and IT resources. You can easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets by calling Secrets Manager APIs, which eliminates the need to hard-code sensitive information in plain text. Secrets Manager offers secret rotation with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB. The service is extensible to other types of secrets, including API keys and OAuth tokens. Secrets Manager enables you to control access to secrets using fine-grained permissions and to audit secret rotation centrally for resources in the AWS Cloud, third-party services, and on premises.

Epics

Store database credentials in Secrets Manager

Task	Description	Skills required
Create a database user for the Lambda function.	It is a good practice to use separate database users for different parts of your application. If a separate database user already exists for your cron jobs, use that. Otherwise, create a new database user. For more information, see Managing PostgreSQL users and roles (AWS blog post).	DBA
Store database credentials as a secret in Secrets Manager.	Follow the instructions in Create a database secret	DBA, DevOps

Task	Description	Skills required
	(Secrets Manager documentation).	

Author the code for the Lambda function

Task	Description	Skills required
Choose a programming language supported by Lambda.	For a list of supported languages, see [Lambda runtimes](#) (Lambda documentation).	Developer
Write the logic to fetch the database credentials from Secrets Manager.	For sample code, see [How to securely provide database credentials to Lambda functions by using AWS Secrets Manager](#) (AWS blog post).	Developer
Write the logic to perform the scheduled database activity.	Migrate your existing code for the scheduling job that you're using on premises to the Lambda function. For more information, see [Deploying Lambda functions](#) (Lambda documentation).	Developer

Deploy the code and create the Lambda function

Task	Description	Skills required
Create the Lambda function deployment package.	This package contains the code and its dependencies. For more information,	Developer

Task	Description	Skills required
	see Deployment packages (Lambda documentation).	
Create the Lambda function.	In the Lambda console, choose **Create function**, enter a function name, choose the runtime environment, and then choose **Create function**.	DevOps
Upload the deployment package.	Choose the Lambda function you created to open its configuration. You can write your code directly in the code section or upload your deployment package. To upload your package, go to the **Function code** section, choose the **Code entry type** to upload a .zip file, and then select the package.	DevOps
Configure the Lambda function per your requireme nts.	For example, you can set the **Timeout** parameter to the duration you expect your Lambda function to take. For more information, see Configuring function options (Lambda documentation).	DevOps
Set permissions for the Lambda function role to access Secrets Manager.	For instructions, see Use secrets in AWS Lambda functions (Secrets Manager documentation).	DevOps

Task	Description	Skills required
Test the Lambda function.	Initiate the Lambda function manually to make sure it works as expected.	DevOps

Schedule the Lambda function by using CloudWatch Events

Task	Description	Skills required
Create a rule to run your Lambda function on a schedule.	Schedule the Lambda function by using CloudWatc h Events. For instructions, see Schedule Lambda functions using CloudWatch Events (CloudWatch Events tutorial).	DevOps

Related resources

- AWS Secrets Manager
- Getting started with Lambda
- Creating a CloudWatch Events Rule That Triggers on an Event
- AWS Lambda Limits
- Query your AWS database from your serverless application (blog post)

Send notifications for an Amazon RDS for SQL Server database instance by using an on-premises SMTP server and Database Mail

Nishad Mankar, Amazon Web Services

Summary

Database Mail (Microsoft documentation) sends email messages, such as notifications or alerts, from a Microsoft SQL Server database by using a Simple Mail Transfer Protocol (SMTP) server. The Amazon Relational Database Service (Amazon RDS) for Microsoft SQL Server documentation provides instructions for using Amazon Simple Email Service (Amazon SES) as the SMTP server for Database Mail. For more information, see Using Database Mail on Amazon RDS for SQL Server. As an alternative configuration, this pattern explains how to configure Database Mail to send email from an Amazon RDS for SQL Server database (DB) instance by using an on-premises SMTP server as the mail server.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS DB instance running a Standard or Enterprise edition of SQL Server
- The IP address or hostname of the on-premises SMTP server
- An inbound security group rule that allows connections to the Amazon RDS for SQL Server DB instance from the IP address of the SMTP server
- A connection, such as an AWS Direct Connect connection, between your on-premises network and the virtual private cloud (VPC) that contains the Amazon RDS DB instance

Limitations

- Express editions of SQL Server aren't supported.
- For more information about limitations, see Limitations in *Using Database Mail on Amazon RDS for SQL Server* in the Amazon RDS documentation.

Product versions

- Standard and Enterprise editions of SQL Server versions supported in RDS

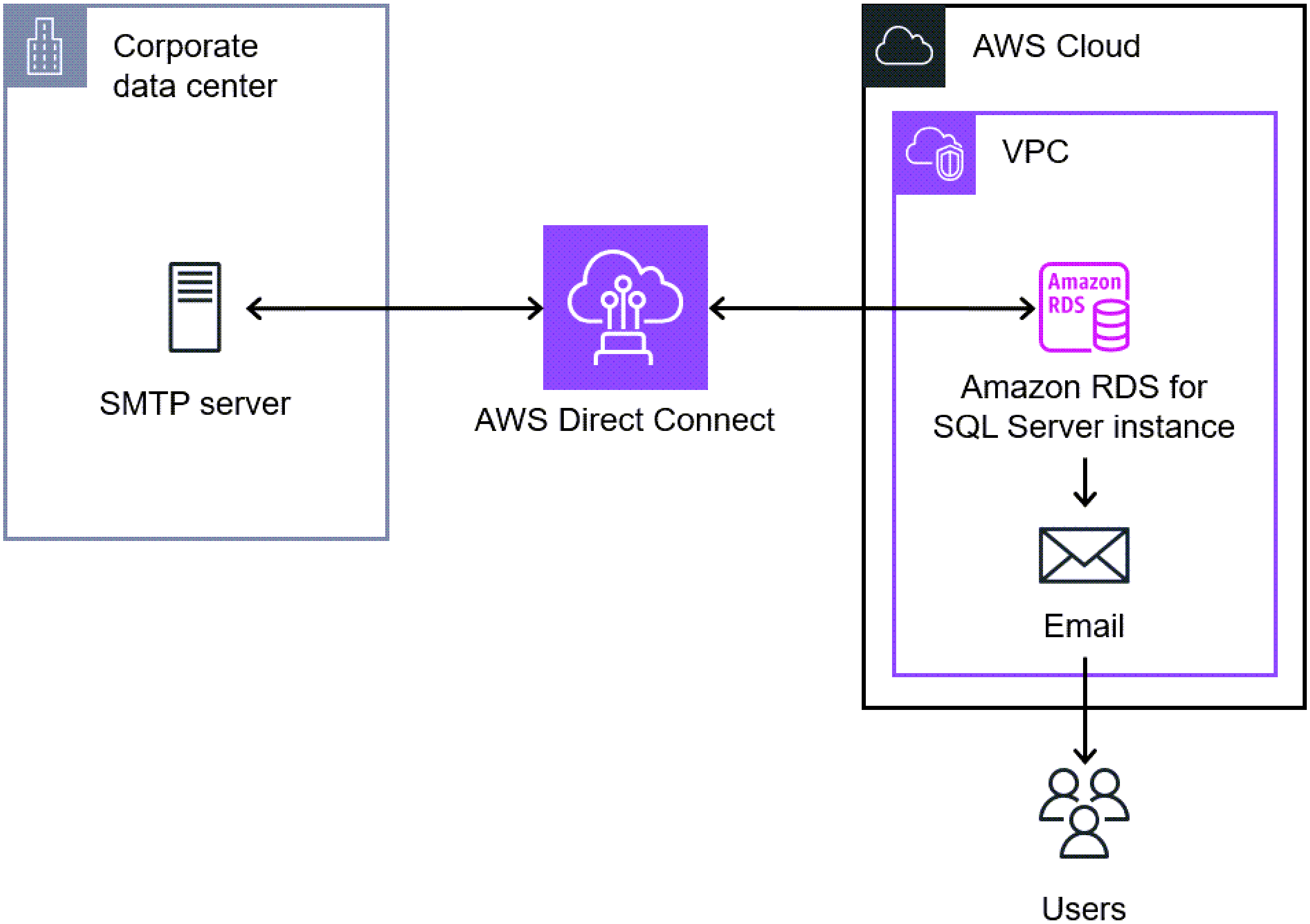
Architecture

Target technology stack

- Amazon RDS for SQL Server database instance
- Amazon Route 53 forwarding rule
- Database Mail
- On-premises SMTP server
- Microsoft SQL Server Management Studio (SSMS)

Target architecture

The following image shows the target architecture for this pattern. When an event or action occurs that initiates a notification or alert regarding the database instance, Amazon RDS for SQL Server uses Database Mail to send an email notification. Database Mail uses the on-premises SMTP server to send the email.



Tools

AWS services

- [Amazon Relational Database Service (Amazon RDS) for Microsoft SQL Server](#) helps you set up, operate, and scale a SQL Server relational database in the AWS Cloud.
- [Amazon Route 53](#) is a highly available and scalable DNS web service.

Other tools

- [Database Mail](#) is a tool that sends e-mail messages, such as notifications and alerts, from the SQL Server Database Engine to users.
- [Microsoft SQL Server Management Studio (SSMS)](#) is a tool for managing SQL Server, including accessing, configuring, and administering SQL Server components. In this pattern, you use

SSMS to run the SQL commands to set up Database Mail on an Amazon RDS for SQL Server DB instance.

Epics

Enable network connectivity with the on-premises SMTP server

Task	Description	Skills required
Remove Multi-AZ from the RDS DB instance.	If you're using a Multi-Zone RDS DB instance, convert the Multi-AZ instance to a Single-AZ instance. When you have finished configuring Database Mail, you will convert the DB instance back to a Multi-AZ deployment. The Database Mail configuration then works in both the primary and secondary nodes. For instructions, see Removing Multi-AZ from a Microsoft SQL Server DB instance.	DBA
Create an allow list for the Amazon RDS endpoint or IP address on the on-premises SMTP server.	The SMTP server is outside of the AWS network. On the on-premises SMTP server, create an allow list that permits the server to communicate with the outbound endpoint or IP address for the Amazon RDS instance or the Amazon Elastic Compute Cloud (Amazon EC2) instance hosted on Amazon RDS. This procedure varies from organization to organization.	DBA

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>For more information about the DB instance endpoint, see Finding the DB instance endpoint and port number.</td><td></td></tr>
<tr><td>Remove port 25 restrictions.</td><td>By default, AWS restricts port 25 on EC2 instances. To remove the port 25 restricti on, do the following:

1. Sign in with your AWS account, and then open the Request to remove email sending limitations form.
2. Enter your email address so that AWS Support can contact you with updates about your request.
3. Provide the required information in the Use case description field.
4. Choose Submit.

Note

• If you have instances in more than one AWS Region, then submit a separate request for each Region.
• It can take up to 48 hours to process your request.</td><td>General AWS</td></tr>
</table>

Task	Description	Skills required
Add a Route 53 rule to resolve DNS queries for the SMTP server.	Use Route 53 to resolve DNS queries between your AWS resources and the on-premises SMTP server. You must create a rule that forwards the DNS queries to the SMTP server domain, such as `example.com` . For instructions, see [Creating forwarding rules](#) in the Route 53 documentation.	Network administrator

Set up Database Mail on the Amazon RDS for SQL Server DB instance

Task	Description	Skills required
Enable Database Mail.	Create a parameter group for Database Mail, set the `database mail xps` parameter to 1, and then associate the Database Mail parameter group with the target RDS DB instance. For instructions, see [Enabling Database Mail](#) in the Amazon RDS documentation. Do not proceed to the *Configuring Database Mail* section in these instructions. The configuration for the on-premises SMTP server differs from Amazon SES.	DBA

Task	Description	Skills required
Connect to the DB instance.	From a bastion host, use Microsoft SQL Server Management Studio (SSMS) to connect to the Amazon RDS for SQL Server database instance. For instructions, see Connecting to a DB instance running the Microsoft SQL Server database engine. If you encounter any errors, see the connection troubleshooting references in the Related resources section.	DBA

Task	Description	Skills required
Create the profile.	In SSMS, enter the following SQL statement to create the Database Mail profile. Replace the following values: • For `profile_name` , enter a name for the new profile. • For `description` , enter a brief description of the new profile. For more information about this stored procedure and its arguments, see sysmail_add_profile_sp in the Microsoft documentation. `EXECUTE msdb.dbo. sysmail_add_profile_sp @profile_name = 'SQL Alerts profile', @description = 'Profile used for sending outgoing notifications using OM SMTP Server.';`	DBA

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add principals to the profile.</td><td>Enter the following SQL statement to add public or private principals to the Database Mail profile. A principal is an entity that can request SQL Server resources. Replace the following values:
For <code>profile_name</code> , enter the name of the profile you created previously.
For <code>principal_name</code> , enter the name of the database user or role. This value must map to an SQL Server authentication user, a Windows Authentic ation user, or a Windows Authentication group.
For more information about this stored procedure and its arguments, see sysmail_a dd_principalprofile_sp in the Microsoft documentation.<pre>EXECUTE msdb.dbo.
sysmail_add_princi
palprofile_sp
 @profile_name = 'SQL
 Alerts profile',
 @principal_name =
 'public',
 @is_default = 1 ;</pre></td><td>DBA</td></tr>
</table>

Task	Description	Skills required
Create the account.	Enter the following SQL statement to create the Database Mail account. Replace the following values: • For `account_name`, enter a name for the new account. • For `description`, enter a brief description of the new account. • For `email_address`, enter the e-mail address to send the Database Mail messages from. • For `display_address`, enter a display name to use for outgoing messages for this account, such as `SQL Server Automated Notification`. You can also use the value you entered for `email_address`. • For `mailserver_name`, enter the name or IP address of the SMTP mail server. • For `port`, leave the value of `25`. • For `enable_ssl`, leave the value at `1` or enter `0` if you don't want Database	DBA

Task	Description	Skills required
	Mail to encrypt communication by using SSL. • For `username`, enter the username for logging on to the SMTP mail server. If the server doesn't require authentication, enter `NULL`. • For `password`, enter the password for logging on to the SMTP mail server. If the server doesn't require authentication, enter `NULL`. For more information about this stored procedure and its arguments, see sysmail_add_account_sp in the Microsoft documentation. `EXECUTE msdb.dbo.sysmail_add_account_sp @account_name = 'SQL Alerts account', @description = 'Database Mail account for sending outgoing notifications.', @email_address = 'xyz@example.com', @display_name = 'xyz@example.com', @mailserver_name = 'test_smtp.example.com', @port = 25, @enable_ssl = 1,`	

Task	Description	Skills required
	` @username = 'SMTP-username',` ` @password = 'SMTP-password';`	

Add the account to the profile.	Enter the following SQL statement to add the Database Mail account to the Database Mail profile. Replace the following values:
	
	• For `profile_name` , enter the name of the profile you created previously.
	• For `account_name` , enter the name of the account you created previously.
	
	For more information about this stored procedure and its arguments, see [sysmail_add_profileaccount_sp](#) in the Microsoft documentation.
	
	`EXECUTE msdb.dbo.sysmail_add_profileaccount_sp`
	` @profile_name = 'SQL Alerts profile',`
	` @account_name = 'SQL Alerts account',`
	` @sequence_number = 1;`
		DBA

Task	Description	Skills required
(Optional) Add Multi-AZ to the RDS DB instance.	If you want to add Multi-AZ with Database Mirroring (DBM) or Always On Availability Groups (AGs), see the instructions in Adding Multi-AZ to a Microsoft SQL Server DB instance.	DBA

Related resources

- Using Database Mail on Amazon RDS for SQL Server (Amazon RDS documentation)
- Working with file attachments (Amazon RDS documentation)
- Troubleshooting connections to your SQL Server DB instance (Amazon RDS documentation)
- Can't connect to Amazon RDS DB instance (Amazon RDS documentation)

Set up disaster recovery for SAP on IBM Db2 on AWS

Ambarish Satarkar and Debasis Sahoo, Amazon Web Services

Summary

This pattern outlines the steps to set up a disaster recovery (DR) system for SAP workloads with IBM Db2 as the database platform, running on the Amazon Web Services (AWS) Cloud. The objective is to provide a low-cost solution for providing business continuity in the event of an outage.

The pattern uses the pilot light approach. By implementing pilot light DR on AWS, you can reduce downtime and maintain business continuity. The pilot light approach focuses on setting up a minimal DR environment in AWS, including an SAP system and a standby Db2 database, that is synchronized with the production environment.

This solution is scalable. You can extend it to a full-scale disaster recovery environment as needed.

Prerequisites and limitations

Prerequisites

- An SAP instance running on an Amazon Elastic Compute Cloud (Amazon EC2) instance
- An IBM Db2 database
- An operating system that is supported by the SAP Product Availability Matrix (PAM)
- Different physical database hostnames for production and standby database hosts
- An Amazon Simple Storage Service (Amazon S3) bucket in each AWS Region with Cross-Region Replication (CRR) enabled

Product versions

- IBM Db2 Database version 11.5.7 or later

Architecture

Target technology stack

- Amazon EC2

- Amazon Simple Storage Service (Amazon S3)
- Amazon Virtual Private Cloud (VPC peering)
- Amazon Route 53
- IBM Db2 High Availability Disaster Recovery (HADR)

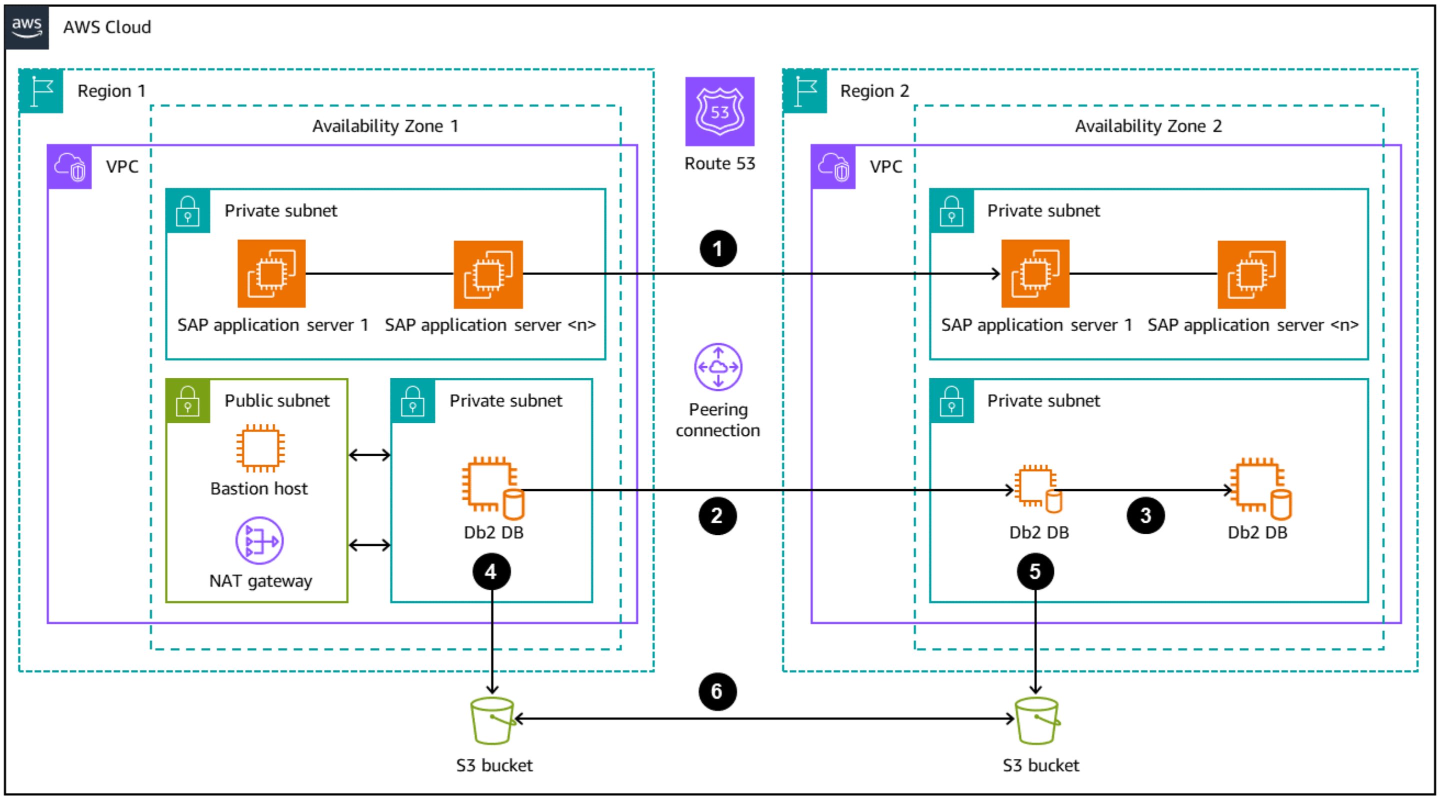
Target architecture

This architecture implements a DR solution for SAP workloads with Db2 as the database platform. The production database is deployed in AWS Region 1 and a standby database is deployed in a second Region. The standby database is referred to as the DR system. Db2 Database supports multiple standby databases (up to three). It uses Db2 HADR for setting up the DR database and automating log shipping between the production and standby databases.

In the event of a disaster that makes Region 1 unavailable, the standby database in the DR Region takes over the production database role. SAP application servers can be built in advance or by using AWS Elastic Disaster Recovery or an Amazon Machine Image (AMI) to meet the recovery time objective (RTO) requirements. This pattern uses an AMI.

Db2 HADR implements a production-standby setup, where production acts as the primary server, and all users are connected to it. All transactions are written to log files, which are transferred to the standby server by using TCP/IP. The standby server updates its local database by rolling forward the transferred log records, which helps to ensure that it is kept in sync with the production server.

VPC peering is used so that instances in the production Region and DR Region can communicate with each other. Amazon Route 53 routes end users to internet applications.



1. [Create an AMI](#) of the application server in Region 1 and [copy the AMI](#) to Region 2. Use the AMI to launch servers in Region 2 in the event of a disaster.
2. Set up Db2 HADR replication between the production database (in Region 1) and the standby database (in Region 2).
3. Change the EC2 instance type to match the production instance in the event of a disaster.
4. In Region 1, `LOGARCHMETH1` is set to `db2remote: S3 path`.
5. In Region 2, `LOGARCHMETH1` is set to `db2remote: S3 path`.
6. Cross-Region Replication is performed between the S3 buckets.

Tools

AWS services

- [Amazon Elastic Compute Cloud (Amazon EC2)](#) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- [Amazon Route 53](#) is a highly available and scalable DNS web service.

- [Amazon Simple Storage Service (Amazon S3)](#) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- [Amazon Virtual Private Cloud (Amazon VPC)](#) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. This pattern uses [VPC peering](#).

Best practices

- The network plays a key role in deciding the HADR replication mode. For DR across AWS Regions, we recommend that you use Db2 HADR ASYNC or SUPERASYNC mode.
- For more information about replication modes for Db2 HADR, see the [IBM documentation](#).
- You can use the AWS Management Console or the AWS Command Line Interface (AWS CLI) to [create a new AMI](#) of your existing SAP system. You can then use the AMI to recover your existing SAP system or to create a clone.
- [AWS Systems Manager Automation](#) can help with the common maintenance and deployment tasks of EC2 instances and other AWS resources.
- AWS provides multiple native services to monitor and manage your infrastructure and applications on AWS. Services such as Amazon CloudWatch and AWS CloudTrail can be used to monitor your underlying infrastructure and API operations, respectively. For more details, see [SAP on AWS – IBM Db2 HADR with Pacemaker](#).

Epics

Prepare the environment

Task	Description	Skills required
Check the system and logs.	1. Confirm that the productio n SAP on Db2 system is set up. 2. Confirm that log backup is turned on and configure d to save the logs in the S3 bucket. This can	AWS administrator, SAP Basis administrator

Task	Description	Skills required
	be checked by the Db2 parameter `LOGARCHMETH1` . 3. Create an AMI of the additional application server.	

Set up the servers and replication

Task	Description	Skills required
Create the SAP and database servers.	1. To deploy the infrastructure for the DR Region, use an AWS CloudFormation script or use an AMI of the production instance. As a part of the pilot light approach, you can use a smaller EC2 instance in the same family as the production instance. For example, if your production instance type is `r6i.12xlarge` , you can use the `r6i.xlarge` instance type for the DR build. However, make sure that you allocate the same storage capacity on the DR instance to restore the production database backup. 2. Create Amazon Elastic File System (Amazon	SAP Basis administrator

Task	Description	Skills required
	EFS) mount points for /`sapmnt/<SID>/`, and make sure that it is [set to be replicated](#) from the primary system. 3. Take a FULL database backup (online or offline) from the production system. You will use this backup to build the DR database. 4. In the DR system, use the SAP Software Provisioning Manager (SWPM) system copy method with **Using system copy with backup/restore for HA/DR purposes** to build the DR SAP system. 5. When asked by SWPM, restore the database in DR with the backup that you took from the production. The DR database will be in the rollforward pending state. The rollforward pending state is set by default after the full backup is restored. The rollforward pending state indicates that the database is in the process of being restored and that some	

Task	Description	Skills required
	changes might need to be applied. For more informati on, see the IBM documenta tion.	

Task	Description	Skills required

Check the configuration.	1. To set up log archiving for HADR, both the production and DR databases must be able to retrieve logs automatically from all log archive locations. Verify that the `LOGARCHMETH1` parameter in the DR database is set to the same location as in the production database. If the same location is not accessible because of Regional limitations, ensure that the DR system can automatically fetch logs from the primary system.
	2. To enable TCP/IP ports for database replication enablement, modify `/etc/services` in the production and DR hosts by adding the following two entries. In the code, `<SID>` refers to the System ID (SID) of the Db2 database (for example, PR1).
	`<SID>_HADR_1 55001/tcp # DB2 HADR Port1`
	`<SID>_HADR_2 55002/tcp # DB2 HADR Port2`
		AWS administrator, SAP Basis administrator

Task	Description	Skills required
	Confirm that both ports allow inbound and outbound traffic between both the primary and the standby. 3. Check `/etc/hosts` in the production and DR hosts to confirm that hostnames for both production and standby hosts are pointing to the correct IP addresses.	

Task	Description	Skills required

Set up replication from the production DB to the DR DB (using ASYNC mode).	1. In the production database, run the following commands to update the parameters.
	`db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_HOST HOST1`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_SVC <SID>_HADR_1`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_HOST HOST2`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_SVC <SID>_HADR_2`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_INST db2<sid>`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_TIMEOUT 120`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_SYNCMODE ASYNC`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_SPOOL_LIMIT 1000`
	`db2 UPDATE DB CFG FOR <SID> USING HADR_PEER_WINDOW 240`
	`db2 UPDATE DB CFG FOR <SID> USING indexrec RESTART logindexbuild ON`
	2. In the DR database, run the following commands to update the parameters.
		SAP Basis administrator

Task	Description	Skills required

	<pre>db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_HOST HOST2
	db2 UPDATE DB CFG FOR <SID> USING HADR_LOCAL_SVC <SID>_HADR_2
	db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_HOST HOST1
	db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_SVC <SID>_HADR_1
	db2 UPDATE DB CFG FOR <SID> USING HADR_REMOTE_INST db2<sid>
	db2 UPDATE DB CFG FOR <SID> USING HADR_TIMEOUT 120
	db2 UPDATE DB CFG FOR <SID> USING HADR_SYNCMODE ASYNC
	db2 UPDATE DB CFG FOR <SID> USING HADR_SPOOL_LIMIT 1000
	db2 UPDATE DB CFG FOR <SID> USING HADR_PEER_WINDOW 240
	db2 UPDATE DB CFG FOR <SID> USING indexrec RESTART logindexbuild ON</pre>
	These parameters are required to provide HADR-related information to both databases. In the Db2 database, HADR gets activated based on the values for each of the	

Task	Description	Skills required

	previously set parameters. For more information about these parameters, see the [IBM documentation]().
	3. Start HADR first on the newly created standby database by using the following command.
	`db2 deactivate db <SID>`
	`db2 start hadr on db <SID> as standby`
	4. Start HADR on the production database by using the following command.
	`db2 deactivate db <SID>`
	`db2 start hadr on db <SID> as primary`
	5. Check whether the production and standby Db2 databases are in sync and log shipping is ongoing.
	To monitor HADR replication status, use the following `db2pd` command.
	`db2pd -d <SID> -hadr`	

Task	Description	Skills required
	For more information about monitoring HADR, see the [IBM documentation]().	

Test DR failover tasks

Task	Description	Skills required
Plan the production business downtime for the DR test.	Make sure that you plan the required business downtime on production environment for testing the DR failover scenario.	SAP Basis administrator
Create a test user.	Create a test user (or any test changes) that can be validated in the DR host to confirm log replication after DR failover.	SAP Basis administrator
On the console, stop the production EC2 instances.	Ungraceful shutdown is initiated in this step to mimic a disaster scenario.	AWS systems administrator
Scale up the DR EC2 instance to match the requirements.	On the EC2 console, change the instance type in the DR Region. 1. Stop the instance: If the instance is running, you must stop it before you can change its instance type. On the EC2 console, select	SAP Basis Admin

Task	Description	Skills required

	the instance, and choose **Stop**.
	2. Modify the instance type: On the EC2 console, select the instance, and choose **Actions**, **Instance Settings**, **Change Instance Type**. Select the instance type that matches the primary instance, and choose **Apply**.
	3. Start the instance: After the instance type change is complete, start the instance from the EC2 console by selecting the instance and choosing **Start**.
	4. To start the Db2 database, use the following command.
	`db2start`
	`    db2 start HADR on db <SID> as standby`	

Task	Description	Skills required

Initiate takeover.	From the DR system (`host2`), initiate the take-over process and bring up the DR database as the primary.
	`db2 takeover hadr on database <SID> by force`
	Optionally, you can set the following parameters to adjust database memory allocation automatically based on the instance type. The `INSTANCE_MEMORY` value can be decided based on the dedicated portion of memory to be allocated to the Db2 database.
	`db2 update db cfg for <SID> using INSTANCE_MEMORY <FIXED VALUE> IMMEDIATE;`
	`db2 get db cfg for <SID> | grep -i DATABASE_MEMORY AUTOMATIC IMMEDIATE;`
	`db2 update db cfg for <SID> using self_tuning_mem ON IMMEDIATE;`
	Verify the change by using the following commands.
	`db2 get db cfg for <SID> | grep -i MEMORY`
		SAP Basis administrator

Task	Description	Skills required
	`db2 get db cfg for <SID> | grep -i self_tuning_mem`	
Launch the application server for SAP in the DR Region.	Using the AMI that you made of the production system, [launch a new additional application server](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/launching-instance.html) in the DR Region.	SAP Basis administrator
Perform validation before starting the SAP application.	1. Validate the `/etc/hosts` and `/etc/fstab` entries. 2. Mount `/sapmnt/<SID>/` on the DR system. 3. Validate that the DR file system `/sapmnt/<SID>/` is synced with the production `/sapmnt/<SID>/`. 4. Log in to `<sid>adm` user, run `R3trans -d`, and verify the output in the `trans.log` file. The `trans.log` file is generated in the same location where you ran the `R3trans -d` command.	AWS administrator, SAP Basis administrator

Task	Description	Skills required

Start the SAP application on the DR system.	Start the SAP application on the DR system by using `<sid>adm` user. Use the following code, in which XX represents the instance number of your SAP ABAP SAP Central Services (ASCS) server, and YY represents the instance number of your SAP application server.
	`sapconrol -nr XX -function StartService <SID>`
	`sapconrol -nr XX -function StartSystem`
	`sapconrol -nr YY -function StartService <SID>`
	`sapconrol -nr YY -function StartSystem`
		SAP Basis administrator

Perform SAP validation.	This is performed as a DR test to provide evidence or to check the data replication success to the DR Region.	Test engineer

Perform DR failback tasks

Task	Description	Skills required
Start the production SAP and database servers.	On the console, start the EC2 instances that host SAP and the database in the productio n system.	SAP Basis administrator

Task	Description	Skills required

Start the production database and set up HADR.	Log in to production system (`host1`) and verify that the DB is in recovery mode by using the following command.
	`db2start`
	`db2 start HADR on db P3V as standby`
	`db2 connect to <SID>`
	Verify that the HADR status is `connected` . Replication status should be `peer`.
	`db2pd -d <SID> -hadr`
	If the database is not inconsistent and is not at `connected` and `peer` status, a backup and restore might be required to bring the database (on `host1`) in sync with the currently active database (`host2` in the DR Region). In that case, restore the DB backup from the database in the `host2` DR Region to the database in the `host1` production Region.
		SAP Basis administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Fail back the database to the production Region.</td><td>In a normal business-as-usual scenario, this step is performed in a scheduled downtime. Applications running on the DR system are stopped, and the database is failed back to the production Region (Region 1) to resume operations from the production Region.

1. Log in to the SAP application server in the DR Region, and stop the SAP application.
2. Unmount <code>/sapmnt/<SID></code> from the DR system, making sure that the changes are reverse-replicated to <code>/sapmnt/<SID></code> of the production system.
3. Log in to the database server (<code>host1</code>) in the production Region, and perform the takeover.
<pre>db2 takeover hadr on
 database <SID></pre>4. Check the HADR status: <code>HADR_ROLE</code> should be <code>PRIMARY</code> on <code>host1</code> and <code>StandBy</code> on <code>host2</code>.</td><td>SAP Basis administrator</td></tr>
</table>

Task	Description	Skills required
	`db2pd -d <SID> -hadr`	
Perform validation before starting the SAP application.	1. Validate the `/etc/hosts` and `/etc/fstab` entries. 2. Mount `/sapmnt/<SID>/` on the production system. 3. Make sure it is in sync with the DR system `/sapmnt/<SID>/` . 4. Log in to `<sid>adm` user, run `R3trans -d`, and verify the output in the `trans.log` file. The `trans.log` file is generated in the same location where you ran the `R3trans -d` command.	AWS administrator, SAP Basis administrator

Task	Description	Skills required

Start the SAP application.	1. Start the SAP application on the production system using `<sid>adm` user. Use the following code, in which XX represents the instance number of your SAP ASCS server, and YY represents the instance number of your SAP application server.
	`sapconrol -nr XX -function StartService <SID>`
	`sapconrol -nr XX -function StartSystem`
	`sapconrol -nr YY -function StartService <SID>`
	`sapconrol -nr YY -function StartSystem`
	2. To confirm that applicati on servers are available, log in to SAP and perform checks by using the SICK and SM51 transactions.
		SAP Basis administrator

Troubleshooting

Issue	Solution
Key log files and commands to troubleshoot HADR-related issues	• `db2 get db cfg | grep -i hadr` • `db2pd -d sid -hadr`

Issue	Solution
	• `Db2diag.log` (This file is generally located inside the `db2dump` directory, and the `db2dump` path is defined by the parameter `DIAGPATH`.)
SAP note for troubleshooting HADR issues on Db2 UDB	Refer to SAP Note 1154013 - DB6: DB problems in HADR environment. (You need SAP portal credentials to access this note.)

Related resources

- Disaster recovery approaches for Db2 databases on AWS (blog post)
- SAP on AWS – IBM Db2 HADR with Pacemaker
- Step by Step Procedure to set up HADR replication between DB2 databases
- Db2 HADR Wiki

Additional information

Using this pattern, you can set up a disaster recovery system for an SAP system running on the Db2 database. In a disaster situation, business should be able to continue within your defined recovery time objective (RTO) and recovery point objective (RPO) requirements:

- **RTO** is the maximum acceptable delay between the interruption of service and restoration of service. This determines what is considered an acceptable time window when service is unavailable.
- **RPO** is the maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

For FAQs related to HADR, see SAP note #1612105 - DB6: FAQ on Db2 High Availability Disaster Recovery (HADR). (You need SAP portal credentials to access this note.)

Set up a CI/CD pipeline for database migration by using Terraform

Dr. Rahul Sharad Gaikwad, Ashish Bhatt, Aniket Dekate, Ruchika Modi, Tamilselvan P, Nadeem Rahaman, Aarti Rajput, and Naveen Suthar, Amazon Web Services

Summary

This pattern is about establishing a continuous integration and continuous deployment (CI/CD) pipeline for managing database migrations in a reliable and automated manner. It covers the process of provisioning the necessary infrastructure, migrating data, and customizing schema changes by using Terraform, which is an infrastructure as code (IaC) tool.

Specifically, the pattern sets up a CI/CD pipeline to migrate an on-premises Microsoft SQL Server database to Amazon Relational Database Service (Amazon RDS) on AWS. You can also use this pattern to migrate a SQL Server database that's on a virtual machine (VM) or in another cloud environment to Amazon RDS.

This pattern addresses the following challenges associated with database management and deployment:

- Manual database deployments are time-consuming, error-prone, and lack consistency across environments.
- Coordinating infrastructure provisioning, data migrations, and schema changes can be complex and difficult to manage.
- Ensuring data integrity and minimizing downtime during database updates is crucial for production systems.

This pattern provides the following benefits:

- Streamlines the process of updating and deploying database changes by implementing a CI/CD pipeline for database migrations. This reduces the risk of errors, ensures consistency across environments, and minimizes downtime.
- Helps improve reliability, efficiency, and collaboration. Enables faster time to market and reduced downtime during database updates.
- Helps you adopt modern DevOps practices for database management, which leads to increased agility, reliability, and efficiency in your software delivery processes.

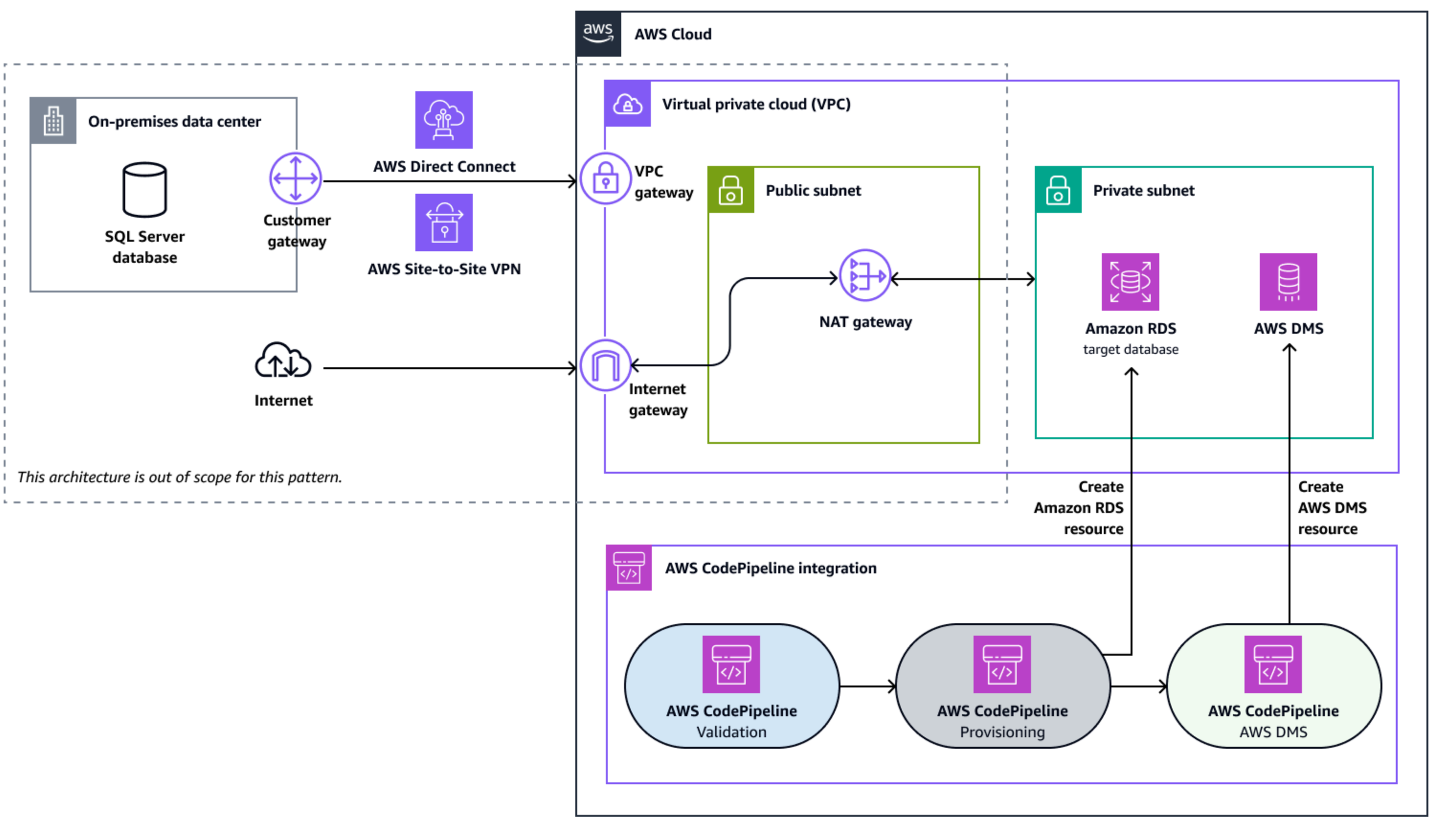
Prerequisites and limitations

Prerequisites

- An active AWS account
- Terraform 0.12 or later installed on your local machine (for instructions, see the Terraform documentation)
- Terraform AWS Provider version 3.0.0 or later from HashiCorp (see the GitHub repository for this provider)
- Least privilege AWS Identity and Access Management (IAM) policy (see the blog post Techniques for writing least privilege IAM policies)

Architecture

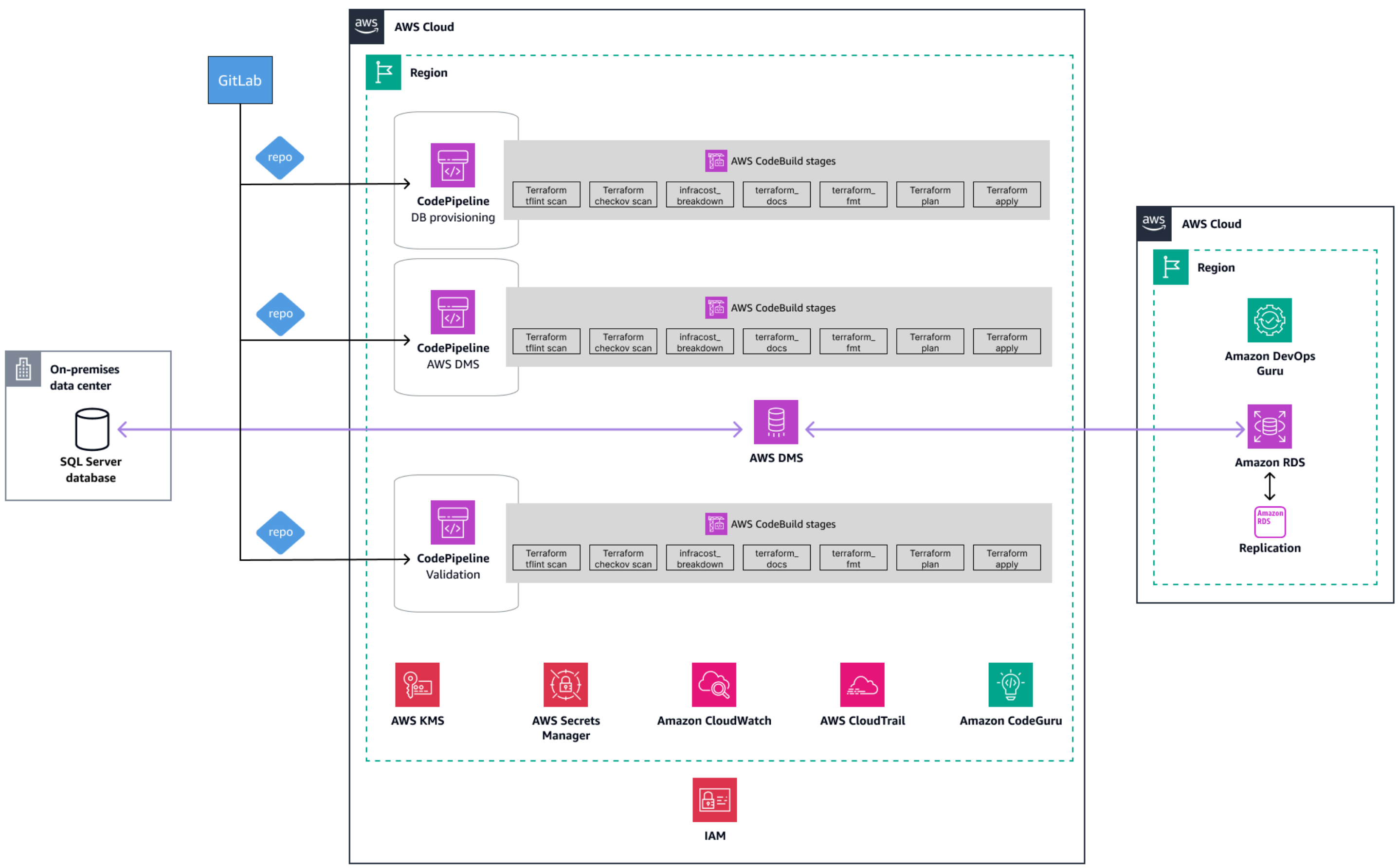
This pattern implements the following architecture, which provides the complete infrastructure for the database migration process.



In this architecture:

- The source database is a SQL Server database that is on premises, on a virtual machine (VM), or hosted by another cloud provider. The diagram assumes that the source database is in an on-premises data center.
- The on-premises data center and AWS are connected through a VPN or AWS Direct Connect connection. This provides secure communications between the source database and the AWS infrastructure.
- The target database is an Amazon RDS database that is hosted inside the virtual private cloud (VPC) on AWS with the help of a database provisioning pipeline.
- AWS Database Migration Service (AWS DMS) replicates your on-premises database to AWS. It is used to configure the replication of the source database to the target database.

The following diagram shows the infrastructure set up with different levels of the database migration process, which involves provisioning, AWS DMS setup, and validation.



In this process:

- The validation pipeline validates all checks. The integrated pipeline moves to the next step when all necessary validations are complete.
- The DB provisioning pipeline consists of various AWS CodeBuild stages that perform Terraform actions on the provided Terraform code for the database. When these steps are complete, it deploys resources in the target AWS account.
- The AWS DMS pipeline consists of various CodeBuild stages that perform tests and then provision the AWS DMS infrastructure for performing the migration by using IaC.

Tools

AWS services and tools

- AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces ready-to-deploy software packages.
- AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance.
- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.

Other services

- Terraform is an IaC tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub Database Migration DevOps Framework using Terraform samples repository.

Best practices

- Implement automated tests for your database migration to verify the correctness of schema changes and data integrity. This includes unit tests, integration tests, and end-to-end tests.
- Implement a robust backup and restore strategy for your databases, especially before migration. This ensures data integrity and provides a fallback option in case of failures.
- Implement a robust rollback strategy to revert database changes in case of failures or issues during migration. This could involve rolling back to a previous database state or reverting individual migration scripts.
- Set up monitoring and logging mechanisms to track the progress and status of database migrations. This helps you identify and resolve issues quickly.

Epics

Set up your local workstation

Task	Description	Skills required
Set up and configure Git on your local workstation.	Install and configure Git on your local workstation by following the instructions in the [Git documentation]().	DevOps engineer
Create a project folder and add the files from the GitHub repository.	1. Open the [GitHub repository]() for this pattern. 2. Choose **Code** to see cloning options, and copy the URL provided in the HTTPS tab. 3. Create a folder for your project on your workstation. 4. Open a terminal and navigate to this folder. 5. Clone the GitHub repository:	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>
<pre><code>git clone <github-r
epository-url></code></pre>
where <code><github-r epository-url></code> is the URL you copied in step 2.
<ol start="6">
When cloning is complete, go to the cloned repository in your project folder:
<pre><code>cd <folder-name>/aws-
terraform-db-migra
tion-framework-sam
ples</code></pre>
Open this project in an integrated development environment (IDE) of your choice.

</td><td></td></tr>
</table>

Provision the target architecture

Task	Description	Skills required
Update required parameters.	The `ssm-parameters.sh` file stores all required AWS Systems Manager parameter s. You can configure these parameters with the custom values for your project. In the `setup/db-ssm-params` folder on your local workstation, open the `ssm-`	DevOps engineer

Task	Description	Skills required
	`parameters.sh` file and set these parameters before you run the CI/CD pipeline.	
Initialize the Terraform configuration.	In the `db-cicd-integration` folder, enter the following command to initialize your working directory that contains the Terraform configuration files: `terraform init`	DevOps engineer
Preview the Terraform plan.	To create a Terraform plan, enter the following command: `terraform plan -var-file="terraform.sample"` Terraform evaluates the configuration files to determine the target state for the declared resources. It then compares the target state against the current state and creates a plan.	DevOps engineer
Verify the plan.	Review the plan and confirm that it configures the required architecture in your target AWS account.	DevOps engineer

Task	Description	Skills required
Deploy the solution.	1. Enter the following command to apply the plan: `terraform apply -var-file="terraform.sample"` 2. Enter `yes` to confirm. Terraform creates, updates, or destroys infrastructure to achieve the target state declared in the configuration files. For more information about the sequence, see the [Architecture](#) section of this pattern.	DevOps engineer

Verify the deployment

Task	Description	Skills required
Validate the deployment.	Verify the status of the `db-cicd-integration` pipeline to confirm that the database migration is complete. 1. Sign in to the AWS Management Console, and then open the [AWS CodePipeline console](#).	DevOps engineer

Task	Description	Skills required
	2. In the navigation pane, choose **Pipelines**. 3. Choose the `db-cicd-integration` pipeline. 4. Validate that the pipeline execution has completed successfully.	

Clean up infrastructure after use

Task	Description	Skills required
Clean up the infrastructure.	1. After your project is complete, clean up the infrastructure you created by using the command: `terraform destroy --var-file=terraform.sample` 2. Enter `yes` to confirm.	DevOps engineer

Related resources

AWS documentation

- Getting started with a Terraform product

Terraform documentation

- Terraform installation
- Terraform backend configuration

- Terraform AWS Provider documentation

Set up an HA/DR architecture for Oracle E-Business Suite on Amazon RDS Custom with an active standby database

Simon Cunningham, Amazon Web Services

Nitin Saxena, None

Summary

This pattern describes how you can architect your Oracle E-Business solution on Amazon Relational Database Service (Amazon RDS) Custom for high availability (HA) and disaster recovery (DR) by setting up an Amazon RDS Custom read replica database in another Amazon Web Services (AWS) Availability Zone and converting it to an active standby database. The creation of the Amazon RDS Custom read replica is fully automated through the AWS Management Console.

This pattern doesn't discuss the steps for adding additional application tiers and shared file systems, which can also be part of an HA/DR architecture. For more information about those topics, see the following Oracle Support Notes: 1375769.1, 1375670.1, and 1383621.1 (section 5, *Advanced Cloning Options*). (Access requires an Oracle Support account.)

To migrate E-Business Suite system to a single-tier, Single-AZ architecture on Amazon Web Services (AWS), see the pattern Migrate Oracle E-Business Suite to Amazon RDS Custom.

Oracle E-Business Suite is an Enterprise Resource Planning (ERP) solution for automating enterprise-wide processes such as financials, human resources, supply chains, and manufacturing. It has a three-tier architecture: client, application, and database. Previously, you had to run your E-Business Suite database on a self-managed Amazon Elastic Compute Cloud (Amazon EC2) instance, but you can now benefit from Amazon RDS Custom.

Prerequisites and limitations

Prerequisites

- An existing E-Business Suite installation on Amazon RDS Custom; see the pattern Migrate Oracle E-Business Suite to Amazon RDS Custom
- If you want to change the read replica to read-only and use it to offload reporting to the standby, an Oracle Active Data Guard database license (see the *Oracle Technology Commercial Price List*)

Limitations

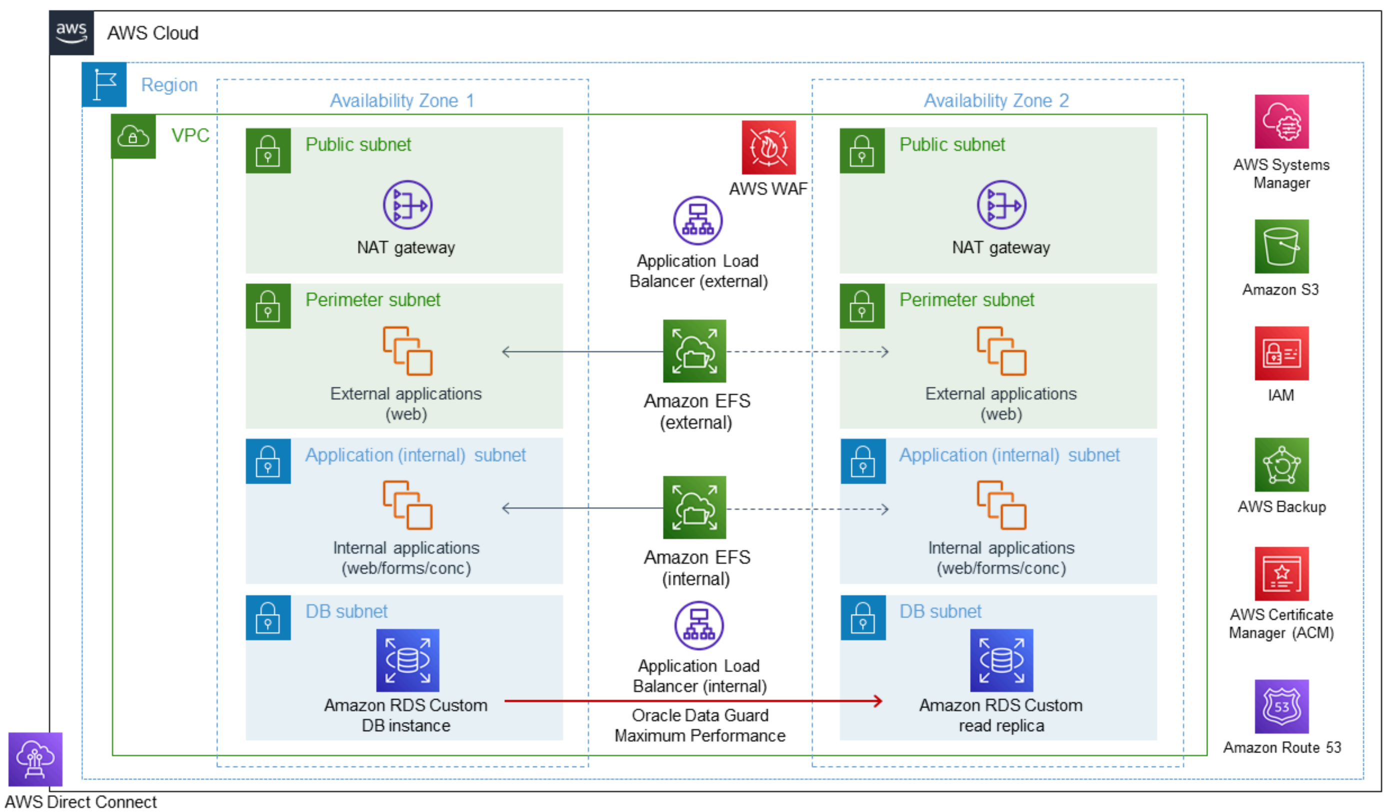
- Limitations and unsupported configurations for Oracle databases on Amazon RDS Custom
- Limitations associated with Amazon RDS Custom for Oracle read replicas

Product versions

For Oracle Database versions and instance classes supported by Amazon RDS Custom, see Requirements and limitations for Amazon RDS Custom for Oracle.

Architecture

The following diagram illustrates a representative architecture for E-Business Suite on AWS that includes multiple Availability Zones and application tiers in an active/passive setup. The database uses an Amazon RDS Custom DB instance and Amazon RDS Custom read replica. The read replica uses Active Data Guard to replicate to another Availability Zone. You can also use the read replica to offload read traffic on the primary database and for reporting purposes.



For more information, see Working with read replicas for Amazon RDS Custom for Oracle in the Amazon RDS documentation.

The Amazon RDS Custom read replica is created by default as mounted. However, if you want to offload some of your read-only workloads to the standby database to reduce the load on your primary database, you can manually change the mode of mounted replicas to read-only by following the steps in the Epics section. A typical use case for this would be to run your reports from the standby database. Changing to read-only requires an active standby database license.

When you create a read replica on AWS, the system uses Oracle Data Guard broker under the covers. This configuration is automatically generated and set up in Maximum Performance mode as follows:

```
DGMGRL> show configuration
Configuration - rds_dg
  Protection Mode: MaxPerformance
  Members:
  vis_a - Primary database
    vis_b - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS   (status updated 58 seconds ago)
```

Tools

AWS services

- Amazon RDS Custom for Oracle is a managed database service for legacy, custom, and packaged applications that require access to the underlying operating system and database environment. It automates database administration tasks and operations while making it possible for you, as a database administrator, to access and customize your database environment and operating system.

Other tools

- Oracle Data Guard is a tool that helps you create and manage Oracle standby databases. This pattern uses Oracle Data Guard to set up an active standby database on Amazon RDS Custom.

Epics

Create a read replica

Task	Description	Skills required
Create a read replica of the Amazon RDS Custom DB instance.	To create a read replica, follow the instructions in the [Amazon RDS documentation](#) and use the Amazon RDS Custom DB instance you created (see the [Prerequisites](#) section) as the source database. By default, the Amazon RDS Custom read replica is created as a physical standby and is in the mounted state. This is intentional to ensure compliance with the Oracle Active Data Guard license. Follow the next steps to convert the read replica to read-only mode.	DBA

Change the read replica to a read-only active standby

Task	Description	Skills required
Connect to the Amazon RDS Custom read replica.	Use the following commands to convert your physical standby database to an active standby database.	DBA

Task	Description	Skills required

	Important These commands require an Oracle active standby license. To get a license, contact your Oracle representative.
	
	`$ sudo su - rdsdb`
	`-bash-4.2$ sql`
	`SQL> select process,status,sequence# from v$managed_standby;`
	
	`PROCESS   STATUS   SEQUENCE#`
	`--------- ------------ ----------`
	`ARCH      CLOSING      3956`
	`ARCH      CONNECTED    0`
	`ARCH      CLOSING      3955`
	`ARCH      CLOSING      3957`
	`RFS       IDLE         0`
	`RFS       IDLE         3958`
	`MRP0      APPLYING_LOG 3958`
	`SQL> select name, database_role, open_mode from v$database;`	

Task	Description	Skills required

	<pre>NAME DATABASE_
	ROLE OPEN_MODE
	--------- ---------
	------- ---------
	----------- 	
	VIS PHYSICAL
	 STANDBY MOUNTED
	SQL> alter database
	 recover managed standby
	 database cancel;
	Database altered.
	Open the standby
	 database
	SQL> alter database
	 open;
	Database altered.
	SQL> select name,
	 database_role,
	 open_mode from v
	$database;
	
	NAME DATABASE_
	ROLE OPEN_MODE
	--------- ---------
	------- ---------
	----------- 	
	VIS PHYSICAL
	 STANDBY READ ONLY</pre>	

Task	Description	Skills required
Start media recovery with real-time log apply.	To enable the real-time log apply feature, use the following commands. These convert and validate the standby (read replica) as an active standby database, so you can connect and run read-only queries. ``` SQL> alter database recover managed standby database using current logfile disconnect from session; Database altered ```	DBA
Check the database status.	To check the status of the database, use the following command. ``` SQL> select name, database_role, open_mode from v $database; NAME DATABASE_ROLE OPEN_MODE --------- --------- ------- --------- ----------- VIS PHYSICAL STANDBY READ ONLY WITH APPLY ```	DBA

Task	Description	Skills required

Check redo apply mode.	To check redo apply mode, use the following command.
	
	`SQL> select process,status,sequence# from v$managed_standby;`
	`PROCESS STATUS SEQUENCE#`
	`--------- ------------ ----------`
	`ARCH CLOSING 3956`
	`ARCH CONNECTED 0`
	`ARCH CLOSING 3955`
	`ARCH CLOSING 3957`
	`RFS IDLE 0`
	`RFS IDLE 3958`
	`MRP0 APPLYING_LOG 3958`
	
	`SQL> select open_mode from v$database;`
	`OPEN_MODE`
	`--------------------`
	`READ ONLY WITH APPLY`
		DBA

Related resources

- Migrate Oracle E-Business Suite to Amazon RDS Custom (AWS Prescriptive Guidance)
- Working with Amazon RDS Custom (Amazon RDS documentation)
- Working with read replicas for Amazon RDS Custom for Oracle (Amazon RDS documentation)

- Amazon RDS Custom for Oracle – New Control Capabilities in Database Environment (AWS News blog)
- Migrating Oracle E-Business Suite on AWS (AWS whitepaper)
- Oracle E-Business Suite architecture on AWS (AWS whitepaper)

Set up data replication between Amazon RDS for MySQL and MySQL on Amazon EC2 using GTID

Rajesh Madiwale, Amazon Web Services

Summary

This pattern describes how to set up data replication on the Amazon Web Services (AWS) Cloud between an Amazon Relational Database Service (Amazon RDS) for MySQL DB instance and a MySQL database on an Amazon Elastic Compute Cloud (Amazon EC2) instance by using MySQL native global transaction identifier (GTID) replication.

With GTIDs, transactions are identified and tracked when they are committed on the originating server and applied by replicas. You don't need to refer to log files when starting a new replica during failover.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Linux instance deployed

Restrictions

- This setup requires an internal team to run the read-only queries.
- The source and target MySQL versions must be the same.
- Replication is set up in the same AWS Region and virtual private cloud (VPC).

Product versions

- Amazon RDS versions 5.7.23 and later, which are the versions that support GTID

Architecture

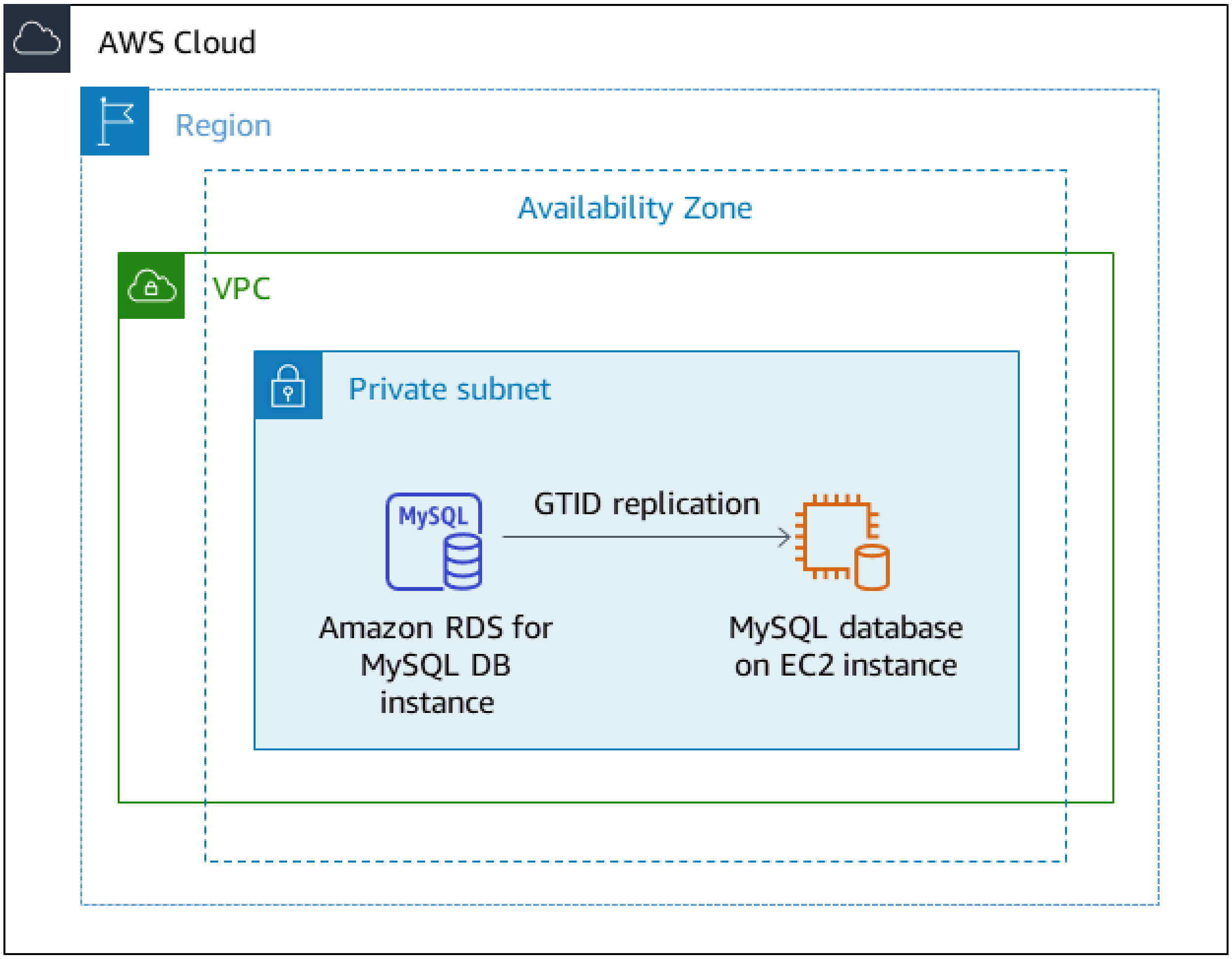
Source technology stack

- Amazon RDS for MySQL

Target technology stack

- MySQL on Amazon EC2

Target architecture



Tools

AWS services

- [Amazon Elastic Compute Cloud (Amazon EC2)](#) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- [Amazon Relational Database Service (Amazon RDS) for MySQL](#) helps you set up, operate, and scale a MySQL relational database in the AWS Cloud.

Other services

- Global transaction identifiers (GTIDs) are unique identifiers generated for committed MySQL transactions.
- mysqldump is a client utility for performing logical backups by producing SQL statements that can be run to reproduce the source database object definitions and table data.
- mysql is the command-line client for MySQL.

Epics

Create and prepare the Amazon RDS for MySQL DB instance

Task	Description	Skills required
Create the RDS for MySQL instance.	To create the RDS for MySQL instance, follow the steps in the Amazon RDS documentation, using the parameter values that are covered in the next task.	DBA, DevOps engineer
Enable GTID-related settings in the DB parameter group.	Enable the following parameters in the Amazon RDS for MySQL DB parameter group. Set `enforce_gtid_consistency` to on, and set `gtid-mode` to on.	DBA
Reboot the Amazon RDS for MySQL instance.	A reboot is required for the parameter changes to take effect.	DBA
Create a user and grant it replication permissions.	To install MySQL, use the following commands.	DBA

Task	Description	Skills required
	``` CREATE USER 'repl'@'%' IDENTIFIED BY 'xxxx'; GRANT REPLICATI ON slave ON *.* TO 'repl'@'%' ; FLUSH PRIVILEGES; ```	

Install and prepare MySQL on the Amazon EC2 instance

Task	Description	Skills required
Install MySQL on Amazon Linux.	To install MySQL, use the following commands. ``` sudo yum update sudo wget https://d ev.mysql.com/get/m ysql57-community-r elease-el7-11.noar ch.rpm sudo yum localinstall mysql57-community- release-el7-11.noa rch.rpm sudo yum install mysql- community-server sudo systemctl start mysqld ```	DBA
Log in to MySQL on the EC2 instance and create the database.	The database name should be the same as the database name in Amazon RDS for MySQL. In the following example, the database name is `replication` .	DBA

Task	Description	Skills required
	`create database replication;`	

Edit the MySQL config file, and restart the database.	Edit the `my.conf` file that is located in `/etc/` by adding the following parameters.
	`server-id=3`
	`gtid_mode=ON`
	`enforce_gtid_consistency=ON`
	`replicate-ignore-db=mysql`
	`binlog-format=ROW`
	`log_bin=mysql-bin`
	Then restart the `mysqld` service.
	`systemctl mysqld restart`
		DBA

Set up replication

Task	Description	Skills required
Export the data dump from the Amazon RDS for MySQL database.	To export the dump from Amazon RDS for MySQL, use the following command. `mysqldump --single-transaction  -h mydb.xxxxxxx.amazonaws.com  -uadmin -p --databases replication > replication-db.sql`	DBA

Task	Description	Skills required
Restore the .sql dump file in the MySQL database on Amazon EC2.	To import the dump to the MySQL database on Amazon EC2, use the following command. `mysql -D replication -uroot -p < replication-db.sql`	DBA

Configure the MySQL database on Amazon EC2 as a replica.	To start the replication and to check the replication status, log in to the MySQL database on Amazon EC2, and use the following command.
	`CHANGE MASTER TO MASTER_HOST="mydb.xxxxxxx.amazonaws.com", MASTER_USER="repl", MASTER_PASSWORD="rep123", MASTER_PORT=3306, MASTER_AUTO_POSITION = 1;`
	`START SLAVE;`
	`SHOW SLAVE STATUS\G`
		DBA

Related resources

- [Amazon EC2 User Guide for Linux Instances]
- [Installing MySQL on Linux Using the MySQL Yum Repository]
- [Replication with Global Transaction Identifiers]
- [Using GTID-based replication for Amazon RDS for MySQL]

Transition roles for an Oracle PeopleSoft application on Amazon RDS Custom for Oracle

sampath kathirvel, Amazon Web Services

Summary

To run the Oracle PeopleSoft enterprise resource planning (ERP) solution on Amazon Web Services (AWS), you can use Amazon Relational Database Service (Amazon RDS) or Amazon RDS Custom for Oracle, which supports legacy, custom, and packaged applications that require access to the underlying operating system (OS) and database environment. For key factors to consider when planning a migration, see Oracle database migration strategies in AWS Prescriptive Guidance.

This pattern focuses on the steps to perform an Oracle Data Guard switchover, or role transition, for a PeopleSoft application database running on Amazon RDS Custom as the primary database with a read replica database. The pattern includes steps to configure fast-start failover (FSFO). During this process, the databases in the Oracle Data Guard configuration continue to function in their new roles. Typical use cases for Oracle Data Guard switchover are disaster recovery (DR) drills, scheduled maintenance activities on databases, and Standby-First Patch Apply rolling patches. For more information, see the blog post Reduce database patching downtime in Amazon RDS Custom.

Prerequisites and limitations

Prerequisites

- Completion of the Add HA to Oracle PeopleSoft on Amazon RDS Custom by using a read replica pattern.

Limitations

- Limitations and unsupported configurations for RDS Custom for Oracle
- Limitations associated with Amazon RDS Custom for Oracle read replicas

Product versions

- For Oracle Database versions supported by Amazon RDS Custom, see RDS Custom for Oracle.
- For Oracle Database instance classes supported by Amazon RDS Custom, see DB instance class support for RDS Custom for Oracle.

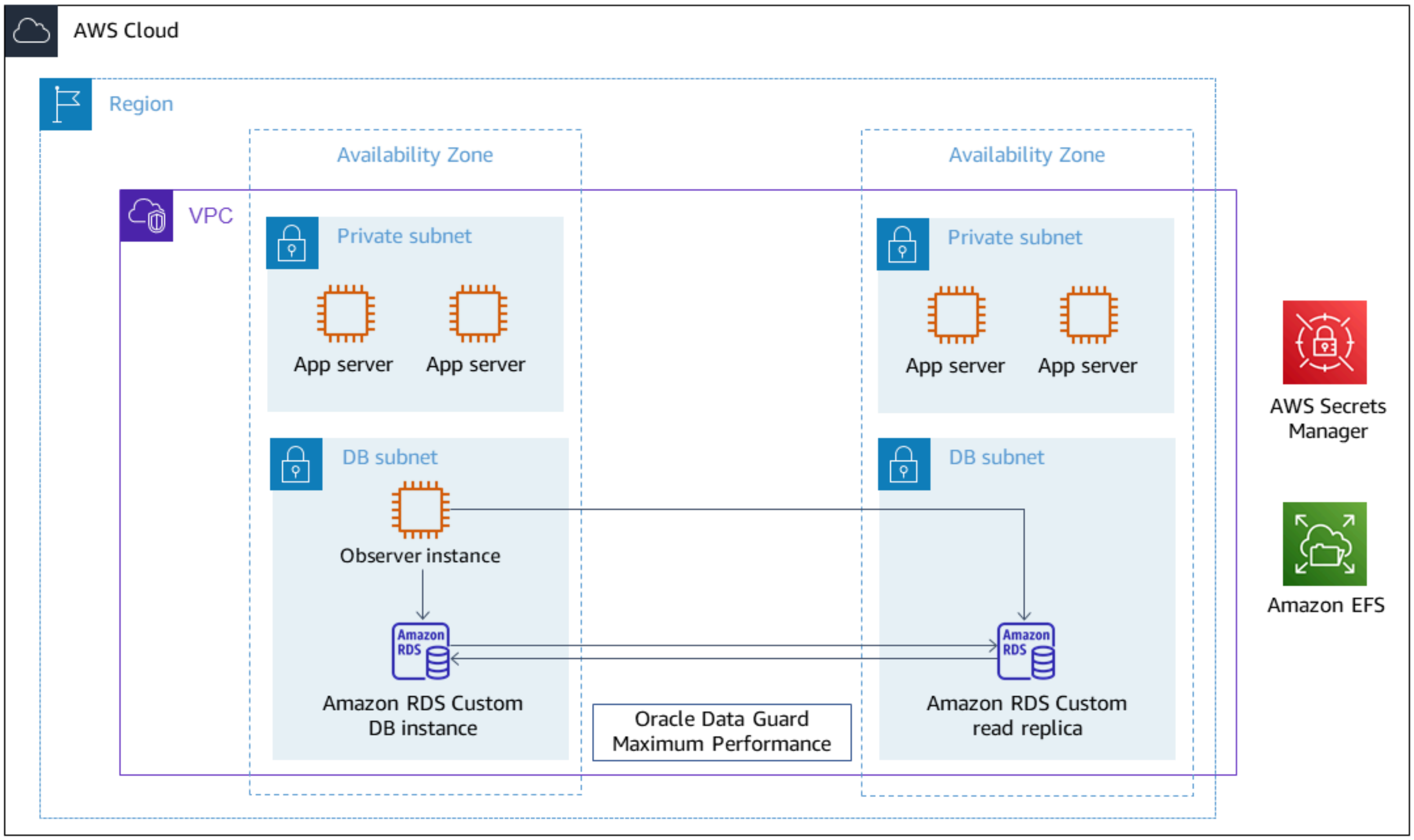
Architecture

Technology stack

- Amazon RDS Custom for Oracle

Target architecture

The following diagram shows an Amazon RDS Custom DB instance and an Amazon RDS Custom read replica. Oracle Data Guard provides role transition during failover for DR.



For a representative architecture using Oracle PeopleSoft on AWS, see [Set up a highly available PeopleSoft architecture on AWS](#).

Tools

AWS services

- [Amazon RDS Custom for Oracle](#) is a managed database service for legacy, custom, and packaged applications that require access to the underlying OS and database environment.

- [AWS Secrets Manager](#) helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. In this pattern, you retrieve the database user passwords from Secrets Manager for `RDS_DATAGUARD` with the secret name `do-not-delete-rds-custom-+<<RDS Resource ID>>+-dg`.

Other services

- [Oracle Data Guard](#) helps you create, maintain, manage, and monitor standby databases. This pattern uses Oracle Data Guard Maximum Performance for transitioning roles ([Oracle Data Guard switchover](#)).

Best practices

For your production deployment, we recommend launching the observer instance in a third Availability Zone, separate from the primary and read replica nodes.

Epics

Initiate role transition

Task	Description	Skills required
Pause database automation for both the primary and the replica.	Although the RDS Custom automation framework doesn't interfere with the role transition process, it's a good practice to pause automatio n during Oracle Data Guard switchover. To pause and resume RDS Custom database automatio n, follow the instructions at [Pausing and resuming RDS Custom automation](#).	Cloud administrator, DBA
Check the Oracle Data Guard status.	To check the Oracle Data Guard status, log in to the	DBA

Task	Description	Skills required

	primary database. This pattern includes code for using a multitenant container database (CDB) or a non-CDB instance.
	
	Non-CDB
	
	`-bash-4.2$ dgmgrl RDS_DATAGUARD@RDS_CUSTOM_ORCL_A`
	`DGMGRL for Linux: Release 19.0.0.0.0 - Production on Mon Nov 28 20:55:50 2022`
	`Version 19.10.0.0.0`
	`Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.`
	`Welcome to DGMGRL, type "help" for information.`
	`Password:`
	`Connected to "ORCL_A"`
	`Connected as SYSDG.`
	`DGMGRL> show configuration`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_a - Primary database`
	`orcl_d - Physical standby database`
	`Fast-Start Failover: Disabled`
	`Configuration Status:`
	`SUCCESS (status updated 59 seconds ago)`	

Task	Description	Skills required

	`DGMGRL>`
	
	CDB
	
	``
	CDB-bash-4.2$ dgmgrl C##RDS_DATAGUARD@RDS_CUSTOM_RDSCDB_A
	DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Jan 18 06:13:07 2023
	Version 19.16.0.0.0
	Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved.
	Welcome to DGMGRL, type "help" for information.
	Password:
	Connected to "RDSCDB_A"
	Connected as SYSDG.
	DGMGRL> show configuration
	Configuration - rds_dg
	  Protection Mode: MaxAvailability
	  Members:
	  rdscdb_a - Primary database
	    rdscdb_b - Physical standby database
	Fast-Start Failover: Disabled
	Configuration Status:
	SUCCESS (status updated 52 seconds ago)
	DGMGRL>
	``	

Task	Description	Skills required
Verify the instance role.	Open the AWS Managemen t Console, and navigate to the Amazon RDS console. In the database's **Replication** section, on the **Connectiv ity & security** tab, verify the instance role for the primary and replica. The primary role should match the Oracle Data Guard primary database, and the replica role should match the Oracle Data Guard physical standby database.	Cloud administrator, DBA

Task	Description	Skills required

Perform the switchover.	To perform the switchover, connect to DGMGRL from the primary node.
	
	Non-CDB
	
	`DGMGRL> switchover to orcl_d;`
	`Performing switchover NOW, please wait...`
	`Operation requires a connection to database "orcl_d"`
	`Connecting ...`
	`Connected to "ORCL_D"`
	`Connected as SYSDG.`
	`New primary database "orcl_d" is opening...`
	`Operation requires start up of instance "ORCL" on database "orcl_a"`
	`Starting instance "ORCL"...`
	`Connected to an idle instance.`
	`ORACLE instance started.`
	`Connected to "ORCL_A"`
	`Database mounted.`
	`Database opened.`
	`Connected to "ORCL_A"`
	`Switchover succeeded, new primary is "orcl_d"`
	`DGMGRL>`
	
	CDB
	
	`DGMGRL> switchover to rdscdb_b`
		DBA

Task	Description	Skills required
	`Performing switchover NOW, please wait... New primary database "rdscdb_b" is opening... Operation requires start up of instance "RDSCDB" on database "rdscdb_a" Starting instance "RDSCDB"... Connected to an idle instance. ORACLE instance started. Connected to "RDSCDB_A " Database mounted. Database opened. Connected to "RDSCDB_A " Switchover succeeded , new primary is "rdscdb_b"`	

Task	Description	Skills required

Verify the Oracle Data Guard connection.	After switchover, verify the Oracle Data Guard connection from the primary node to DGMGRL.
	
	Non-CDB
	
	`DGMGRL> show configuration;`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_d - Primary database`
	`orcl_a - Physical standby database`
	`Fast-Start Failover: Disabled`
	`Configuration Status:`
	`SUCCESS (status updated 60 seconds ago)`
	`DGMGRL>`
	
	`DGMGRL> show configuration lag;`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_d - Primary database`
	`orcl_a - Physical standby database`
	`Transport Lag: 0 seconds (computed 0 seconds ago)`
	`Apply Lag: 0 seconds (computed 0 seconds ago)`
		DBA

Task	Description	Skills required

	``
	Fast-Start Failover:
	 Disabled
	Configuration Status:
	SUCCESS (status updated
	 44 seconds ago)
	DGMGRL>
	``
	CDB
	``
	DGMGRL> show configura
	tion
	DGMGRL> show configura
	tion
	Configuration - rds_dg
	  Protection Mode:
	 MaxAvailability
	  Members:
	  rdscdb_b - Primary
	 database
	    rdscdb_a - Physical
	 standby database
	Fast-Start Failover:
	 Disabled
	Configuration Status:
	SUCCESS (status
	 updated 52 seconds ago)
	DGMGRL>
	
	DGMGRL> show configura
	tion lag
	Configuration - rds_dg
	  Protection Mode:
	 MaxAvailability
	  Members:
	  rdscdb_b - Primary
	 database
	    rdscdb_a - Physical
	 standby database
	               Transport
	 Lag: 0 seconds
	``	

Task	Description	Skills required
	``` (computed 0 seconds ago) Apply Lag: 0 seconds (computed 0 seconds ago) Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 53 seconds ago) DGMGRL> ```	
Verify the instance role on the Amazon RDS console.	After you perform the role switch, the Amazon RDS console shows the new roles under the **Replication** section on the **Connectiv ity & Security** tab under **Databases**. It might take a few minutes for **Replication state** to update from empty to **Replicating**.	DBA

Configure FSFO

Task	Description	Skills required
Reset the switchover.	Set the switchover back to the primary node.	DBA
Install and start the observer.	An observer process is a DGMGRL client component , typically running in a different machine from the primary and standby	DBA

Task	Description	Skills required
	databases. The ORACLE HOME installation for the observer can be an Oracle Client Administrator installat ion, or you can install either Oracle Database Enterprise Edition or Personal Edition. For more information about observer installation for your database release, see Installin g and Starting the Observer. To configure high availabil ity for the observer process, you might want to do the following: • Enable EC2 instance automatic recovery for the EC2 instance running your observer. You need to automate the observer startup process as part of the OS startup. • Deploy an observer in the EC2 instance and configure an Amazon EC2 Auto Scaling group with size one (1). In the event of EC2 instance failure, the automatic scaling group automatically spins up another EC2 instance. For Oracle 12c Release 2 and later, you can deploy up to	

Task	Description	Skills required
	three observers. One observer is the primary observer, and the rest are backup observers . When the primary observer fails, one of the backup observers takes the primary role.	

Task	Description	Skills required

Connect to DGMGRL from the observer host.	The observer host is configured with `tnsnames.ora` entries for primary and standby database connectivity. You can enable FSFO with maximum performance protection mode as long as data loss is within the [FastStartFailoverLagLimit](#) configuration (value in seconds), However, you must use maximum availability protection mode to work to achieve zero data loss (RPO=0).
	
	Non-CDB
	
	`DGMGRL> show configuration;`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_a - Primary database`
	`orcl_d - Physical standby database`
	`Fast-Start Failover: Disabled`
	`Configuration Status:`
	`SUCCESS (status updated 58 seconds ago)`
	`DGMGRL> show configuration lag`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
		DBA

Task	Description	Skills required
	``` Members: orcl_a - Primary database orcl_d - Physical standby database Transport Lag: 0 seconds (computed 1 second ago) Apply Lag: 0 seconds (computed 1 second ago) Fast-Start Failover: Disabled Configuration Status: SUCCESS (status updated 5 seconds ago) DGMGRL> ``` **CDB** ``` -bash-4.2$ dgmgrl C##RDS_DATAGUARD@R DS_CUSTOM_RDSCDB_A DGMGRL for Linux: Release 19.0.0.0.0 - Production on Wed Jan 18 06:55:09 2023 Version 19.16.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Welcome to DGMGRL, type "help" for informati on. Password: Connected to "RDSCDB_A " Connected as SYSDG. DGMGRL> show configura tion Configuration - rds_dg ```	

Task	Description	Skills required

	``
	  Protection Mode:
	 MaxAvailability
	  Members:
	  rdscdb_a - Primary
	 database
	    rdscdb_b - Physical
	 standby database
	Fast-Start Failover:
	 Disabled
	Configuration Status:
	SUCCESS (status
	 updated 18 seconds ago)
	DGMGRL>
	``	

Task	Description	Skills required

Modify the standby database to be the failover target.	Connect from either the primary node or the observer node to one standby database. (Although your onfiguration could have mulitiple standby databases , you need to connect to only one at this time.)
	
	Non-CDB
	
	`DGMGRL> edit database orcl_a set property FastStartFailoverT arget='orcl_d';`
	`Property "faststar tfailovertarget" updated`
	`DGMGRL> edit database orcl_d set property FastStartFailoverT arget='orcl_a';`
	`Property "faststar tfailovertarget" updated`
	`DGMGRL> show database orcl_a FastStart FailoverTarget;`
	`FastStartFailoverTar get = 'orcl_d'`
	`DGMGRL> show database orcl_d FastStart FailoverTarget;`
	`FastStartFailoverTar get = 'orcl_a'`
	`DGMGRL>`
	
	CDB
		DBA

Task	Description	Skills required

	<pre>DGMGRL> edit database
	 orcl_a set property
	 FastStartFailoverT
	arget='rdscdb_b';
	Object "orcl_a" was not
	 found
	DGMGRL> edit database
	 rdscdb_a set property
	 FastStartFailoverT
	arget='rdscdb_b';
	Property "faststar
	tfailovertarget"
	 updated
	DGMGRL> edit database
	 rdscdb_b set property
	 FastStartFailoverT
	arget='rdscdb_a';
	Property "faststar
	tfailovertarget"
	 updated
	DGMGRL> show database
	 rdscdb_a FastStart
	FailoverTarget;
	  FastStartFailoverT
	arget = 'rdscdb_b'
	DGMGRL> show database
	 rdscdb_b FastStart
	FailoverTarget;
	  FastStartFailoverT
	arget = 'rdscdb_a'
	DGMGRL></pre>	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Configure FastStartFailoverT hreshold for the connection to DGMGRL.</td><td>The default value is 30 seconds in Oracle 19c, and the minimum value is 6 seconds. A lower value can potentially shorten the recovery time objective (RTO) during the failover. A higher value helps reduce the chance of unnecessary failover transient errors on the primary database.

The RDS Custom for Oracle automation framework monitors database health and performs corrective actions every few seconds. Therefore, we recommend setting FastStartFailoverT hreshold to a value higher than 10 seconds. The following example configure s the threshold value at 35 seconds.

Non-CBD or CDB

<pre>DGMGRL> edit configura
tion set property
 FastStartFailoverT
hreshold=35;
Property "faststar
tfailoverthreshold"
 updated
DGMGRL> show configura
tion FastStart
FailoverThreshold;</pre></td><td>DBA</td></tr>
</table>

Task	Description	Skills required
	``` FastStartFailover Threshold = '35' DGMGRL> ```	

Task	Description	Skills required

Enable FSFO by connecting to DGMGRL from the primary or observer node.	If the database doesn't have Flashback Database enabled, the warning message `ORA-16827` appears. The optional flashback database helps automatically reinstate failed primary databases to a point in time before failover if the FastStartFailoverAutoReinstate configuration property is set to TRUE (which is the default).
	
	Non-CDB
	
	`DGMGRL> enable fast_start failover;`
	`Warning: ORA-16827: Flashback Database is disabled`
	`Enabled in Zero Data Loss Mode.`
	`DGMGRL>`
	`DGMGRL> show configuration`
	`Configuration - rds_dg`
	`Protection Mode: MaxAvailability`
	`Members:`
	`orcl_a - Primary database`
	`Warning: ORA-16819: fast-start failover observer not started`
	`orcl_d - (*) Physical standby database`
	`Warning: ORA-16819: fast-start failover observer not started`
		DBA

Task	Description	Skills required

	<pre>Fast-Start Failover:
	 Enabled in Zero Data
	 Loss Mode
	Configuration Status:
	WARNING (status updated
	 29 seconds ago)
	DGMGRL></pre>
	CDB
	<pre>DGMGRL> enable fast_star
	t failover;
	Warning: ORA-16827:
	 Flashback Database is
	 disabled
	Enabled in Zero Data
	 Loss Mode.
	DGMGRL> show configura
	tion;
	Configuration - rds_dg
	  Protection Mode:
	 MaxAvailability
	  Members:
	  rdscdb_a - Primary
	 database
	    Warning: ORA-16819
	: fast-start failover
	 observer not started
	    rdscdb_b - (*)
	 Physical standby
	 database
	Fast-Start Failover:
	 Enabled in Zero Data
	 Loss Mode
	Configuration Status:
	WARNING (status
	 updated 11 seconds ago)
	DGMGRL></pre>	

Task	Description	Skills required

Start the observer for FSFO monitoring, and verify the status.	You can start the observer before or after you enable FSFO. If FSFO is already enabled, the observer immediately begins monitoring the status and connections to the primary and target standby databases. If FSFO is not enabled, the observer doesn’t start monitoring until after FSFO is enabled.
	
	When you start the observer, the primary DB configuration will be displayed without any error messages, as evidenced by the previous `show configuration` command.
	
	Non-CDB
	
	`DGMGRL> start observer;`
	`[W000 2022-12-01T06:16:51.271+00:00] FSFO target standby is orcl_d`
	`Observer 'ip-10-0-1-89' started`
	`[W000 2022-12-01T06:16:51.352+00:00] Observer trace level is set to USER`
	
	`DGMGRL> show configuration`
	`Configuration - rds_dg`
		DBA

Task	Description	Skills required
	``` Protection Mode: MaxAvailability Members: orcl_a - Primary database orcl_d - (*) Physical standby database Fast-Start Failover: Enabled in Zero Data Loss Mode Configuration Status: SUCCESS (status updated 56 seconds ago) DGMGRL> DGMGRL> show observer Configuration - rds_dg Primary: orcl_a Active Target: orcl_d Observer "ip-10-0- 1-89" - Master Host Name: ip-10-0-1 -89 Last Ping to Primary: 1 second ago Last Ping to Target: 1 second ago DGMGRL> ``` **CDB** ``` DGMGRL> start observer; Succeeded in opening the observer file "/home/oracle/fsfo _ip-10-0-1-56.dat". [W000 2023-01-1 8T07:31:32.589+00:00] FSFO target standby is rdscdb_b ```	

Task	Description	Skills required

	``
	Observer 'ip-10-0-
	1-56' started
	The observer log file
	 is '/home/oracle/obse
	rver_ip-10-0-1-56.
	log'.
	
	DGMGRL> show configura
	tion
	Configuration - rds_dg
	  Protection Mode:
	 MaxAvailability
	  Members:
	  rdscdb_a - Primary
	 database
	    rdscdb_b - (*)
	 Physical standby
	 database
	Fast-Start Failover:
	 Enabled in Zero Data
	 Loss Mode
	Configuration Status:
	SUCCESS (status
	 updated 12 seconds ago)
	DGMGRL>
	
	DGMGRL> show observer;
	Configuration - rds_dg
	  Primary:
	 rdscdb_a
	  Active Target:
	 rdscdb_b
	Observer "ip-10-0-
	1-56" - Master
	  Host Name:
	        ip-10-0-1-56
	  Last Ping to Primary:
	        1 second ago
	  Last Ping to Target:
	        2 seconds ago
	DGMGRL>
	``	

Task	Description	Skills required
Verify the failover.	In this scenario, a failover test can be performed by manually stopping the primary EC2 instance. Before stopping the EC2 instance, use the `tail` command to monitor the observer log file based on your configura tion. Use `DGMGRL` to log in to standby database `orcl_d` with user `RDS_DATAGUARD` , and check the Oracle Data Guard status. It should show that `orcl_d` is the new primary database. **Note** In this failover testing scenario, `orcl_d` is the non-CDB database. Before failover, the flashback database has been enabled on `orcl_a`. After the former primary database returns online and starts in `MOUNT` state, the observer reinstate s it into a new standby database. The reinstated database acts as the FSFO target for the new primary	DBA

Task	Description	Skills required

	database. You can verify the details in observer logs.
	``
	DGMGRL> show configura
	tion
	Configuration - rds_dg
	Protection Mode:
	 MaxAvailability
	Members:
	orcl_d - Primary
	 database
	Warning: ORA-16824
	: multiple warnings,
	 including fast-star
	t failover-related
	 warnings, detected for
	 the database
	orcl_a - (*) Physical
	 standby database
	 (disabled)
	ORA-16661: the standby
	 database needs to be
	 reinstated
	Fast-Start Failover:
	 Enabled in Zero Data
	 Loss Mode
	Configuration Status:
	WARNING (status updated
	 25 seconds ago)
	DGMGRL>
	``
	The following shows example output in `observer.log` .
	``
	$ tail -f /tmp/obse
	rver.log
	Unable to connect
	 to database using
	 rds_custom_orcl_a
	``	

Task	Description	Skills required

	`[W000 2023-01-18T07:50:32.589+00:00] Primary database cannot be reached.`
	`[W000 2023-01-18T07:50:32.589+00:00] Fast-Start Failover threshold has expired.`
	`[W000 2023-01-18T07:50:32.590+00:00] Try to connect to the standby.`
	`[W000 2023-01-18T07:50:32.590+00:00] Making a last connection attempt to primary database before proceeding with Fast-Start Failover.`
	`[W000 2023-01-18T07:50:32.591+00:00] Check if the standby is ready for failover.`
	`[S002 2023-01-18T07:50:32.591+00:00] Fast-Start Failover started...`
	`2023-01-18T07:50:32.591+00:00 Initiating Fast-Start Failover to database "orcl_d"...`
	`[S002 2023-01-18T07:50:32.592+00:00] Initiating Fast-start Failover.`
	`Performing failover NOW, please wait...`
	`Failover succeeded, new primary is "orcl_d"`
	`2023-01-18T07:55:32.101+00:00`	

Task	Description	Skills required
	``` [S002 2023-01-1 8T07:55:32.591+00:00] Fast-Start Failover finished... [W000 2023-01-1 8T07:55:32.591+00:00] Failover succeeded. Restart pinging. [W000 2023-01-1 8T07:55:32.603+00:00] Primary database has changed to orcl_d. [W000 2023-01-1 8T07:55:33.618+00:00] Try to connect to the primary. [W000 2023-01-1 8T07:55:33.622+00: 00] Try to connect to the primary rds_custo m_orcl_d. [W000 2023-01-1 8T07:55:33.634+00: 00] The standby orcl_a needs to be reinstated [W000 2023-01-1 8T07:55:33.654+00:00] Try to connect to the new standby orcl_a. [W000 2023-01-1 8T07:55:33.654+00: 00] Connection to the primary restored! [W000 2023-01-1 8T07:55:35.654+00: 00] Disconnecting from database rds_custo m_orcl_d. [W000 2023-01-1 8T07:55:57.701+00:00] Try to connect to the new standby orcl_a. ```	

Task	Description	Skills required
	`ORA-12170: TNS:Connect timeout occurred`	

Configure connectivity between the Oracle Peoplesoft application and the database

Task	Description	Skills required

Create and start the service in the primary database.	You can avoid application configuration changes during a role transition by using a TNS entry that contains both the primary and standby database endpoints in the configuration. You can define two role-based database services to support both read/write and read-only workloads. In the following example, `orcl_rw` is the read/write service that's active on the primary database. `orcl_ro` is the read-only service and is active on the standby database that has been opened in read-only mode.
	`SQL> select name,open_mode from v$database;`
	`NAME OPEN_MODE`
	`--------- --------- -----------`
	`ORCL READ WRITE`
	`SQL> exec dbms_service.create_service`
		DBA

Task	Description	Skills required

	<pre>('orcl_rw','orcl_r
	w');
	PL/SQL procedure
	 successfully completed
	.
	SQL> exec dbms_serv
	ice.create_service
	('orcl_ro','orcl_r
	o');
	PL/SQL procedure
	 successfully completed
	.
	
	SQL> exec dbms_serv
	ice.start_service(
	'orcl_rw');
	PL/SQL procedure
	 successfully completed
	.
	SQL></pre>	

Start the service in the standby database.	To start the service in the read-only standby database, use the following code.
	<pre>SQL> select name,open
	_mode from v$database;
	NAME OPEN_MODE
	--------- ---------
	----------- 	
	ORCL READ ONLY WITH
	 APPLY
	SQL> exec dbms_serv
	ice.start_service(
	'orcl_ro');
	PL/SQL procedure
	 successfully completed
	.
	SQL></pre>
		DBA

Task	Description	Skills required
Automate starting the service when the primary DB is restarted.	To automatically start the service in the primary database when it's restarted, use the following code. `SQL> CREATE OR REPLACE TRIGGER TrgDgServices after startup on database DECLARE db_role VARCHAR(30); db_open_mode VARCHAR(30); BEGIN SELECT DATABASE_ROLE, OPEN_MODE INTO db_role, db_open_mode FROM V$DATABASE; IF db_role = 'PRIMARY' THEN DBMS_SERV 2 ICE.START_SERVICE('orcl_rw'); END IF; IF db_role = 'PHYSICAL STANDBY' AND db_open_mode LIKE 'READ ONLY%' THEN DBMS_SERVICE.START_SERVICE('orcl_ro'); END IF; END; / Trigger created. SQL>`	DBA

Task	Description	Skills required

Configure a connection between the read/write and read-only databases.	You can use the following application-configuration example for the read/write and read-only connection.
	`ORCL_RW = (DESCRIPTION =`
	`(CONNECT_TIMEOUT= 120)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)`
	`(ADDRESS_LIST =`
	`(ADDRESS = (PROTOCOL = TCP)(HOST=devpsftdb.******.us-west-2.rds.amazonaws.com)(PORT=1521))`
	`(ADDRESS = (PROTOCOL = TCP)(HOST=psftread.******.us-west-2.rds.amazonaws.com)(PORT=1521))`
	`)`
	`(CONNECT_DATA=(SERVICE_NAME = orcl_rw))`
	`)`
	`ORCL_RO = (DESCRIPTION =`
	`(CONNECT_TIMEOUT= 120)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)`
	`(ADDRESS_LIST =`
	`(ADDRESS = (PROTOCOL = TCP)(HOST=devpsftdb.******.us-west-2.rds.amazonaws.com)(PORT=1521))`
		DBA

Task	Description	Skills required
	`(ADDRESS = (PROTOCOL = TCP)(HOST=psftread.******.us-west-2.rds.amazonaws.com)(PORT=1521)) ) (CONNECT_DATA=(SERVICE_NAME = orcl_ro)) )`	

Related resources

- Enabling High Availability with Data Guard on Amazon RDS Custom for Oracle (AWS Technical Guide)
- Configuring Amazon RDS as an Oracle PeopleSoft Database (AWS whitepaper)
- Oracle Data Guard Broker guide (Oracle reference documentation)
- Data Guard Concepts and Administration (Oracle reference documentation)
- Oracle Data Guard Specific FAN and FCF Configuration Requirements (Oracle reference documentation)

Unload data from an Amazon Redshift cluster across accounts to Amazon S3

Andrew Kamel, Amazon Web Services

Summary

When you test applications, it's helpful to have production data in your test environment. Using production data can give you a more accurate assessment of the application that you're developing.

This pattern extracts data from an Amazon Redshift cluster in a production environment to an Amazon Simple Storage Service (Amazon S3) bucket in a development environment on Amazon Web Services (AWS).

The pattern steps through the setup of both DEV and PROD accounts, including the following:

- Required resources
- AWS Identity and Access Management (IAM) roles
- Network adjustments to subnets, security groups, and the virtual private cloud (VPC) to support the Amazon Redshift connection
- An example AWS Lambda function with a Python runtime for testing the architecture

To grant access to the Amazon Redshift cluster, the pattern uses AWS Secrets Manager to store the relevant credentials. The benefit is having all the needed information to directly connect to the Amazon Redshift cluster without needing to know where the Amazon Redshift cluster resides. Additionally, you can monitor use of the secret.

The secret saved in Secrets Manager includes the Amazon Redshift cluster's host, database name, port, and relevant credentials.

For information about security considerations when using this pattern, see the Best practices section.

Prerequisites and limitations

Prerequisites

- An Amazon Redshift cluster running in the PROD account
- An S3 bucket created in the DEV account
- VPC peering between the DEV and PROD accounts, with route tables adjusted accordingly
- DNS hostnames and DNS resolution enabled for both peered VPCs

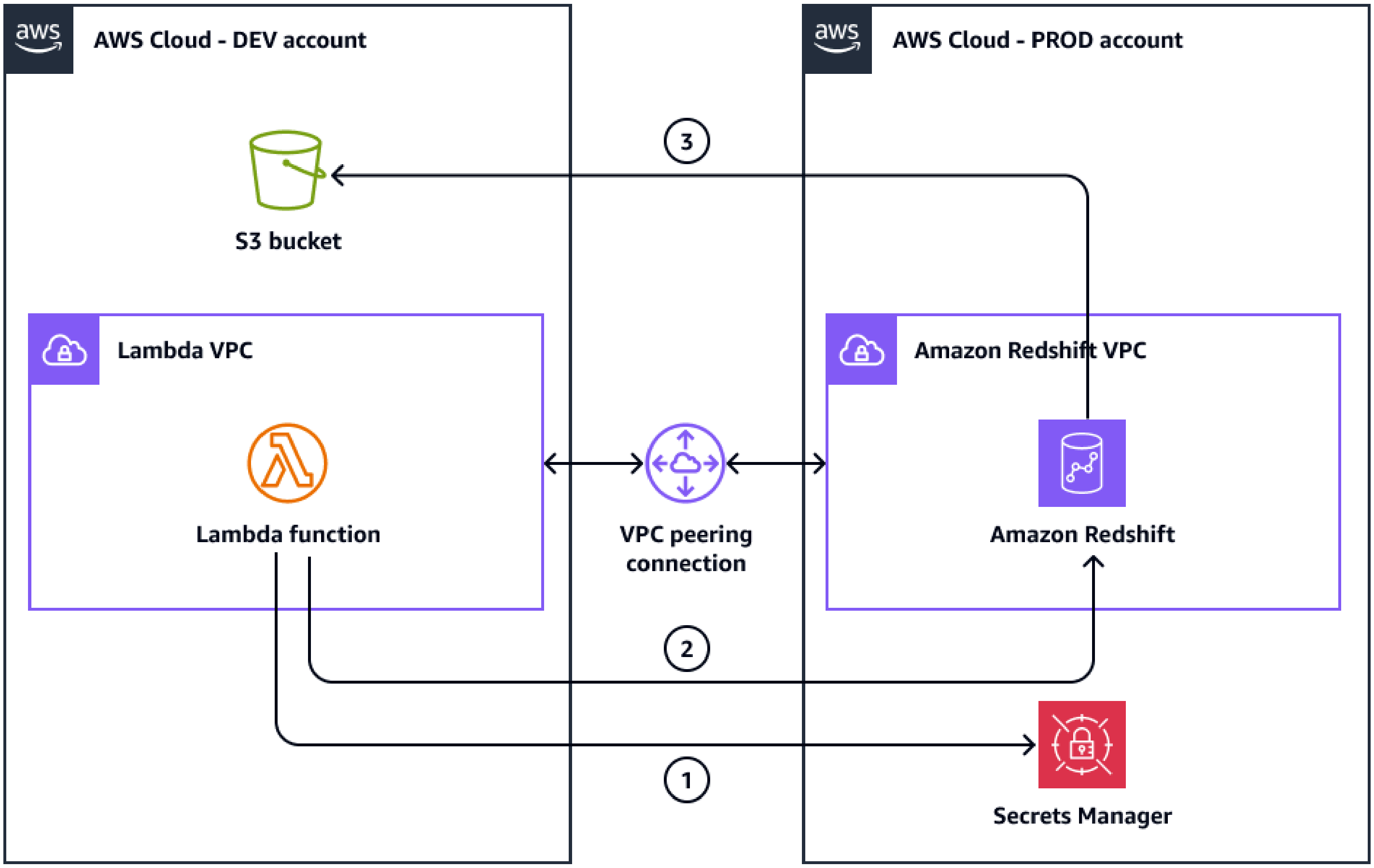
Limitations

- Depending on the amount of data that you want to query, the Lambda function might time out.

  If your run takes more time than the maximum Lambda timeout (15 minutes), use an asynchronous approach for your Lambda code. The code example for this pattern uses the psycopg2 library for Python, which doesn't currently support asynchronous processing.
- Some AWS services aren’t available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

The following diagram shows the target architecture, with DEV and PROD accounts.



The diagram shows the following workflow:

1. The Lambda function in the DEV account assumes the IAM role that's required to access the Amazon Redshift credentials in Secrets Manager in the PROD account.

   The Lambda function then retrieves the Amazon Redshift cluster secret.

2. The Lambda function in the DEV account uses the information to connect to the Amazon Redshift cluster in the PROD account through the peered VPCs.

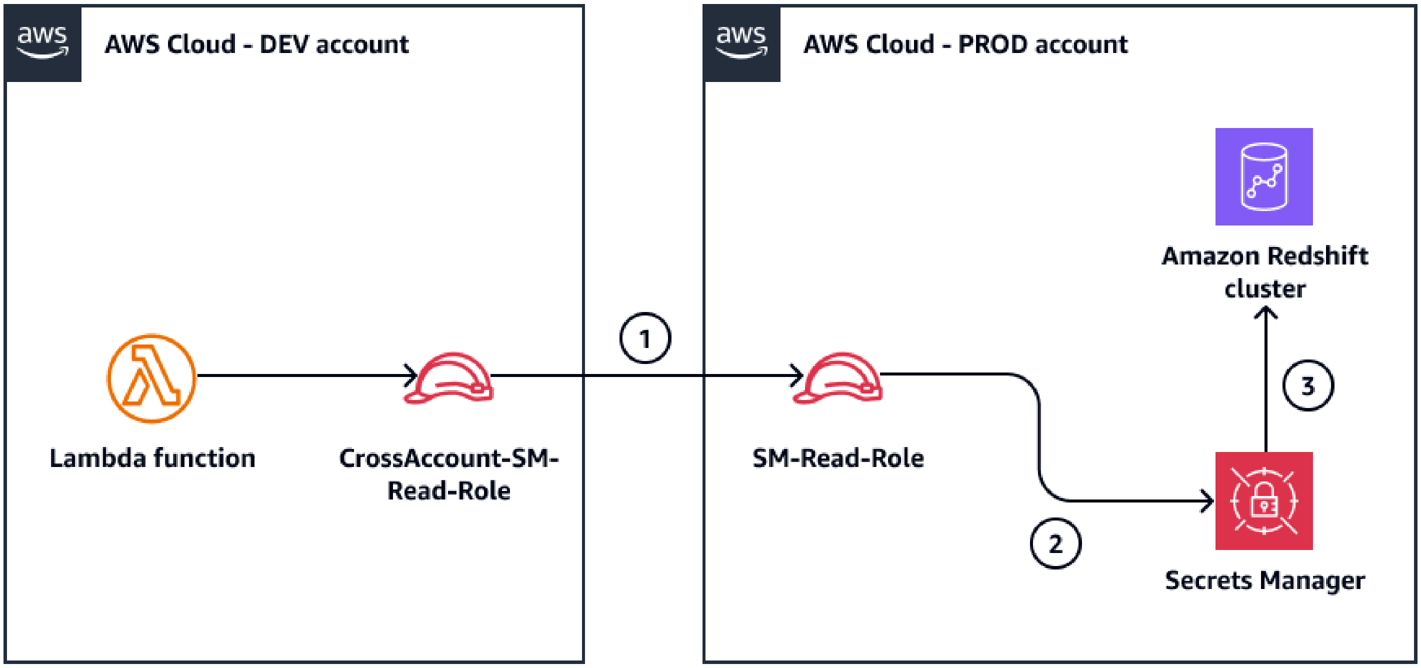
   The Lambda function then sends an unload command to query the Amazon Redshift cluster in the PROD account.

3. The Amazon Redshift cluster in the PROD account assumes the relevant IAM role to access the S3 bucket in the DEV account.

   The Amazon Redshift cluster unloads the queried data to the S3 bucket in the DEV account.

Querying data from Amazon Redshift

The following diagram shows the roles that are used to retrieve the Amazon Redshift credentials and connect to the Amazon Redshift cluster. The workflow is initiated by the Lambda function.

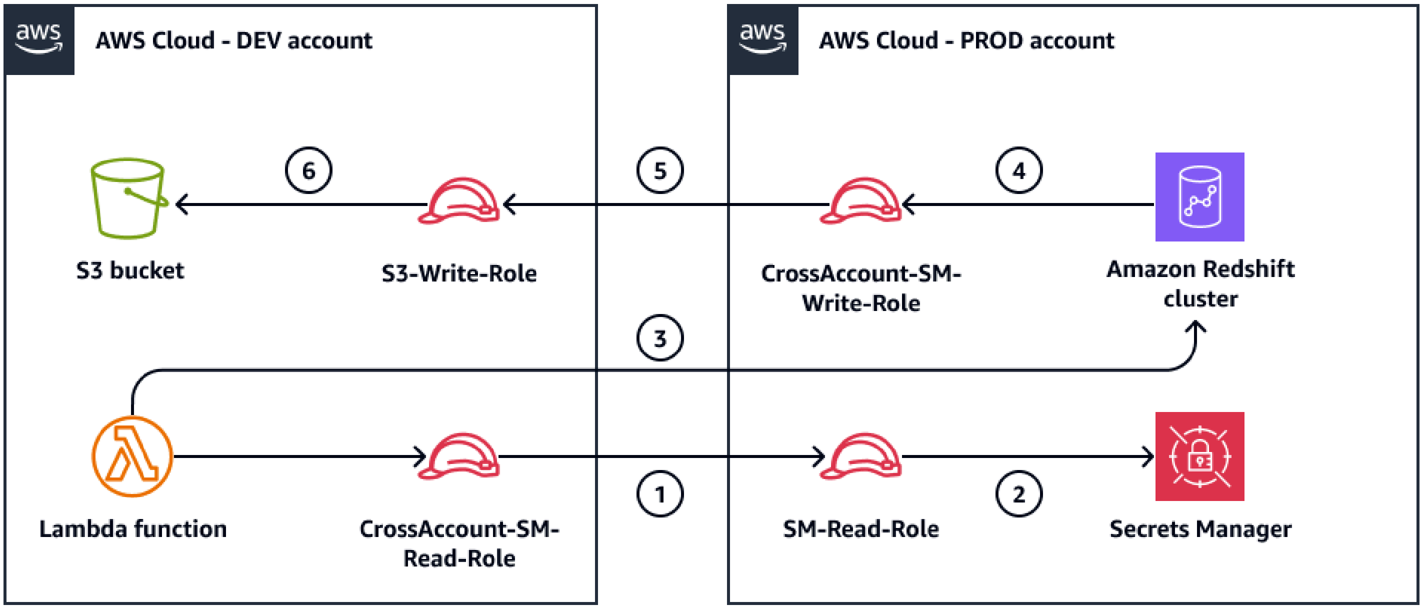


The diagram shows the following workflow:

1. The `CrossAccount-SM-Read-Role` in the DEV account assumes the `SM-Read-Role` in the PROD account.
2. The `SM-Read-Role` role uses the attached policy to retrieve the secret from Secrets Manager.
3. The credentials are used to access the Amazon Redshift cluster.

Uploading data to Amazon S3

The following diagram shows the cross-account read-write process for extracting data and uploading it to Amazon S3. The workflow is initiated by the Lambda function. The pattern chains IAM roles in Amazon Redshift. The unload command that comes from the Amazon Redshift cluster assumes the `CrossAccount-S3-Write-Role`, and then assumes the `S3-Write-Role`. This role chaining gives Amazon Redshift access to Amazon S3.



The workflow includes the following steps:

1. The `CrossAccount-SM-Read-Role` in the DEV account assumes the `SM-Read-Role` in the PROD account.
2. The `SM-Read-Role` retrieves the Amazon Redshift credentials from Secrets Manager.
3. The Lambda function connects to the Amazon Redshift cluster and sends a query.
4. The Amazon Redshift cluster assumes the `CrossAccount-S3-Write-Role`.
5. The `CrossAccount-S3-Write-Role` assumes the `S3-Write-Role` in the DEV account.
6. The query results are unloaded to the S3 bucket in the DEV account.

Tools

AWS services

- [AWS Key Management Service (AWS KMS)](#) helps you create and control cryptographic keys to help protect your data.
- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [Amazon Redshift](#) is a managed petabyte-scale data warehouse service in the AWS Cloud.

- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code repository

The code for this pattern is available in the GitHub unload-redshift-to-s3-python repository.

Best practices

Security disclaimer

Before you implement this solution, consider the following important security recommendations:

- Remember that connecting development and production accounts can increase the scope and lower overall security posture. We recommend deploying this solution only temporarily, extracting the required portion of data and then immediately destroying the deployed resources. To destroy the resources, you should delete the Lambda function, remove any IAM roles and policies created for this solution, and revoke any network access that was granted between the accounts.
- Consult your security and compliance teams before copying any data from production to development environments. Personally identifiable information (PII), Protected health information (PHI), and other confidential or regulated data should generally not be copied in this manner. Copy only publicly available, non-confidential information (for example, public stock data from a shop frontend). Consider tokenizing or anonymizing data, or generating synthetic test data, instead of using production data whenever possible. One of the AWS security principles is to keep people away from data. In other words, developers should not perform operations in the production account.
- Restrict access to the Lambda function in the development account because it can read data from the Amazon Redshift cluster in the production environment.
- To avoid disrupting the production environment, implement the following recommendations:
  - Use a separate, dedicated development account for testing and development activities.
  - Implement strict network access controls and limit traffic between accounts to only what is necessary.
  - Monitor and audit access to the production environment and data sources.
  - Implement least-privilege access controls for all resources and services involved.

  - Regularly review and rotate credentials, such as AWS Secrets Manager secrets and IAM role access keys.
- Refer to the following security documentation for the services used in this article:
  - AWS Lambda security
  - Amazon Redshift security
  - Amazon S3 security
  - AWS Secrets Manager security
  - IAM security best practices

Security is a top priority when accessing production data and resources. Always follow best practices, implement least-privilege access controls, and regularly review and update your security measures.

Epics

Query data from Amazon Redshift

Task	Description	Skills required
Create a secret for the Amazon Redshift cluster.	To create the secret for the Amazon Redshift cluster, do the following: 1. In the PROD account, sign in to the AWS Management Console, and open the Secrets Manager console at https://console.aws.amazon.com/secretsmanager/. 2. Choose **Store a new Secret**. 3. Select **Credentials for Amazon Redshift data warehouse**.	DevOps engineer

Task	Description	Skills required
	4. For **User name** and **Password**, enter values for your instance, and confirm or choose a value for **Encryption key**. 5. Choose the Amazon Redshift data warehouse that your secret will access. 6. Enter `Redshift-Creds-Secret` for the secret name. 7. Complete the remaining the creation steps with the default choices, and then choose **Store**. 8. View your secret, and note the **Secret ARN** value that was generated to identify the secret.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a role to access Secrets Manager.</td><td>To create the role, do the following:

1. In the PROD account, open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose Policies.
3. Choose Create policy.
4. Choose the JSON tab, and then enter an IAM policy such as the following:
<pre>{
    "Version":
 "2012-10-17",
    "Statement": [
        {
            "Effect":
 "Allow",
            "Action":
 [

 "secretsmanager:Ge
tResourcePolicy",

 "secretsmanager:Ge
tSecretValue",

 "secretsmanager:De
scribeSecret",

 "secretsmanager:Li
stSecretVersionIds"
            ],
            "Resource
": [</pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required

	``
	 "<Redshift-Creds-S
	ecret-ARN>"
	            ]
	        },
	        {
	            "Effect":
	 "Allow",
	            "Action":
	 "secretsmanager:Li
	stSecrets",
	            "Resource
	": "*"
	        }
	    ]
	}
	``
	Replace `Redshift-Creds-Secret-ARN` with the Amazon Resource Name (ARN) of the Secrets Manager secret that contains the information and credentials for the Amazon Redshift cluster.	

Upload data to Amazon S3

Task	Description	Skills required
Create a role to access the S3 bucket.	To create the role for accessing the S3 bucket, do the following: 1. In the DEV account, open the IAM console. 2. Choose **Policies**.	DevOps engineer

Task	Description	Skills required

	3. Choose **Create policy**.
	4. Choose the **JSON** tab, and then enter an IAM policy such as the following:
	<pre>{
	  "Version":
	 "2012-10-17",
	  "Statement": [
	    {
	      "Sid": "kmsstmt"
	,
	      "Effect":
	 "Allow",
	      "Action": [
	        "kms:Decr
	ypt",
	        "kms:Encr
	ypt",
	        "kms:Gene
	rateDataKey"
	      ],
	      "Resource": [
	        "<kms-key-
	arn>"
	      ]
	    },
	    {
	      "Sid":
	 "s3stmt",
	      "Effect":
	 "Allow",
	      "Action": [
	        "s3:PutOb
	ject",
	        "s3:Get*",
	        "s3:List*"
	      ],
	      "Resource": [
	        "arn:aws:
	s3:::mybucket",</pre>	

Task	Description	Skills required

	``
	        "arn:aws:
	s3:::mybucket/*"
	      ]
	    }
	  ]
	}
	``
	Replace `mybucket` with the name of the S3 bucket that you want to access. Also, if the S3 bucket is encrypted, replace `kms-key-arn` with the ARN of the AWS Key Management Service (AWS KMS) key used to encrypt the S3 bucket. Otherwise, you don't need the AWS KMS section in the policy.
	5. Choose **Review policy,** enter `S3-Write-Policy` as the policy name, and then choose **Create policy**.
	6. In the navigation pane, choose **Roles.**
	7. Choose **Create role**.
	8. For the trusted entity role, choose **Custom trust policy**.
	9. Choose **Next: Permissions**, and then select the **S3-Write-Policy** policy that you created.	

Task	Description	Skills required
	10. Enter `S3-Write-Role` as the role name, and then choose **Create role**.	

Task	Description	Skills required
Create the Amazon Redshift role.	To create the Amazon Redshift role, do the following: 1. In the PROD account, open the IAM console. 2. Choose **Policies**. 3. Choose **Create policy**. 4. Choose the **JSON** tab, and then enter an IAM policy such as the following: `{ "Version": "2012-10-17", "Statement": [ { "Sid": "CrossAccountPolicy", "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": "S3-Write-Role-ARN" } ] }` Replace `S3-Write-Role-ARN` with the ARN for the `S3-Write-Role` in the DEV account. 5. Choose **Review policy,** enter `S3-Write-Role-`	DevOps engineer

Task	Description	Skills required
	`Assume-Policy` as the policy name, and then choose **Create policy**. 6. In the navigation pane, choose **Roles**, and then choose **Create role**. 7. Choose **AWS service** as your trusted entity type, and then choose **Redshift**, **Redshift - Customizable**. 8. Choose **Next: Permissions**, and then select the **S3-Write-Role-Assume-Policy** policy that you created. 9. Enter `CrossAccount-S3-Write-Role` as the role name, and then choose **Create role**. 10. Associate the IAM role with your Amazon Redshift cluster.	

Deploy the Lambda function

Task	Description	Skills required
Deploy the Lambda function.	To deploy a Lambda function in the peered VPC, do the following: 1. Open the Lambda console at https://console.aws.amazon.com/lambda/.	DevOps engineer

Task	Description	Skills required
	2. Choose **Functions**. 3. Choose **Create function**. 4. Under **Basic information**, for **Function name**, enter a name for your function. 5. For **Runtime**, choose **Python 3.8**. 6. Expand **Change default execution role**, and then do the following: a. Choose **Use an existing role**. b. For **Existing role**, select the **CrossAccount-RM-Read-Role** Lambda role that you created previously. 7. Expand **Advanced settings**, and do the following: a. Select the **Enable VPC** check box. b. For **VPC**, select the peered VPC in the DEV account. c. For **Subnets**, select the private subnet. d. For **Security groups**, select the default security group. 8. Choose **Create function.**	

Task	Description	Skills required
	9. Add the psycopg2 library as a layer to the Lambda function. **Note** You can use an already deployed layer from the psycopg2-lambda-layer repository. Make sure to use the URL based on your AWS Region and Python runtime.	

Test the architecture

Task	Description	Skills required

Import the required resources.	To import the required resources, run the following commands:
	
	`import ast`
	`import boto3`
	`import psycopg2`
	`import base64`
	`from botocore.exceptions import ClientError`
		App developer

Run the Lambda handler function.	The Lambda function uses AWS Security Token Service	App developer

Task	Description	Skills required

	(AWS STS) for cross-account access and temporary credential management. The function uses the AssumeRole API operation to temporarily assume the permissions of the `sm_read_role` IAM role.
	
	To run the Lambda function, use the following example code:
	
	`def lambda_handler(event, context):`
	`    sts_client = boto3.client('sts')`
	
	`    # Secrets Manager Configurations`
	`    secret_name = "redshift_creds"`
	`    sm_region = "eu-west-1"`
	`    sm_read_role = "arn:aws:iam::PROD_ACCOUNT_NUMBER:role/SM-Read-Role"`
	
	`    # S3 Bucket Configurations`
	`    s3_bucket_path = "s3://mybucket/"`
	`    s3_bucket_region = "eu-west-1"`
	`    s3_write_role = "arn:aws:iam::DEV_ACCOUNT_NUMBER:role/S3-Write-Role"`	

Task	Description	Skills required

	``
	    # Redshift Configura
	tions
	    sql_query = "select
	 * from category"
	    redshift_db = "dev"
	    redshift_s3_write_
	role = "arn:aws:
	iam::PROD_ACCOUNT_
	NUMBER:role/CrossA
	ccount-S3-Write-Role"
	
	    chained_s3_write_r
	ole = "%s,%s" %
	 (redshift_s3_write
	_role, s3_write_role)
	
	    assumed_role_objec
	t = sts_client.assume_
	role(
	        RoleArn=s
	m_read_role,
	        RoleSessi
	onName="CrossAccou
	ntRoleAssumption",
	        ExternalI
	d="YOUR_EXTERNAL_ID",
	    )
	    credentials =
	 assumed_role_objec
	t['Credentials']
	
	    secret_dict =
	 ast.literal_eval(g
	et_secret(credenti
	als, secret_name,
	 sm_region))
	    execute_query(secr
	et_dict, sql_query
	, s3_bucket_path,
	 chained_s3_write_r
	ole, s3_bucket_region,
	 redshift_db)
	``	

Task	Description	Skills required

	``
	    return {
	        'statusCode':
	200
	    }
	``	

Task	Description	Skills required

Get the secret.	To get the Amazon Redshift secret, use the following example code:
	
	`def get_secret(credentials, secret_name, sm_region):`
	`    # Create a Secrets Manager client`
	`    session = boto3.session.Session()`
	`    sm_client = session.client(`
	`        service_name='secretsmanager',`
	`        aws_access_key_id=credentials['AccessKeyId'],`
	`        aws_secret_access_key=credentials['SecretAccessKey'],`
	`        aws_session_token=credentials['SessionToken'],`
	`        region_name=sm_region`
	`    )`
	
	`    try:`
	`        get_secret_value_response = sm_client.get_secret_value(`
	`            SecretId=secret_name`
	`        )`
	`    except ClientError as e:`
	`        print(e)`
	`        raise e`
		App developer

Task	Description	Skills required

	<pre> else:
	        if 'SecretSt
	ring' in get_secre
	t_value_response:
	            return
	 get_secret_value_r
	esponse['SecretStr
	ing']
	        else:
	            return
	 base64.b64decode(g
	et_secret_value_re
	sponse['SecretBina
	ry'])</pre>	

Task	Description	Skills required

Run the unload command.	To unload the data to the S3 bucket, use the following example code.
	<pre>def execute_query(secr
	et_dict, sql_query
	, s3_bucket_path,
	 chained_s3_write_r
	ole, s3_bucket_region,
	 redshift_db):
	    conn_string =
	 "dbname='%s' port='%s'
	 user='%s' password=
	'%s' host='%s'" \
	                  %
	 (redshift_db,
	
	 secret_dict["port"],
	
	 secret_dict["usern
	ame"],
	
	 secret_dict["passw
	ord"],
	
	 secret_dict["host"])
	
	    con = psycopg2.
	connect(conn_string)
	
	    unload_command =
	 "UNLOAD ('{}') TO '{}'
	 IAM_ROLE '{}' DELIMITER
	 '|' REGION '{}';" \
	        .format(s
	ql_query,
	
	 s3_bucket_path +
	 str(datetime.datet
	ime.now()) + ".csv",</pre>
		App developer

Task	Description	Skills required

	<pre> chained_s3_write_r
	ole,
	
	 s3_bucket_region)
	
	    # Opening a cursor
	 and run query
	    cur = con.cursor()
	    cur.execute(unload
	_command)
	
	    print(cur.fetchone
	())
	    cur.close()
	    con.close()</pre>	

Clean up

Task	Description	Skills required
Delete the Lambda function.	To avoid incurring unplanned costs, remove the resources and the connection between the DEV and PROD accounts. To remove the Lambda function, do the following: 1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/. 2. Locate and select the Lambda function you created.	DevOps engineer

Task	Description	Skills required
	3. Choose **Actions**, and then choose **Delete**. 4. Confirm the deletion.	

Task	Description	Skills required

Remove the IAM roles and policies.	Remove the IAM roles and policies from the DEV and PROD accounts.
	
	In the DEV account, do the following:
	
	1. Open the IAM console.
	2. Delete the following roles:
	  • `S3-Write-Role`
	  • `CrossAccount-RM-Read-Role` (Lambda role)
	3. Delete the associated policies:
	  • `S3-Write-Policy`
	  • The CrossAccount policy for assuming PROD account roles
	
	In the PROD account, do the following:
	
	1. Open the IAM console.
	2. Delete the following roles:
	  • `SM-Read-Role`
	  • `CrossAccount-S3-Write-Role`
	3. Delete the associated policies:
	  • The CrossAccount policy for accessing Secrets Manager
		DevOps engineer

Task	Description	Skills required
	• `S3-Write-Role-Assume-Policy`	
Delete the secret in Secrets Manager.	To delete the secret, do the following: 1. In the PROD account, open the Secrets Manager console. 2. Locate and select the secret named `Redshift-Creds-Secret` . 3. Choose **Actions**, and then choose **Delete secret**. 4. Confirm the deletion.	DevOps engineer

Task	Description	Skills required
Remove VPC peering and security group rules.	To remove VPC peering and security group rules, do the following: 1. In the PROD account, open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2. Navigate to **Security Groups**. 3. Find the security group used by the Amazon Redshift cluster. 4. Edit the inbound rules, and remove the rule that allows connections from the DEV account's Lambda VPC. 5. Navigate to **VPC Peering Connections**, and delete the peering connection.	DevOps engineer
Remove data from the S3 bucket.	To remove the data from Amazon S3, do the following: 1. In the DEV account, open the Amazon S3 console at https://console.aws.amazon.com/s3/. 2. Locate the bucket you used for data storage. 3. Delete the objects within the bucket, or delete the entire bucket if it's no longer needed.	DevOps engineer

Task	Description	Skills required
Clean up AWS KMS keys.	If you created any custom AWS KMS keys for encryption, do the following: 1. Open the AWS KMS console at https://console.aws.amazon.com/kms/. 2. Locate any keys created for this pattern. 3. Schedule the keys for deletion. (There is a mandatory waiting period for key deletion).	DevOps engineer
Review and delete Amazon CloudWatch logs.	To delete the CloudWatch logs, do the following: 1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/. 2. Check for any log groups created by your Lambda function or Amazon Redshift cluster. 3. Delete these log groups if they are no longer needed.	DevOps engineer

Related resources

- Amazon CloudWatch documentation
- IAM documentation
- Lambda documentation

- Amazon Redshift documentation
- Amazon S3 documentation
- AWS Secrets Manager documentation
- AWS security principles

Additional information

After you unload the data from Amazon Redshift to Amazon S3, you can analyze it by using Amazon Athena.

Amazon Athena is a big data query service that's beneficial when you need to access large volumes of data. You can use Athena without having to provision servers or databases. Athena supports complex queries, and you can run it on different objects.

As with most AWS services, the main benefit to using Athena is that it provides great flexibility in how you run queries without the added complexity. When you use Athena, you can query different data types, such as CSV and JSON, in Amazon S3 without changing the data type. You can query data from various sources, including outside AWS. Athena reduces complexity because you don't have to manage servers. Athena reads data directly from Amazon S3 without loading or changing the data before you run the query.

Database migration patterns by workload

Topics

- IBM
- Microsoft
- N/A
- Open source
- Oracle
- SAP

IBM

- Migrate a Db2 database from Amazon EC2 to Aurora MySQL-Compatible by using AWS DMS
- Migrate Db2 for LUW to Amazon EC2 by using log shipping to reduce outage time
- Migrate Db2 for LUW to Amazon EC2 with high availability disaster recovery
- Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL-Compatible using AWS DMS and AWS SCT
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2
- Stream data from IBM Db2, SAP, Sybase, and other databases to MongoDB Atlas on AWS

Microsoft

- Accelerate the discovery and migration of Microsoft workloads to AWS
- Access on-premises Microsoft SQL Server tables from Microsoft SQL Server on Amazon EC2 using linked servers
- Assess query performance for migrating SQL Server databases to MongoDB Atlas on AWS
- Automate database tasks in SQL Server Express edition running on Amazon EC2 by using AWS Lambda and Task Scheduler
- Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition
- Configure read-only routing in an Always On availability group in SQL Server on AWS
- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python
- Deploy SQL Server failover cluster instances on Amazon EC2 and Amazon FSx by using Terraform
- Export a Microsoft SQL Server database to Amazon S3 by using AWS DMS
- Export Amazon RDS for SQL Server tables to an S3 bucket by using AWS DMS
- Implement SHA1 hashing for PII data when migrating from SQL Server to PostgreSQL
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account
- Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS
- Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT
- Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using native backup and restore methods
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents

- Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux
- Migrate data from Microsoft Azure Blob to Amazon S3 by using Rclone
- Migrate Microsoft SQL Server Always On availability group using AWS Application Migration Service
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using Application Migration Service
- Migrate a relational database to MongoDB Atlas on AWS
- Migrate SQL Server to AWS using distributed availability groups
- Migrate Windows SSL certificates to an Application Load Balancer using ACM
- Rehost on-premises workloads in the AWS Cloud: migration checklist
- Resolve connection errors after migrating Microsoft SQL Server to the AWS Cloud
- Send notifications for an Amazon RDS for SQL Server database instance by using an on-premises SMTP server and Database Mail
- Set up a CI/CD pipeline for database migration by using Terraform
- Set up Multi-AZ infrastructure for a SQL Server Always On FCI by using Amazon FSx

N/A

- Create an approval process for firewall requests during a rehost migration to AWS
- Encrypt an existing Amazon RDS for PostgreSQL DB instance
- Estimate storage costs for an Amazon DynamoDB table
- Implement cross-Region disaster recovery with AWS DMS and Amazon Aurora

Open source

- Automatically generate a PynamoDB model and CRUD functions for Amazon DynamoDB by using a Python application
- Connect by using an SSH tunnel in pgAdmin
- Create application users and roles in Aurora PostgreSQL-Compatible
- Enable encrypted connections for PostgreSQL DB instances in Amazon RDS
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation
- Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools
- Migrate an on-premises MySQL database to Amazon EC2
- Migrate an on-premises MySQL database to Amazon RDS for MySQL
- Migrate an on-premises MySQL database to Aurora MySQL
- Migrate an on-premises PostgreSQL database to Aurora PostgreSQL
- Migrate a Couchbase Server database to Amazon EC2
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling
- Migrate from Oracle 8i or 9i to Amazon RDS for Oracle using SharePlex and AWS DMS
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk
- Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical
- Migrate on-premises Java applications to AWS using AWS App2Container
- Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3
- Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible
- Migrate Oracle functions and procedures that have more than 100 arguments to PostgreSQL
- Migrate Redis workloads to Redis Enterprise Cloud on AWS
- Monitor Amazon Aurora for instances without encryption
- Restart the AWS Replication Agent automatically without disabling SELinux after rebooting a RHEL source server
- Schedule jobs for Amazon RDS for PostgreSQL and Aurora PostgreSQL by using Lambda and Secrets Manager
- Set up data replication between Amazon RDS for MySQL and MySQL on Amazon EC2 using GTID

- Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport
- Unload data from an Amazon Redshift cluster across accounts to Amazon S3

Oracle

- Add HA to Oracle PeopleSoft on Amazon RDS Custom by using a read replica
- Convert JSON Oracle queries into PostgreSQL database SQL
- Convert VARCHAR2(1) data type for Oracle to Boolean data type for Amazon Aurora PostgreSQL
- Emulate Oracle DR by using a PostgreSQL-compatible Aurora global database
- Emulate Oracle PL/SQL associative arrays in Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL
- Estimate the Amazon RDS engine size for an Oracle database by using AWR reports
- Handle anonymous blocks in Dynamic SQL statements in Aurora PostgreSQL
- Handle overloaded Oracle functions in Aurora PostgreSQL-Compatible
- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT
- Load BLOB files into TEXT by using file encoding in Aurora PostgreSQL-Compatible
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL in SSL mode by using AWS DMS
- Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication
- Migrate an Amazon RDS for Oracle DB instance to another VPC
- Migrate an on-premises Oracle database to Amazon EC2 by using Oracle Data Pump
- Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash
- Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate an on-premises Oracle database to Amazon RDS for Oracle
- Migrate an on-premises Oracle database to Amazon RDS for Oracle by using direct Oracle Data Pump Import over a database link
- Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump
- Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL by using an Oracle bystander and AWS DMS
- Migrate an on-premises Oracle database to Oracle on Amazon EC2
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS

- Migrate an Oracle database to Amazon DynamoDB using AWS DMS
- Migrate an Oracle database to Amazon RDS for Oracle by using Oracle GoldenGate flat file adapters
- Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT
- Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT
- Migrate an Oracle JD Edwards EnterpriseOne database to AWS by using Oracle Data Pump and AWS DMS
- Migrate an Oracle partitioned table to PostgreSQL by using AWS DMS
- Migrate an Oracle PeopleSoft database to AWS by using AWS DMS
- Migrate data from an on-premises Oracle database to Aurora PostgreSQL
- Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS
- Migrate from Oracle Database to Amazon RDS for PostgreSQL by using Oracle GoldenGate
- Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate from Oracle to Amazon DocumentDB using AWS DMS
- Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS
- Migrate function-based indexes from Oracle to PostgreSQL
- Migrate legacy applications from Oracle Pro*C to ECPG
- Migrate Oracle CLOB values to individual rows in PostgreSQL on AWS
- Migrate Oracle Database error codes to an Amazon Aurora PostgreSQL-Compatible database
- Migrate Oracle native functions to PostgreSQL using extensions
- Migrate Oracle PeopleSoft to Amazon RDS Custom
- Migrate Oracle ROWID functionality to PostgreSQL on AWS
- Migrate Oracle SERIALLY_REUSABLE pragma packages into PostgreSQL
- Migrate virtual generated columns from Oracle to PostgreSQL
- Monitor Oracle GoldenGate logs by using Amazon CloudWatch
- Replatform Oracle Database Enterprise Edition to Standard Edition 2 on Amazon RDS for Oracle
- Set up an HA/DR architecture for Oracle E-Business Suite on Amazon RDS Custom with an active standby database

- Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible
- Transition roles for an Oracle PeopleSoft application on Amazon RDS Custom for Oracle
- Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL

SAP

- Automatically back up SAP HANA databases using Systems Manager and EventBridge
- Migrate an on-premises SAP ASE database to Amazon EC2
- Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS
- Migrate SAP ASE on Amazon EC2 to Amazon Aurora PostgreSQL-Compatible using AWS SCT and AWS DMS
- Migrate SAP HANA to AWS using SAP HSR with the same hostname
- Reduce homogeneous SAP migration cutover time by using Application Migration Service
- Set up disaster recovery for SAP on IBM Db2 on AWS

More patterns

- Access an Amazon Neptune database from an Amazon EKS container
- Access, query, and join Amazon DynamoDB tables using Athena
- Aggregate data in Amazon DynamoDB for ML forecasting in Athena
- Allow EC2 instances write access to S3 buckets in AMS accounts
- Analyze and visualize nested JSON data with Amazon Athena and Amazon QuickSight
- Authenticate Microsoft SQL Server on Amazon EC2 using AWS Directory Service
- Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch
- Automatically archive items to Amazon S3 using DynamoDB TTL
- Automatically remediate unencrypted Amazon RDS DB instances and clusters
- Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows
- Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9
- Build COBOL Db2 programs by using AWS Mainframe Modernization and AWS CodeBuild
- Build an enterprise data mesh with Amazon DataZone, AWS CDK, and AWS CloudFormation
- Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition
- Convert and unpack EBCDIC data to ASCII on AWS by using Python
- Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL
- Convert the Teradata RESET WHEN feature to Amazon Redshift SQL
- Convert VARCHAR2(1) data type for Oracle to Boolean data type for Amazon Aurora PostgreSQL
- Create application users and roles in Aurora PostgreSQL-Compatible
- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python
- Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Firehose with AWS CDK
- Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing
- Develop advanced generative AI chat-based assistants by using RAG and ReAct prompting
- Emulate Oracle DR by using a PostgreSQL-compatible Aurora global database
- Enable transparent data encryption in Amazon RDS for SQL Server
- Export a Microsoft SQL Server database to Amazon S3 by using AWS DMS
- Implement SHA1 hashing for PII data when migrating from SQL Server to PostgreSQL

- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT
- Load BLOB files into TEXT by using file encoding in Aurora PostgreSQL-Compatible
- Manage credentials using AWS Secrets Manager
- Migrate a Db2 database from Amazon EC2 to Aurora MySQL-Compatible by using AWS DMS
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS
- Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT
- Migrate a self-hosted MongoDB environment to MongoDB Atlas on AWS
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL in SSL mode by using AWS DMS
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation
- Migrate an Amazon RDS DB instance to another VPC or account
- Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication
- Migrate an Amazon RDS for Oracle DB instance to another VPC
- Migrate an Amazon Redshift cluster to an AWS Region in China
- Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using native backup and restore methods
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents
- Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux
- Migrate an on-premises MySQL database to Amazon EC2
- Migrate an on-premises MySQL database to Amazon RDS for MySQL
- Migrate an on-premises MySQL database to Aurora MySQL

- Migrate an on-premises Oracle database to Amazon EC2 by using Oracle Data Pump
- Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash
- Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate an on-premises Oracle database to Amazon RDS for Oracle
- Migrate an on-premises Oracle database to Amazon RDS for Oracle by using direct Oracle Data Pump Import over a database link
- Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump
- Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL by using an Oracle bystander and AWS DMS
- Migrate an on-premises Oracle database to Oracle on Amazon EC2
- Migrate an on-premises PostgreSQL database to Aurora PostgreSQL
- Migrate an on-premises SAP ASE database to Amazon EC2
- Migrate an on-premises ThoughtSpot Falcon database to Amazon Redshift
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS
- Migrate an Oracle database to Amazon DynamoDB using AWS DMS
- Migrate an Oracle database to Amazon RDS for Oracle by using Oracle GoldenGate flat file adapters
- Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT
- Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT
- Migrate an Oracle JD Edwards EnterpriseOne database to AWS by using Oracle Data Pump and AWS DMS
- Migrate an Oracle partitioned table to PostgreSQL by using AWS DMS
- Migrate an Oracle PeopleSoft database to AWS by using AWS DMS
- Migrate a Couchbase Server database to Amazon EC2
- Migrate data from an on-premises Oracle database to Aurora PostgreSQL
- Migrate data to the AWS Cloud by using Starburst
- Migrate Db2 for LUW to Amazon EC2 by using log shipping to reduce outage time
- Migrate Db2 for LUW to Amazon EC2 with high availability disaster recovery

- Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL
- Migrate from Couchbase Server to Couchbase Capella on AWS
- Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL-Compatible using AWS DMS and AWS SCT
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS
- Migrate from Oracle Database to Amazon RDS for PostgreSQL by using Oracle GoldenGate
- Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate from Oracle to Amazon DocumentDB using AWS DMS
- Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical
- Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS
- Migrate function-based indexes from Oracle to PostgreSQL
- Migrate legacy applications from Oracle Pro*C to ECPG
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using Application Migration Service
- Migrate on-premises Cloudera workloads to Cloudera Data Platform on AWS
- Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3
- Migrate Oracle Business Intelligence 12c to the AWS Cloud from on-premises servers
- Migrate Oracle CLOB values to individual rows in PostgreSQL on AWS
- Migrate Oracle Database error codes to an Amazon Aurora PostgreSQL-Compatible database
- Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible
- Migrate Oracle native functions to PostgreSQL using extensions
- Migrate Oracle PeopleSoft to Amazon RDS Custom
- Migrate Oracle ROWID functionality to PostgreSQL on AWS
- Migrate Oracle SERIALLY_REUSABLE pragma packages into PostgreSQL
- Migrate Redis workloads to Redis Enterprise Cloud on AWS
- Migrate a relational database to MongoDB Atlas on AWS
- Migrate SAP ASE on Amazon EC2 to Amazon Aurora PostgreSQL-Compatible using AWS SCT and AWS DMS

- Migrate virtual generated columns from Oracle to PostgreSQL
- Set up a minimum viable data space to share data between organizations
- Monitor Amazon ElastiCache clusters for at-rest encryption
- Monitor ElastiCache clusters for security groups
- Query Amazon DynamoDB tables with SQL by using Amazon Athena
- Reduce homogeneous SAP migration cutover time by using Application Migration Service
- Rotate database credentials without restarting containers
- Run message-driven workloads at scale by using AWS Fargate
- Secure and streamline user access in a Db2 federation database on AWS by using trusted contexts
- Set up a highly available PeopleSoft architecture on AWS
- Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible
- Stream data from IBM Db2, SAP, Sybase, and other databases to MongoDB Atlas on AWS
- Streamline PostgreSQL deployments on Amazon EKS by using PGO
- Transfer large-scale Db2 z/OS data to Amazon S3 in CSV files
- Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport
- Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL
- Verify that new Amazon Redshift clusters launch in a VPC

Storage & backup

Topics

- Allow EC2 instances write access to S3 buckets in AMS accounts
- Automate data stream ingestion into a Snowflake database by using Snowflake Snowpipe, Amazon S3, Amazon SNS, and Amazon Data Firehose
- Automatically encrypt existing and new Amazon EBS volumes
- Back up Sun SPARC servers in the Stromasys Charon-SSP emulator on the AWS Cloud
- Back up and archive data to Amazon S3 with Veeam Backup & Replication
- Configure Veritas NetBackup for VMware Cloud on AWS
- Copy data from an S3 bucket to another account and Region by using the AWS CLI
- Copy data from an S3 bucket to another account and Region by using S3 Batch Replication
- Migrate data from an on-premises Hadoop environment to Amazon S3 using DistCp with AWS PrivateLink for Amazon S3
- More patterns

Allow EC2 instances write access to S3 buckets in AMS accounts

Mansi Suratwala, Amazon Web Services

Summary

AWS Managed Services (AMS) helps you operate your AWS infrastructure more efficiently and securely. AMS accounts have security guardrails for standardized administration of your AWS resources. One guardrail is that default Amazon Elastic Compute Cloud (Amazon EC2) instance profiles don't allow write access to Amazon Simple Storage Service (Amazon S3) buckets. However, your organization might have multiple S3 buckets and require more control over access by EC2 instances. For example, you might want to store database backups from EC2 instances in an S3 bucket.

This pattern explains how to use requests for change (RFCs) to allow your EC2 instances write access to S3 buckets in your AMS account. An RFC is a request created by you or AMS to make a change in your managed environment and that includes a change type (CT) ID for a particular operation.

Prerequisites and limitations

Prerequisites

- An AMS Advanced account. For more information about this, see AMS operations plans in the AMS documentation.
- Access to the AWS Identity and Access Management (IAM) `customer-mc-user-role` role to submit RFCs.
- AWS Command Line Interface (AWS CLI), installed and configured with the EC2 instances in your AMS account.
- An understanding of how to create and submit RFCs in AMS. For more information about this, see What are AMS change types? in the AMS documentation.
- An understanding of manual and automated change types (CTs). For more information about this, see Automated and manual CTs in the AMS documentation.

Architecture

Technology stack

- AMS
- AWS CLI
- Amazon EC2
- Amazon S3
- IAM

Tools

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Managed Services (AMS) helps you operate your AWS infrastructure more efficiently and securely.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.

Epics

Create an S3 bucket with an RFC

Task	Description	Skills required
Create an S3 bucket by using an automated RFC.	1. Sign in to your AMS account, choose the **Choose change type** page, choose **RFCs**, and then choose **Create RFC**. 2. Submit the **Create S3 Bucket** automated RFC.	AWS systems administrator, AWS developer

Task	Description	Skills required
	Note Make sure that you record the S3 bucket's name.	

Create an IAM instance profile and associate it with the EC2 instances

Task	Description	Skills required
Submit a manual RFC to create an IAM role.	When an AMS account is onboarded, a default IAM instance profile named `customer-mc-ec2-instance-profile` is created and associated with each EC2 instance in your AMS account. However, the instance profile doesn't have write permissions to your S3 buckets. To add the write permissions, submit the **Create IAM Resource** manual RFC to create an IAM role that has the following three policies: `customer_ec2_instance_`, `customer_deny_policy`, and `customer_ec2_s3_integration_policy`.	AWS systems administrator, AWS developer

Task	Description	Skills required
	Important The `customer_ec2_instance_` and `customer_deny_policy` policies already exist in your AMS account. However, you need to create `customer_ec2_s3_integration_policy` by using the following sample policy:	

```
{
  "Version": "2012-10-
17",
   "Statement": [
    {
      "Sid": "",
       "Effect":
 "Allow",
       "Principal": {
         "Service":
 "ec2.amazonaws.com"
      },
       "Action":
 "sts:AssumeRole"
    }
  ]
}

Role Permissions:
{
```

Task	Description	Skills required

	``
	    "Version":
	 "2012-10-17",
	    "Statement": [
	        {
	            "Action": [
	
	 "s3:ListBucket",
	
	 "s3:GetBucketLocat
	ion"
	            ],
	            "Resource
	": "arn:aws:s3:::",
	            "Effect":
	 "Allow"
	        },
	        {
	            "Action": [
	
	 "s3:GetObject",
	
	 "s3:PutObject",
	
	 "s3:ListMultipartU
	ploadParts",
	
	 "s3:AbortMultipart
	Upload"
	            ],
	            "Resource
	": "arn:aws:s3:::/*",
	            "Effect":
	 "Allow"
	        }
	    ]
	}
	``	

Task	Description	Skills required
Submit a manual RFC to replace the IAM instance profile.	Submit a manual RFC to associate the target EC2 instances with the new IAM instance profile.	AWS systems administrator, AWS developer
Test a copy operation to the S3 bucket.	Test a copy operation to the S3 bucket by running the following command in the AWS CLI: `aws s3 cp test.txt s3://<S3 bucket>/test2.txt`	AWS systems administrator, AWS developer

Related resources

- Create an IAM instance profile for your Amazon EC2 instances
- Creating an S3 bucket (using the Amazon S3 console, AWS SDKs, or AWS CLI)

Automate data stream ingestion into a Snowflake database by using Snowflake Snowpipe, Amazon S3, Amazon SNS, and Amazon Data Firehose

Bikash Chandra Rout, Amazon Web Services

Summary

This pattern describes how you can use services on the Amazon Web Services (AWS) Cloud to process a continuous stream of data and load it into a Snowflake database. The pattern uses Amazon Data Firehose to deliver the data to Amazon Simple Storage Service (Amazon S3), Amazon Simple Notification Service (Amazon SNS) to send notifications when new data is received, and Snowflake Snowpipe to load the data into a Snowflake database.

By following this pattern, you can have continuously generated data available for analysis in seconds, avoid multiple manual `COPY` commands, and have full support for semi-structured data on load.

Prerequisites and limitations

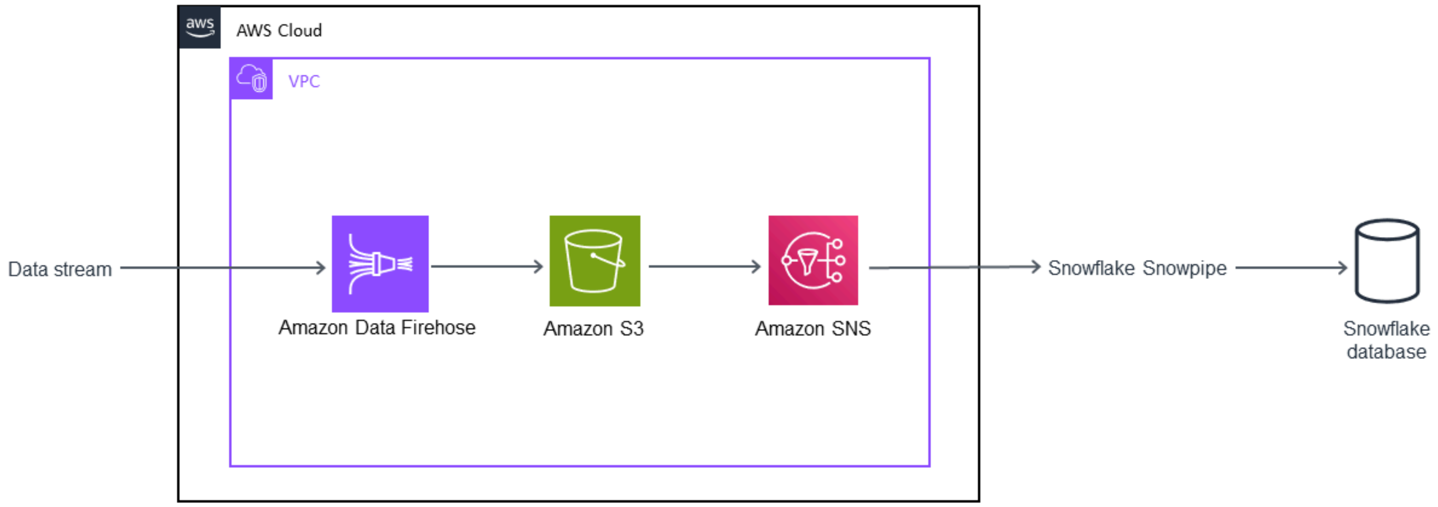
Prerequisites

- An active AWS account.
- A data source that is continuously sending data to a Firehose delivery stream.
- An existing S3 bucket that is receiving the data from the Firehose delivery stream.
- An active Snowflake account.

Limitations

- Snowflake Snowpipe doesn't connect directly to Firehose.

Architecture



Technology stack

- Amazon Data Firehose
- Amazon SNS
- Amazon S3
- Snowflake Snowpipe
- Snowflake database

Tools

- Amazon Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers.
- Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.
- Snowflake – Snowflake is an analytic data warehouse provided as Software-as-a-Service (SaaS).
- Snowflake Snowpipe – Snowpipe loads data from files as soon as they're available in a Snowflake stage.

Epics

Set up a Snowflake Snowpipe

Task	Description	Skills required
Create a CSV file in Snowflake .	Sign in to Snowflake and run the `CREATE FILE FORMAT` command to create a CSV file with a specified field delimiter . For more information about this and other Snowflake commands, see the [Additional information](#) section.	Developer
Create an external Snowflake stage.	Run the `CREATE STAGE` command to create an external Snowflake stage that references the CSV file you created earlier. Important: You will need the URL for the S3 bucket, your AWS access key, and your AWS secret access key. Run the `SHOW STAGES` command to verify that the Snowflake stage is created.	Developer
Create the Snowflake target table.	Run the `CREATE TABLE` command to create the Snowflake table.	Developer
Create a pipe.	Run the `CREATE PIPE` command; make sure that `auto_ingest=true` is in the command. Run the `SHOW PIPES` command to verify that the pipe is created. Copy	Developer

Task	Description	Skills required
	and save the `notification_channel` column value. This value will be used to configure Amazon S3 event notifications.	

Configure the S3 bucket

Task	Description	Skills required
Create a 30-day lifecycle policy for the S3 bucket.	Sign in to the AWS Management Console and open the Amazon S3 console. Choose the S3 bucket that contains the data from Firehose. Then choose the **Management** tab in the S3 bucket and choose **Add lifecycle rule**. Enter a name for your rule in the **Lifecycle rule** dialog box, and configure a 30-day lifecycle rule for your bucket. For help with this and other stories, see the Related resources section.	System Administrator, Developer
Create an IAM policy for the S3 bucket.	Open the AWS Identity and Access Management (IAM) console and choose **Policies**. Choose **Create policy**, and choose the **JSON** tab. Copy and paste the policy from the Additional information section into the JSON field. This policy will grant	System Administrator, Developer

Task	Description	Skills required
	`PutObject` and `DeleteObject` permissions, as well as `GetObject`, `GetObjectVersion`, and `ListBucket` permissions. Choose **Review policy**, enter a policy name, and then choose **Create policy**.	
Assign the policy to an IAM role.	Open the IAM console, choose **Roles**, and then choose **Create role**. Choose **Another AWS account** as the trusted entity. Enter your AWS account ID, and choose **Require external ID**. Enter a placeholder ID that you will change it later. Choose **Next**, and assign the IAM policy you created earlier. Then create the IAM role.	System Administrator, Developer
Copy the Amazon Resource Name (ARN) for the IAM role.	Open the IAM console, and choose **Roles**. Choose the IAM role you created earlier, and then copy and store the **Role ARN**.	System Administrator, Developer

Set up a storage integration in Snowflake

Task	Description	Skills required
Create a storage integration in Snowflake.	Sign in to Snowflake and run the `CREATE STORAGE INTEGRATION` command.	System Administrator, Developer

Task	Description	Skills required
	This will modify the trusted relationship, grant access to Snowflake, and provide the external ID for your Snowflake stage.	
Retrieve the IAM role for your Snowflake account.	Run the `DESC INTEGRATION` command to retrieve the ARN for the IAM role. **Important** `<integration_name>` is the name of the Snowflake storage integration you created earlier.	System Administrator, Developer
Record two column values.	Copy and save the values for the `storage_aws_iam_user_arn` and `storage_aws_external_id` columns.	System Administrator, Developer

Allow Snowflake Snowpipe to access the S3 bucket

Task	Description	Skills required
Modify the IAM role policy.	Open the IAM console and choose **Roles**. Choose the IAM role you created earlier and choose the **Trust relationships** tab. Choose **Edit trust relationship**. Replace	System Administrator, Developer

Task	Description	Skills required
	`snowflake_external_id` with the `storage_aws_external_id` value you copied earlier. Replace `snowflake_user_arn` with the `storage_aws_iam_user_arn` value you copied earlier. Then choose **Update trust policy**.	

Turn on and configure SNS notifications for the S3 bucket

Task	Description	Skills required
Turn on event notifications for the S3 bucket.	Open the Amazon S3 console and choose your bucket. Choose **Properties**, and under **Advanced setting**s, choose **Events**. Choose **Add notification**, and enter a name for this event. If you don't enter a name, a globally unique identifier (GUID) will be used.	System Administrator, Developer
Configure Amazon SNS notifications for the S3 bucket.	Under **Events**, choose **ObjectCreate (All)**, and then choose **SQS Queue** in the **Send to** dropdown list. In the **SNS** list, choose **Add SQS queue ARN**, and paste the `notification_channel` value you copied earlier. Then choose **Save**.	System Administrator, Developer

Task	Description	Skills required
Subscribe the Snowflake SQS queue to the SNS topic.	Subscribe the Snowflake SQS queue to the SNS topic you created. For help with this step, see the Related resources section.	System Administrator, Developer

Check the Snowflake stage integration

Task	Description	Skills required
Check and test Snowpipe.	Sign in to Snowflake and open the Snowflake stage. Drop files into your S3 bucket and check if the Snowflake table loads them. Amazon S3 will send SNS notifications to Snowpipe when new objects appear in the S3 bucket.	System Administrator, Developer

Related resources

- Managing your storage lifecycle
- Subscribe the Snowflake SQS Queue to the Amazon SNS Topic

Additional information

Create a file format:

```
CREATE FILE FORMAT <name>
TYPE = 'CSV'
FIELD_DELIMITER = '|'
SKIP_HEADER = 1;
```

Create an external stage:

```
externalStageParams (for Amazon S3) ::=
  URL = 's3://[//]'

  [ { STORAGE_INTEGRATION =  } | { CREDENTIALS = ( {  { AWS_KEY_ID = `` AWS_SECRET_KEY
 = `` [ AWS_TOKEN = `` ] } | AWS_ROLE = ``  } ) ) }` ]
  [ ENCRYPTION = ( [ TYPE = 'AWS_CSE' ] [ MASTER_KEY = '' ] |
                   [ TYPE = 'AWS_SSE_S3' ] |
                   [ TYPE = 'AWS_SSE_KMS' [ KMS_KEY_ID = '' ] |
                   [ TYPE = NONE ] )
```

Create a table:

```
CREATE [ OR REPLACE ] [ { [ LOCAL | GLOBAL ] TEMP[ORARY] | VOLATILE } | TRANSIENT ]
 TABLE [ IF NOT EXISTS ]
  <table_name>
    ( <col_name> <col_type> [ { DEFAULT <expr>
                               | { AUTOINCREMENT | IDENTITY } [ ( <start_num> ,
 <step_num> ) | START <num> INCREMENT <num> ] } ]
                                /* AUTOINCREMENT / IDENTITY supported only for numeric
 data types (NUMBER, INT, etc.) */
                            [ inlineConstraint ]
      [ , <col_name> <col_type> ... ]
      [ , outoflineConstraint ]
      [ , ... ] )
  [ CLUSTER BY ( <expr> [ , <expr> , ... ] ) ]
  [ STAGE_FILE_FORMAT = ( { FORMAT_NAME = '<file_format_name>'
                           | TYPE = { CSV | JSON | AVRO | ORC | PARQUET | XML }
 [ formatTypeOptions ] } ) ]
  [ STAGE_COPY_OPTIONS = ( copyOptions ) ]
  [ DATA_RETENTION_TIME_IN_DAYS = <num> ]
  [ COPY GRANTS ]
  [ COMMENT = '<string_literal>' ]
```

Show stages:

```
SHOW STAGES;
```

Create a pipe:

```
CREATE [ OR REPLACE ] PIPE [ IF NOT EXISTS ]
  [ AUTO_INGEST = [ TRUE | FALSE ] ]
  [ AWS_SNS_TOPIC =  ]
```

```
  [ INTEGRATION = '' ]
  [ COMMENT = '' ]
  AS
```

Show pipes:

```
SHOW PIPES [ LIKE '<pattern>' ]
           [ IN { ACCOUNT | [ DATABASE ] <db_name> | [ SCHEMA ] <schema_name> } ]
```

Create a storage integration:

```
CREATE STORAGE INTEGRATION <integration_name>
  TYPE = EXTERNAL_STAGE
  STORAGE_PROVIDER = S3
  ENABLED = TRUE
  STORAGE_AWS_ROLE_ARN = '<iam_role>'
  STORAGE_ALLOWED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/')
  [ STORAGE_BLOCKED_LOCATIONS = ('s3://<bucket>/<path>/', 's3://<bucket>/<path>/') ]
```

Example:

```
create storage integration s3_int
  type = external_stage
  storage_provider = s3
  enabled = true
  storage_aws_role_arn = 'arn:aws:iam::001234567890:role/myrole'
  storage_allowed_locations = ('s3://amzn-s3-demo-bucket1/mypath1/', 's3://amzn-s3-
demo-bucket2/mypath2/')
  storage_blocked_locations = ('s3://amzn-s3-demo-bucket1/mypath1/sensitivedata/',
 's3://amzn-s3-demo-bucket2/mypath2/sensitivedata/');
```

For more information about this step, see Configuring a Snowflake storage integration to access Amazon S3 from the Snowflake documentation.

Describe an integration:

```
DESC INTEGRATION <integration_name>;
```

S3 bucket policy:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "s3:PutObject",
              "s3:GetObject",
              "s3:GetObjectVersion",
              "s3:DeleteObject",
              "s3:DeleteObjectVersion"
            ],
            "Resource": "arn:aws:s3:::/*"
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "/*"
                    ]
                }
            }
        }
    ]
}
```

Automatically encrypt existing and new Amazon EBS volumes

Tony DeMarco and Josh Joy, Amazon Web Services

Summary

Encryption of Amazon Elastic Block Store (Amazon EBS) volumes is important to an organization's data protection strategy. It is an important step in establishing a well-architected environment. Although there is no direct way to encrypt existing unencrypted EBS volumes or snapshots, you can encrypt them by creating a new volume or snapshot. For more information, see Encrypt EBS resources in the Amazon EC2 documentation. This pattern provides preventative and detective controls for encrypting your EBS volumes, both new and existing. In this pattern, you configure account settings, create automated remediation processes, and implement access controls.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- AWS Command Line Interface (AWS CLI), installed and configured on macOS, Linux, or Windows
- jq, installed and configured on macOS, Linux, or Windows
- AWS Identity and Access Management (IAM) permissions are provisioned to have read and write access to AWS CloudFormation, Amazon Elastic Compute Cloud (Amazon EC2), AWS Systems Manager, AWS Config, and AWS Key Management Service (AWS KMS)
- AWS Organizations is configured with all features enabled, a requirement for service control policies
- AWS Config is enabled in the target accounts

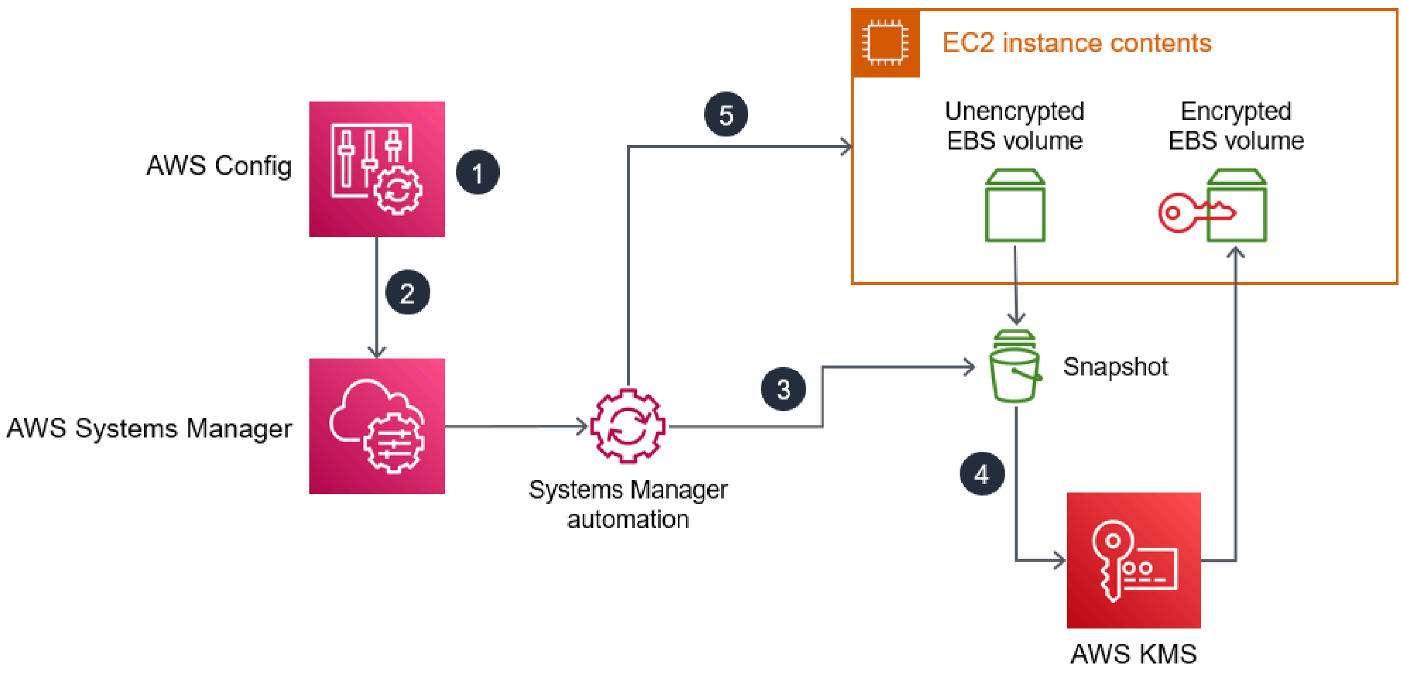
Limitations

- In your target AWS account, there must be no AWS Config rules named **encrypted-volumes**. This solution deploys a rule with this name. Preexisting rules with this name can cause the deployment to fail and result in unnecessary charges related to processing the same rule more than once.
- This solution encrypts all EBS volumes with the same AWS KMS key.

- If you enable encryption of EBS volumes for the account, this setting is Region-specific. If you enable it for an AWS Region, you cannot disable it for individual volumes or snapshots in that Region. For more information, see Encryption by default in the Amazon EC2 documentation.
- When you remediate existing, unencrypted EBS volumes, ensure that the EC2 instance is not in use. This automation shuts down the instance in order to detach the unencrypted volume and attach the encrypted one. There is downtime while the remediation is in progress. If this is a critical piece of infrastructure for your organization, make sure that manual or automatic high-availability configurations are in place so as to not impact the availability of any applications running on the instance. We recommend that you remediate critical resources only during standard maintenance windows.

Architecture

Automation workflow



1. AWS Config detects an unencrypted EBS volume.
2. An administrator uses AWS Config to send a remediation command to Systems Manager.
3. The Systems Manager automation takes a snapshot of the unencrypted EBS volume.
4. The Systems Manager automation uses AWS KMS to create an encrypted copy of the snapshot.
5. The Systems Manager automation does the following:

a. Stops the affected EC2 instance if it is running

b. Attaches the new, encrypted copy of the volume to the EC2 instance

c. Returns the EC2 instance to its original state

Tools

AWS services

- AWS CLI – The AWS Command Line Interface (AWS CLI) provides direct access to the public application programming interfaces (APIs) of AWS services. You can explore a service's capabilities with the AWS CLI and develop shell scripts to manage your resources. In addition to the low-level API-equivalent commands, several AWS services provide customizations for the AWS CLI. Customizations can include higher-level commands that simplify using a service with a complex API.
- AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your AWS resources. You create a template that describes all the AWS resources that you want (such as Amazon EC2 instances), and CloudFormation provisions and configures those resources for you.
- AWS Config – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time.
- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.
- AWS KMS – AWS Key Management Service (AWS KMS) is an encryption and key management service scaled for the cloud. AWS KMS keys and functionality are used by other AWS services, and you can use them to protect data in your AWS environment.
- AWS Organizations – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Systems Manager Automation – Systems Manager Automation simplifies common maintenance and deployment tasks for Amazon EC2 instances and other AWS resources.

Other services

- jq – jq is a lightweight and flexible command-line JSON processor. You use this tool to extract key information from the AWS CLI output.

Code

- The code for this pattern is available in the GitHub Automatically remediate unencrypted EBS Volumes using customer KMS keys repository.

Epics

Automate remediation of unencrypted volumes

Task	Description	Skills required
Download scripts and CloudFormation templates.	Download the shell script, JSON file, and CloudFormation templates from the GitHub Automatically remediate unencrypted EBS Volumes using customer KMS keys repository.	AWS administrator, General AWS
Identify the administrator for the AWS KMS key.	1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/. 2. Identify a user or role who will be the AWS KMS key administrator. If a new user or role needs to be created for this purpose, create it now. For more information, see IAM Identities in the IAM documentation. This automation creates a new AWS KMS key.	AWS administrator, General AWS

Task	Description	Skills required
	3. Once identified, copy the user's or role's Amazon Resource Name (ARN). For more information, see IAM ARNs in the IAM documentation. You use this ARN in the next step.	

Task	Description	Skills required
Deploy the Stack1 CloudFormation template.	1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/. 2. In CloudFormation, deploy the `Stack1.yaml` template. Note the following deployment details: • Give the stack a clear and descriptive name. Note the stack name because you need this value in the next step. • Paste the ARN of the key administrator into the only parameter field in Stack1. This user or role becomes the administrator for the AWS KMS key created by the stack. For more information about deploying a CloudFormation template, see [Working with AWS CloudFormation templates](#) in the CloudFormation documentation.	AWS administrator, General AWS

Task	Description	Skills required
Deploy the Stack2 CloudFormation template.	In CloudFormation, deploy the `Stack2.yaml` template. Note the following deployment details: • Give the stack a clear and descriptive name. • For the only parameter of Stack2, enter the name of the stack you created in the previous step. This allows Stack2 to reference the new AWS KMS key and role deployed by the stack in the previous step.	AWS administrator, General AWS
Create an unencrypted volume for testing.	Create an EC2 instance with an unencrypted EBS volume. For instructions, see [Create an Amazon EBS volume](#) in the Amazon EC2 documentation. The instance type does not matter, and access to the instance is not needed. You can create a t2.micro instance to stay in the free tier, and you don't need to create a key pair.	AWS administrator, General AWS

Task	Description	Skills required
Test the AWS Config rule.	1. Open the AWS Config console at https://console.aws.amazon.com/config/. On the **Rules** page, choose the **encrypted-volumes** rule. 2. Confirm that your new, unencrypted test instance appears in the list of non-compliant resources. If the volume does not appear immediately, wait a few minutes and refresh the results. The AWS Config rule detects the resource changes shortly after the instance and volume are created. 3. Select the resource, and then then choose **Remediate**. You can view the remediati on progress and status in Systems Manager as follows: 1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/. 2. In the navigation pane, choose **Automation**.	AWS administrator, General AWS

Task	Description	Skills required
	3. Choose the **Execution ID** link to view the steps and the status.	
Configure additional accounts or AWS Regions.	As needed for your use case, repeat this epic for any additional accounts or AWS Regions.	AWS administrator, General AWS

Enable account-level encryption of EBS volumes

Task	Description	Skills required
Run the enable script.	1. In a bash shell, use the `cd` command to navigate into the cloned repository. 2. Enter the following command to run the `enable-ebs-encryption-for-account` script. `./Bash/enable-ebs-encryption-for-account.sh`	AWS administrator, General AWS, bash
Confirm the settings are updated.	1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. 2. On the right side of the screen, under **Settings**,	AWS administrator, General AWS

Task	Description	Skills required
	choose **Data protection and security**. 3. Under the **EBS encryption** section, confirm that **Always encrypt new EBS volumes** is turned on and that the **Default encryption key** is set to the ARN that you specified previously. **Note** If the **Always encrypt new EBS volumes** setting is turned off or the key is still set to **alias/aws/ebs**, confirm that you are logged into the same account and AWS Region where you ran the shell script, and check your shell for error messages.	
Configure additional accounts or AWS Regions.	As needed for your use case, repeat this epic for any additional accounts or AWS Regions.	AWS administrator, General AWS

Prevent creation of unencrypted instances

Task	Description	Skills required
Create a service control policy.	1. Open the AWS Organizations console at https://console.aws.amazon.com/organizations/v2/. 2. Create a new service control policy. For more information, see [Creating a service control policy](#) in the AWS Organizations documentation. 3. Add the contents of `DenyUnencryptedEC2.json` to the policy and save it. You downloaded this JSON file from the GitHub repository in the first epic. 4. Attach this policy to the organization root or any necessary organizational units (OUs). For more information, see [Attaching and detaching service control policies](#) in the AWS Organizations documentation.	AWS administrator, General AWS

Related resources

AWS service documentation

- [AWS CLI](#)

- AWS Config
- AWS CloudFormation
- Amazon EC2
- AWS KMS
- AWS Organizations
- AWS Systems Manager Automation

Other resources

- jq manual (jq website)
- jq download (GitHub)

Back up Sun SPARC servers in the Stromasys Charon-SSP emulator on the AWS Cloud

Kevin Yung and Rohit Darji, Amazon Web Services

Luis Ramos, Stromasys

Summary

This pattern provides four options for backing up your Sun Microsystems SPARC servers after a migration from an on-premises environment to the Amazon Web Services (AWS) Cloud. These backup options help you to implement a backup plan that meets your organization's recovery point objective (RPO) and recovery time objective (RTO), uses automated approaches, and lowers your overall operational costs. The pattern provides an overview of the four backup options and steps to implement them.

If you use a Sun SPARC server hosted as a guest on a Stromasys Charon-SSP emulator, you can use one of the following three backup options:

- **Backup option 1: Stromasys virtual tape –** Use the Charon-SSP virtual tape feature to set up a backup facility in the Sun SPARC server and archive your backup files to Amazon Simple Storage Service (Amazon S3) and Amazon Simple Storage Service Glacier using AWS Systems Manager Automation.
- **Backup option 2: Stromasys snapshot –** Use the Charon-SSP snapshot feature to set up a backup facility for the Sun SPARC guest servers in Charon-SSP.
- **Backup option 3**: **Amazon Elastic Block Store (Amazon EBS) volume snapshot –** If you host the Charon-SSP emulator on Amazon Elastic Compute Cloud (Amazon EC2), you can use an Amazon EBS volume snapshot to create backups for a Sun SPARC file system.

If you use a Sun SPARC server hosted as a guest on hardware and Charon-SSP on Amazon EC2, you can use the following backup option:

- **Backup option 4: AWS Storage Gateway virtual tape library (VTL) –** Use a backup application with a Storage Gateway VTL Tape Gateway to back up the Sun SPARC servers.

If you use a Sun SPARC server hosted as a branded zone in a Sun SPARC server, you can use backup options 1, 2, and 4.

Stromasys provides software and services to emulate legacy SPARC, Alpha, VAX, and PA-RISC critical systems. For more information about migrating to the AWS Cloud using Stromasys emulation, see Rehosting SPARC, Alpha, or other legacy systems to AWS with Stromasys on the AWS Blog.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Existing Sun SPARC servers.
- Existing licenses for Charon-SSP. Licenses for Charon-SSP are available from AWS Marketplace and licenses for Stromasys Virtual Environment (VE) are available from Stromasys. For more information, contact Stromasys sales.
- Familiarity with Sun SPARC servers and Linux backups.
- Familiarity with Charon-SSP emulation technology. For more information about this, see Stromasys legacy server emulation in the Stromasys documentation.
- If you want to use the virtual tape facility or backup applications for your Sun SPARC servers file systems, you must create and configure the backup facilities for the Sun SPARC server file system.
- An understanding of RPO and RTO. For more information about this, see Disaster recovery objectives from the Reliability Pillar whitepaper in the AWS Well-Architected Framework documentation.
- To use **Backup option 4**, you must have the following:
  - A software-based backup application that supports a Storage Gateway VTL Tape Gateway. For more information about this, see Working with VTL devices in the AWS Storage Gateway documentation.
  - Bacula Director or a similar backup application, installed and configured. For more information about this, see the Bacula Director documentation.

The following table provides information about the four backup options in this pattern.

Backup options	Achieves crash consistency?	Achieves application consistency?	Virtual backup appliance solution?	Typical use cases

Option 1 – Stromasys virtual tape	**Yes** You can automate Sun SPARC file system snapshots to back up data in a virtual tape. For example, you can use UFS or ZFS snapshots.	**Yes** This backup option requires an automated script to flush in-flight transactions, configure a read-only or temporary offline mode during the file system snapshot, or take an application data dump. You might also require application downtime or read-only mode.	**Yes**	Sun SPARC server file systems backup with .tar or .zip files Application data backup

Option 2 – Stromasys snapshot	**Yes** You must configure [Charon-SSP Manager]() or use a command-line startup argument to enable this feature. You must also run a Linux command to ask the Charon-SSP emulator to save the Sun SPARC guest server state into a snapshot file. **Important** You must shut down the Sun SPARC guest server.	**Yes** This backup option creates a snapshot of the emulated guest server, including its virtual disks and memory dump. **Important** You must shut down the Sun SPARC guest server during the snapshot.	**No**	Sun SPARC server snapshot Application data backup

Option 3 – Amazon EBS volume snapshot	**Yes** You can use AWS Backup to automate the Amazon EBS snapshot.	**Yes** This backup option requires an automated script to flush in-flight transactions and configure a read-only or temporary stop of the EC2 instance during the Amazon EBS volume snapshot. **Important** This backup option might require application downtime or read-only mode to achieve application consistency.	**No**	Sun SPARC server file systems snapshot Application data backup

Option 4 – AWS Storage Gateway VTL	**Yes** You can automatically back up Sun SPARC file system backup data to the VTL by using a backup agent.	**Yes** This backup option requires an automated script to flush in-flight transactions and configure a read-only or temporary offline mode during the file system snapshot or application data dump. **Important** This backup option might require application downtime or read-only mode.	**Yes**	A large fleet of Sun SPARC server file system backups Application data backup

Limitations

- You can use this pattern's approaches to back up individual Sun SPARC servers, but you can also use these backup options for shared data if you have applications that run in a cluster.

Tools

Backup option 1: Stromasys virtual tape

- [Stromasys Charon-SSP emulator]() – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including operating systems (OSs) such as SunOS or Solaris, their layered products, and applications.
- [Amazon EC2]() – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.
- [Amazon EFS]() – Amazon Elastic File System (Amazon EFS) provides a simple, serverless, set-and-forget elastic file system for use with AWS Cloud services and on-premises resources.
- [Amazon S3]() – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- [Amazon S3 Glacier]() – Amazon Simple Storage Service Glacier is a secure, durable, and extremely low-cost Amazon S3 storage class for data archiving and long-term backup.
- [AWS Systems Manager Automation]() – Automation, a capability of AWS Systems Manager, simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources.

Backup option 2: Stromasys snapshot

- [Stromasys Charon-SSP emulator]() – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including OSs such as SunOS or Solaris, their layered products, and applications.
- [Amazon EC2]() – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.
- [Amazon EFS]() – Amazon Elastic File System (Amazon EFS) provides a simple, serverless, set-and-forget elastic file system for use with AWS Cloud services and on-premises resources.
- [Amazon S3]() – Amazon Simple Storage Service (Amazon S3) is storage for the internet.

- Amazon S3 Glacier – Amazon Simple Storage Service Glacier is a secure, durable, and extremely low-cost Amazon S3 storage class for data archiving and long-term backup.
- AWS Systems Manager Automation – Automation, a capability of AWS Systems Manager, simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources.

Backup option 3: Amazon EBS volume snapshot

- Stromasys Charon-SSP emulator – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including OSs such as SunOS or Solaris, their layered products, and applications.
- AWS Backup – AWS Backup is a fully-managed data protection service that makes it easy to centralize and automate across AWS services, in the cloud, and on premises.
- Amazon EBS – Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances.
- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.

Backup option 4: AWS Storage Gateway VTL

- Stromasys Charon-SSP emulator – Charon-SSP emulator creates the virtual replica of the original SPARC hardware inside a standard 64-bit x86 compatible computer system. It runs the original SPARC binary code, including OSs such as SunOS or Solaris, their layered products, and applications.
- Bacula – Bacula is an open-source, enterprise-level computer backup system. For more information about whether your existing backup application supports Tape Gateway, see Supported third-party backup applications for a Tape Gateway in the AWS Storage Gateway documentation.
- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity that you use to build and host your software systems.

- Amazon RDS for MySQL – Amazon Relational Database Service (Amazon RDS) supports DB instances running several versions of MySQL.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet.
- Amazon S3 Glacier – Amazon Simple Storage Service Glacier is a secure, durable, and extremely low-cost Amazon S3 storage class for data archiving and long-term backup.
- AWS Storage Gateway – Storage Gateway connects an on-premises software appliance with cloud-based storage to provide seamless integration with data security features between your on-premises IT environment and the AWS storage infrastructure.

Epics

Backup option 1 – Create a Stromasys virtual tape backup

Task	Description	Skills required
Create an Amazon EFS shared file system for virtual tape file storage.	Sign in to the AWS Management Console or use AWS CLI to create an Amazon EFS file system. For more information about this, see Create an Amazon EFS file system in the Amazon EFS documentation.	Cloud architect
Configure the Linux host to mount the shared file system.	Install the Amazon EFS driver on the Amazon EC2 Linux instance and configure the Linux OS to mount the Amazon EFS shared file system during startup. For more information about this, see Mounting file systems using the EFS mount helper in the Amazon EFS documentation.	DevOps engineer

Task	Description	Skills required
Install the Charon-SSP emulator.	Install the Charon-SSP emulator on the Amazon EC2 Linux instance. For more information about this, see [Setting up an AWS Cloud instance for Charon-SSP]() in the Stromasys documentation.	DevOps engineer
Create a virtual tape file container in the shared file system for each Sun SPARC guest server.	Run the `touch <vtape-container-name>` command to create a virtual tape file container in the shared file system for each Sun SPARC guest server deployed in the Charon-SSP emulator.	DevOps engineer
Configure Charon-SSP Manager to create virtual tape devices for the Sun SPARC guest servers.	Log in to Charon-SSP Manager, create virtual tape devices, and configure them to use the virtual tape container files for each Sun SPARC guest server. For more information about this, see the [Charon-SSP 5.2 for Linux user guide]() in the Stromasys documentation.	DevOps engineer

Task	Description	Skills required
Validate that the virtual tape device is available in the Sun SPARC guest servers.	Log in to each Sun SPARC guest server and run the `mt -f /dev/rmt/1` command to validate that the virtual tape device is configured in the OS.	DevOps engineer
Develop the Systems Manager Automation runbook and automation.	Develop the Systems Manager Automation runbook and set up maintenance windows and associations in Systems Manager for scheduling the backup process. For more information about this, see [Automation walkthroughs]() and [Setting up maintenance windows]() in the AWS Systems Manager documentation.	Cloud architect
Configure Systems Manager Automation to archive rotated virtual tape container files.	Use the code sample from **Back option 1** in the *Additional information* section to develop a Systems Manager Automation runbook to archive rotated virtual tape container files to Amazon S3 and Amazon S3 Glacier.	Cloud architect

Task	Description	Skills required
Deploy the Systems Manager Automation runbook for archiving and scheduling.	Deploy the Systems Manager Automation runbook and schedule it to automatically run in Systems Manager. For more information about this, see Automation walkthroughs in the Systems Manager documentation.	Cloud architect

Backup option 2 – Create a Stromasys snapshot

Task	Description	Skills required
Create an Amazon EFS shared file system for virtual tape file storage.	Sign in to the AWS Management Console or use AWS CLI to create an Amazon EFS file system. For more information about this, see Create your Amazon EFS file system in the Amazon EFS documentation.	Cloud architect
Configure the Linux host to mount the shared file system.	Install the Amazon EFS driver in the Amazon EC2 Linux instance and configure the Linux OS to mount the Amazon EFS shared file system during startup. For more information about this, see Mounting file systems using the EFS mount	DevOps engineer

Task	Description	Skills required
	helper in the Amazon EFS documentation.	
Install the Charon-SSP emulator.	Install the Charon-SSP emulator on the Amazon EC2 Linux instance. For more information about this, see Setting up an AWS Cloud instance for Charon-SSP in the Stromasys documentation.	DevOps engineer
Configure the Sun SPARC guest servers to start up with the snapshot option.	Use Charon-SSP Manager to set up the snapshot option for each Sun SPARC guest servers. For more information about this, see the Charon-SSP 5.2 for Linux user guide in the Stromasys documentation.	DevOps engineer
Develop the Systems Manager Automation runbook.	Use the code sample from **Backup option 2** in the *Additional information* section to develop a Systems Manager Automation runbook to remotely run the snapshot command on a Sun SPARC guest server during a maintenance window.	Cloud architect

Task	Description	Skills required
Deploy the Systems Manager Automation runbook and set up the association to the Amazon EC2 Linux hosts.	Deploy the Systems Manager Automation runbook and set up maintenance windows and associations in Systems Manager for scheduling the backup process. For more information about this, see [Automation walkthroughs]() and [Setting up Maintenance Windows]() in the AWS Systems Manager documentation.	Cloud architect
Archive snapshots into long-term storage.	Use the runbook sample code from the *Additional informati on* section to develop a Systems Manager Automation runbook to archive snapshot files to Amazon S3 and Amazon S3 Glacier.	Cloud architect

Backup option 3 – Create an Amazon EBS volume snapshot

Task	Description	Skills required
Install the Charon-SSP emulator.	Install the Charon-SSP emulator on the Amazon EC2 Linux instance. For more information about this, see [Setting up an AWS Cloud instance for Charon-SSP]() in the Stromasys documentation.	DevOps engineer

Task	Description	Skills required
Create EBS volumes for the Sun SPRAC guest servers.	Sign in to the AWS Management Console, open the Amazon EBS console, and then create EBS volumes for the Sun SPRAC guest servers. For more information about this, see [Setting up an AWS Cloud instance for Charon-SSP]() in the Stromasys documentation.	Cloud architect
Attach the EBS volumes to the Amazon EC2 Linux instance.	On the Amazon EC2 console, attach the EBS volumes to the Amazon EC2 Linux instance. For more information about this, see [Attach an Amazon EBS volume to an instance]() in the Amazon EC2 documenta tion.	AWS DevOps
Map EBS volumes as SCSI drives in the Charon-SSP emulator.	Configure Charon-SSP Manager to map the EBS volumes as SCSI drives in the Sun SPARC guest servers. For more information about this, see the *SCSI storage configuration* section of the [Charon-SSP V5.2 for Linux]() guide in the Stromasys documentation.	AWS DevOps

Task	Description	Skills required
Configure the AWS Backup schedule for snapshotting the EBS volumes.	Set up AWS Backup policy and schedules to snapshot the EBS volumes. For more information about this, see the Amazon EBS backup and restore using AWS Backup tutorial in the AWS Developer Center documentation.	AWS DevOps

Backup option 4 – Create an AWS Storage Gateway VTL

Task	Description	Skills required
Create a Tape Gateway device.	Sign in to the AWS Management Console, open the AWS Storage Gateway console, and then create a Tape Gateway device in a VPC. For more information about this, see Creating a gateway in the AWS Storage Gateway documentation.	Cloud architect
Create an Amazon RDS DB instance for the Bacula Catalog.	Open the Amazon RDS console and create an Amazon RDS for MySQL DB instance. For more information about this, see Creating a MySQL DB instance and connecting to	Cloud architect

Task	Description	Skills required
	a database on a MySQL DB instance in the Amazon RDS documentation.	
Deploy the backup application controller in the VPC.	Install Bacula on the EC2 instance, deploy the backup application controller, and then configure the backup storage to connect with the Tape Gateway device. You can use the sample Bacula Director storage daemon configuration in the `Bacula-storage-daemon-config.txt` file (attached). For more information about this, see the Bacula documentation.	AWS DevOps
Set up backup application on the Sun SPARC guest servers.	Set up a second client to install and set up the backup application on the Sun SPARC guest servers by using the sample Bacula configuration in the `SUN-SPARC-Guest-Bacula-Config.txt` file (attached).	DevOps engineer

Task	Description	Skills required
Set up the backup configuration and schedule.	Set up backup configuration and schedules in the backup application controller by using the sample Bacula Director configuration in the `Bacula-Directory-Config.txt` file (attached). For more information about this, see the [Bacula documentation]().	DevOps engineer
Validate that the backup configuration and schedules are correct.	Follow the instruction from the [Bacula documentation]() to perform the validation and backup testing for your setup in the Sun SPARC guest servers. For example, you can use the following commands to validate the configuration files: • `bacula-dir -t -c bacula-dir.conf` • `bacula-fd -t -c bacula-fd.conf` • `bacula-sd -t -c bacula-sd.conf`	DevOps engineer

Related resources

- [Charon virtual SPARC with VE licensing]()
- [Charon virtual SPARC]()

- Using cloud services and object storage with Bacula Enterprise Edition
- Disaster recovery (DR) objectives
- Charon legacy system emulation solutions

Additional information

Backup option 1 – Create a Stromasys virtual tape

You can use the following sample Systems Manager Automation runbook code to automatically start the backup and then swap the tapes:

```
...
# example backup script saved in SUN SPARC Server
 #!/usr/bin/bash
 mt -f  rewind
 tar -cvf
 mt -f  offline
...
         mainSteps:
         - action: aws:runShellScript
           name:
           inputs:
             onFailure: Abort
             timeoutSeconds: "1200"
             runCommand:
             - |
               # Validate tape backup container file exists
               if [ ! -f {{TapeBackupContainerFile}} ]; then
                 logger -s -p local3.warning "Tape backup container file is not exists
 - {{TapeBackupContainerFile}}, create a new one"
                 touch {{TapeBackupContainerFile}}
               fi
         - action: aws:runShellScript
           name: startBackup
           inputs:
             onFailure: Abort
             timeoutSeconds: "1200"
             runCommand:
             - |
               user={{BACKUP_USER}}
               keypair={{KEYPAIR_PATH}}
               server={{SUN_SPARC_IP}}
```

```
                backup_script={{BACKUP_SCRIPT}}
                ssh -i $keypair $user@$server -c "/usr/bin/bash $backup_script"
          - action: aws:runShellScript
            name: swapVirtualDiskContainer
            inputs:
              onFailure: Abort
              timeoutSeconds: "1200"
              runCommand:
              - |
                mv {{TapeBackupContainerFile}} {{TapeBackupContainerFile}}.$(date +%s)
                touch {{TapeBackupContainerFile}}
          - action: aws:runShellScript
            name: uploadBackupArchiveToS3
            inputs:
              onFailure: Abort
              timeoutSeconds: "1200"
              runCommand:
              - |
                aws s3 cp {{TapeBackupContainerFile}} s3://{{BACKUP_BUCKET}}/
{{SUN_SPARC_IP}}/$(date '+%Y-%m-%d')/
 ...
```

Backup option 2 – Stromasys snapshot

You can use the following sample Systems Manager Automation runbook code to automate the backup process:

```
      ...

         mainSteps:
         - action: aws:runShellScript
           name: startSnapshot
           inputs:
             onFailure: Abort
             timeoutSeconds: "1200"
             runCommand:
             - |
               # You may consider some graceful stop of the application before taking a
 snapshot
               # Query SSP PID by configuration file
               # Example: ps ax | grep ssp-4 | grep Solaris10.cfg | awk '{print $1"
 "$5}' | grep ssp4 | cut -f1 -d" "
```

```
              pid=`ps ax | grep ssp-4 | grep {{SSP_GUEST_CONFIG_FILE}} | awk '{print
 $1" "$5}' | grep ssp4 | cut -f1 -d" "`
              if [ -n "${pid}" ]; then
                kill -SIGTSTP ${pid}
              else
                echo "No PID found for SPARC guest with config
 {{SSP_GUEST_CONFIG_FILE}}"
                exit 1
              fi
        - action: aws:runShellScript
          name: startBackup
          inputs:
            onFailure: Abort
            timeoutSeconds: "1200"
            runCommand:
            - |
              # upload snapshot and virtual disk files into S3
              aws s3 sync {{SNAPSHOT_FOLDER}} s3://{{BACKUP_BUCKET}}/$(date '+%Y-%m-
%d')/
              aws s3 cp {{VIRTUAL_DISK_FILE}} s3://{{BACKUP_BUCKET}}/$(date '+%Y-%m-
%d')/
        - action: aws:runShellScript
          name: restratSPARCGuest
          inputs:
            onFailure: Abort
            timeoutSeconds: "1200"
            runCommand:
            - |
              /opt/charon-ssp/ssp-4u/ssp4u -f {{SSP_GUEST_CONFIG_FILE}} -d -a
 {{SPARC_GUEST_NAME}} --snapshot {{SNAPSHOT_FOLDER}}
 ...
```

Backup option 4 – AWS Storage Gateway VTL

If you use Solaris non-global zones to run virtualized legacy Sun SPARC servers, the backup application approach can be applied to non-global zones running in the Sun SPARC servers (for example, the backup client can run inside the non-global zones). However, the backup client can also run in the Solaris host and take snapshots of the non-global zones. The snapshots can then be backed up on a tape.

The following sample configuration adds the file system that hosts the Solaris non-global zones into the backup configuration for the Solaris host:

```
FileSet {
   Name = "Branded Zones"
   Include {
     Options {
       signature = MD5
     }
     File = /zones
   }
 }
```

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Back up and archive data to Amazon S3 with Veeam Backup & Replication

Jeanna James, Anthony Fiore (AWS), and William Quigley, Amazon Web Services

Summary

This pattern details the process for sending backups created by Veeam Backup & Replication to supported Amazon Simple Storage Service (Amazon S3) object storage classes by using the Veeam scale-out backup repository capability.

Veeam supports multiple Amazon S3 storage classes to best fit your specific needs. You can choose the type of storage based on the data access, resiliency, and cost requirements of your backup or archive data. For example, you can store data that you don't plan to use for 30 days or longer in Amazon S3 infrequent access (IA) for lower cost. If you're planning to archive data for 90 days or longer, you can use Amazon Simple Storage Service Glacier (Amazon S3 Glacier) Flexible Retrieval or S3 Glacier Deep Archive with Veeam's archive tier. You can also use S3 Object Lock to make backups that are immutable within Amazon S3.

This pattern doesn't cover how to set up Veeam Backup & Replication with a tape gateway in AWS Storage Gateway. For information about that topic, see Veeam Backup & Replication using AWS VTL Gateway - Deployment Guide on the Veeam website.

Warning: This scenario requires AWS Identity and Access Management (IAM) users with programmatic access and long-term credentials, which present a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed. Access keys can be updated if necessary. For more information, see Updating access keys in the *IAM User Guide*.

Prerequisites and limitations

Prerequisites

- Veeam Backup & Replication, including Veeam Availability Suite or Veeam Backup Essentials, installed (you can register for a free trial)
- Veeam Backup & Replication license with Enterprise or Enterprise Plus functionality, which includes Veeam Universal License (VUL)

- An active IAM user with access to an Amazon S3 bucket
- An active IAM user with access to Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Virtual Private Cloud (Amazon VPC), if using archive tier
- Network connectivity from on premises to AWS services with available bandwidth for backup and restore traffic through a public internet connection or an AWS Direct Connect public virtual interface (VIF)
- The following network ports and endpoints opened to ensure proper communication with object storage repositories:
  - Amazon S3 storage – TCP – port 443: Used to communicate with Amazon S3 storage.
  - Amazon S3 storage – cloud endpoints – `*.amazonaws.com` for AWS Regions and the AWS GovCloud (US) Regions, or `*.amazonaws.com.cn` for China Regions: Used to communicate with Amazon S3 storage. For a complete list of connection endpoints, see [Amazon S3 endpoints]() in the AWS documentation.
  - Amazon S3 storage – TCP HTTP – port 80: Used to verify certificate status. Consider that certificate verification endpoints—certificate revocation list (CRL) URLs and Online Certificate Status Protocol (OCSP) servers—are subject to change. The actual list of addresses can be found in the certificate itself.
  - Amazon S3 storage – certificate verification endpoints – `*.amazontrust.com`: Used to verify certificate status. Consider that certificate verification endpoints (CRL URLs and OCSP servers) are subject to change. The actual list of addresses can be found in the certificate itself.

Limitations

- Veeam doesn't support S3 Lifecycle policies on any S3 buckets that are used as Veeam object storage repositories. These include polices with Amazon S3 storage class transitions and S3 Lifecycle expiration rules. Veeam **must** be the sole entity that manages these objects. Enabling S3 Lifecycle policies might have unexpected results, including data loss.

Product versions

- Veeam Backup & Replication v9.5 Update 4 or later (backup only or capacity tier)
- Veeam Backup & Replication v10 or later (backup or capacity tier and S3 Object Lock)
- Veeam Backup & Replication v11 or later (backup or capacity tier, archive or archive tier, and S3 Object Lock)

- Veeam Backup & Replication v12 or later (performance tier, backup or capacity tier, archive or archive tier, and S3 Object Lock)
- S3 Standard
- S3 Standard-IA
- S3 One Zone-IA
- S3 Glacier Flexible Retrieval (v11 and later only)
- S3 Glacier Deep Archive (v11 and later only)
- S3 Glacier Instant Retrieval (v12 and later only)

Architecture

Source technology stack

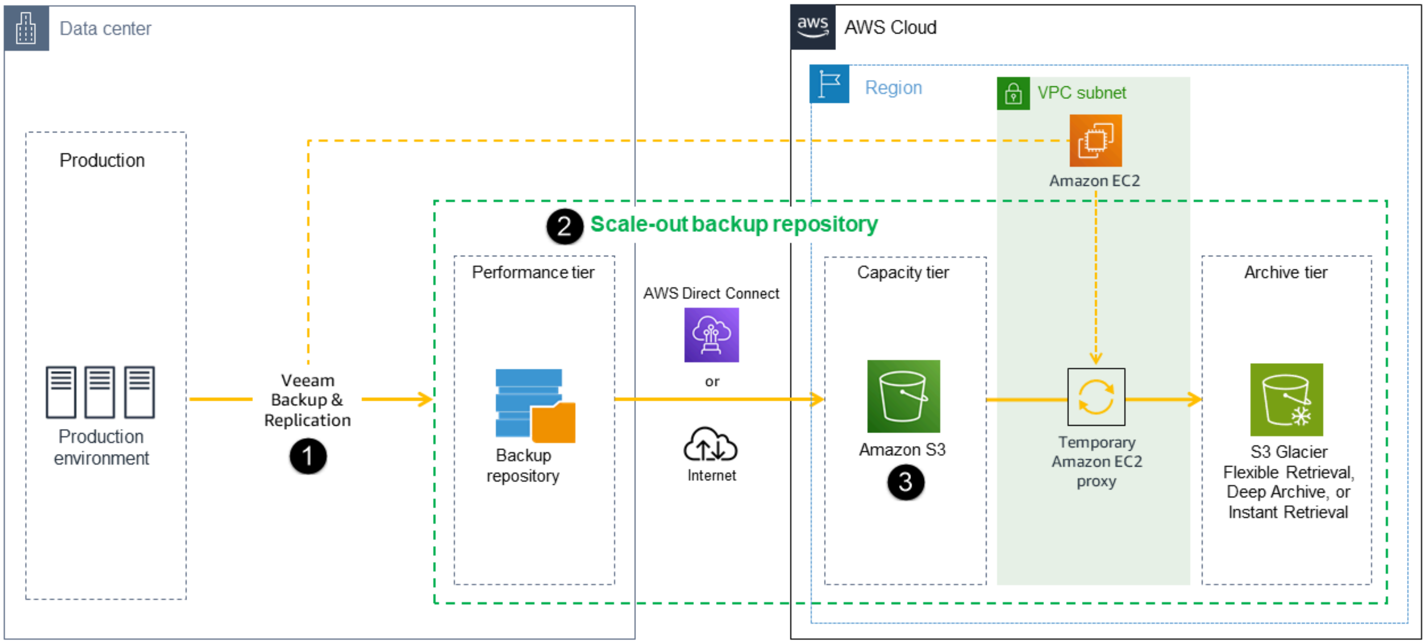
- On-premises Veeam Backup & Replication installation with connectivity from a Veeam backup server or a Veeam gateway server to Amazon S3

Target technology stack

- Amazon S3
- Amazon VPC and Amazon EC2 (if using archive tier)

Target architecture: SOBR

The following diagram shows the scale-out backup repository (SOBR) architecture.



Veeam Backup and Replication software protects data from logical errors such as system failures, application errors, or accidental deletion. In this diagram, backups are run on premises first, and a secondary copy is sent directly to Amazon S3. A backup represents a point-in-time copy of the data.

The workflow consists of three primary components that are required for tiering or copying backups to Amazon S3, and one optional component:

- Veeam Backup & Replication (1) – The backup server that is responsible for coordinating, controlling, and managing backup infrastructure, settings, jobs, recovery tasks, and other processes.
- Veeam gateway server (not shown in the diagram) – An optional on-premises gateway server that is required if the Veeam backup server doesn't have outbound connectivity to Amazon S3.
- Scale-out backup repository (2) – Repository system with horizontal scaling support for multi-tier storage of data. The scale-out backup repository consists of one or more backup repositories that provide fast access to data and can be expanded with Amazon S3 object storage repositories for long-term storage (capacity tier) and archiving (archive tier). Veeam uses the scale-out backup repository to tier data automatically between local (performance tier) and Amazon S3 object storage (capacity and archive tiers).

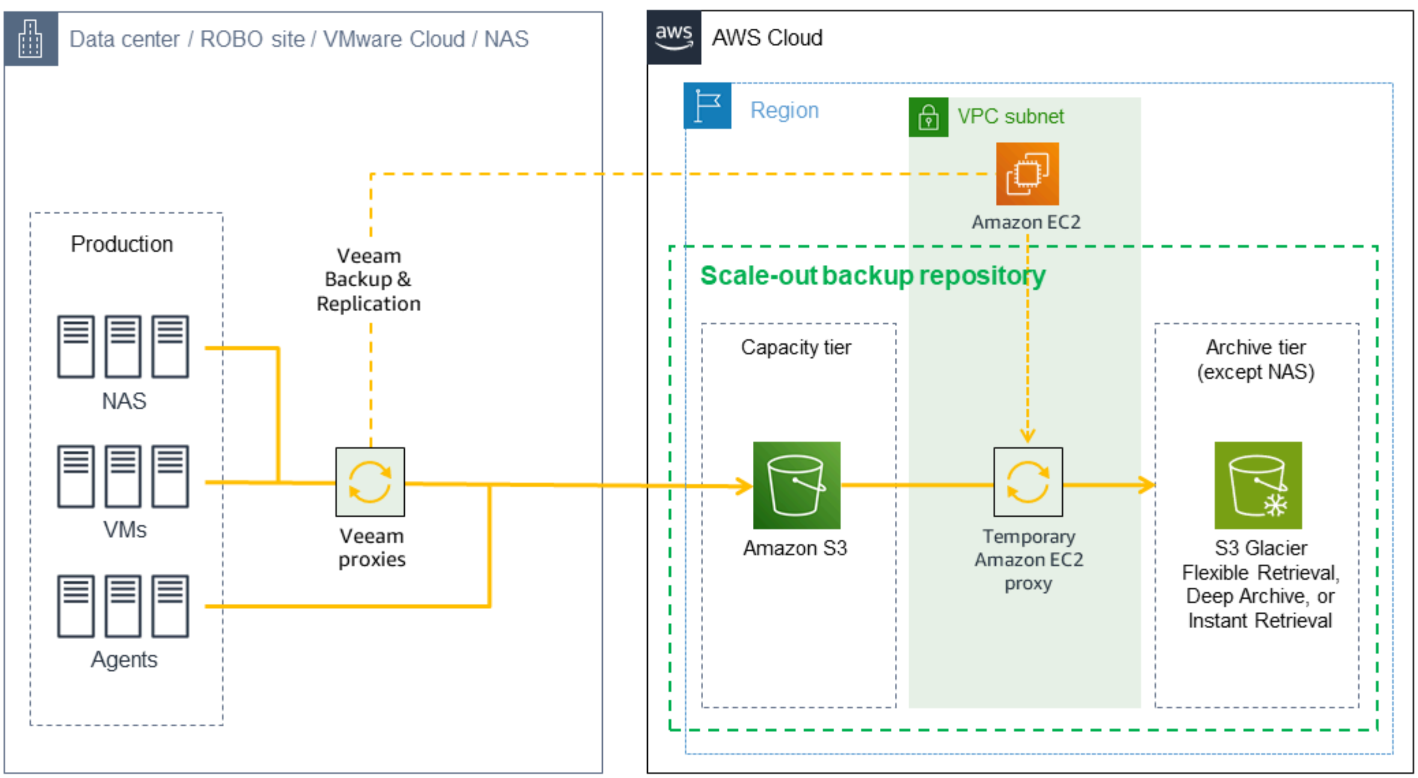
Note

Starting with Veeam Backup & Replication v12.2, the Direct to S3 Glacier feature makes the S3 capacity tier optional. A SOBR can be configured with a performance tier and a S3 Glacier archive tier. This configuration is useful for users who have significant investments in local (on-premises) storage for the capacity tier and who require only long-term archive retention in the cloud. For more information, see the [Veeam Backup & Replication documentation](https://helpcenter.veeam.com/docs/backup/vsphere/archive_tier.html).

- Amazon S3 (3) – AWS object storage service that offers scalability, data availability, security, and performance.

Target architecture: DTO

The following diagram shows the direct-to-object (DTO) architecture.



In this diagram, backup data goes directly to Amazon S3 without being stored on premises first. Secondary copies can be stored in S3 Glacier.

Automation and scale

You can automate the creation of IAM resources and S3 buckets by using the AWS CloudFormation templates provided in the VeeamHub GitHub repository. The templates include both standard and immutable options.

Tools

Tools and AWS services

- Veeam Backup & Replication is a solution from Veeam for protecting, backing up, replicating, and restoring your virtual and physical workloads.
- AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. With IAM, you can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users and applications can access.
- Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- Amazon S3 Glacier (S3 Glacier) is a secure and durable service for low-cost data archiving and long-term backup.
- Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Code

Use the CloudFormation templates provided in the VeeamHub GitHub repository to automatically create the IAM resources and S3 buckets for this pattern. If you prefer to create these resources manually, follow the steps in the *Epics* section.

Best practices

- In accordance with IAM best practices, we strongly recommend that you regularly rotate long-term IAM user credentials, such as the IAM user that you use for writing Veeam Backup & Replication backups to Amazon S3. For more information, see Security best practices in the IAM documentation.

Epics

Configure Amazon S3 storage in your account

Task	Description	Skills required
Create an IAM user.	Follow the instructions in the IAM documentation to create an IAM user. This user should not have AWS console access, and you will need to create an access key for this user. Veeam uses this entity to authenticate with AWS to read and write to your S3 buckets. You must grant least privilege (that is, grant only the permissions required to perform a task) so the user doesn't have more authority than it needs. For example IAM policies to attach to your Veeam IAM user, see the Additional information section. **Note** Alternatively, you can use the	AWS administrator

Task	Description	Skills required
	CloudFormation templates provided in the VeeamHub GitHub repository to create an IAM user and S3 bucket for this pattern.	

Task	Description	Skills required
Create an S3 bucket.	1. Sign in to the AWS Management Console and open the Amazon S3 console. 2. If you don't already have an existing S3 bucket to use as the target storage, choose **Create bucket**, and specify a bucket name, AWS Region, and bucket settings. • We recommend that you enable the Block Public Access option for the S3 bucket and set up the access and user permission policies to meet your organizat ion's requirements. For an example, see the Amazon S3documen tation. • We recommend that you enable S3 Object Lock, even if you don't intend to use it right away. This setting can be enabled only at the time of S3 bucket creation. For more information, see Creating a bucket in the Amazon S3 documentation.	AWS administrator

Add Amazon S3 and S3 Glacier Flexible Retrieval (or S3 Glacier Deep Archive) to Veeam Backup & Replication

Task	Description	Skills required
Launch the New Object Repository wizard.	Before you set up the object storage and scale-out backup repositories in Veeam, you must add the Amazon S3 and S3 Glacier storage repositories that you want to use for the capacity and archive tiers. In the next epic, you’ll connect these storage repositories to your scale-out backup repository. 1. On the Veeam console, open the **Backup Infrastru cture** view. 2. In the inventory pane, choose the **Backup Repositories** node, and then choose **Add Repositor y**. 3. In the **Add Backup Repository** dialog box, choose **Object Storage**, **Amazon S3**.	AWS administrator, App owner
Add Amazon S3 storage for the capacity tier.	1. In the **Amazon Cloud Storage Services** dialog box, choose **Amazon S3**. 2. At the **Name** step of the wizard, specify the object storage name and a brief	AWS administrator, App owner

Task	Description	Skills required
	description, such as the creator and creation date. 3. At the **Account** step of the wizard, specify the object storage account. • For **Credentials**, choose the IAM user that you created in the first epic to access your Amazon S3 object storage. • For **AWS region**, choose the AWS Region where the S3 bucket is located. 4. At the **Bucket** step of the wizard, specify object storage settings. • For **Data center region**, choose the AWS Region where the S3 bucket is located. • For **Bucket**, choose the S3 bucket that you created in the first epic. • For **Folder**, create or select a cloud folder to map your object storage repository to. • If you want to enable immutability, choose **Make recent backups immutable for X days** and set the period of time during which	

Task	Description	Skills required
	your backups should be locked. Note that enabling immutability results in increased costs because of the increased number of API calls to Amazon S3 from Veeam. 5. At the **Summary** step of the wizard, review the configuration information, and then choose **Finish**.	

Task	Description	Skills required

Add S3 Glacier storage for the archive tier.	If you want to create an archive tier, use the IAM permissions detailed in the [Additional information](#) section.
	
	1. Launch the New Object Repository wizard as described previously.
	2. In the **Amazon Cloud Storage Services** dialog box, choose **Amazon S3 Glacier**.
	3. At the **Name** step of the wizard, specify the object storage name and a brief description, such as the creator and creation date.
	4. At the **Account** step of the wizard, specify the object storage account.
	  • For **Credentials**, choose the IAM user that you created in the first epic to access your S3 Glacier object storage.
	  • For **AWS region**, choose the AWS Region where the S3 bucket is located.
	5. At the **Bucket** step of the wizard, specify object storage settings.
		AWS administrator, App owner

Task	Description	Skills required
	• For **Data center region**, choose the AWS Region. • For **Bucket**, choose an S3 bucket to store your backup data. This can be the same bucket you used for the capacity tier. • For **Folder**, create or select a cloud folder to map your object storage repository to. • If you want to enable immutability, choose **Make recent backups immutable for the entire duration of their retention policy**. Note that enabling immutabil ity results in increased costs because of the increased number of API calls to Amazon S3 from Veeam. • If you want to use S3 Glacier Deep Archive as your archival storage class, choose **Use the Deep Archive Storage Class**. 6. At the **Proxy Appliance** step of the wizard, configure the auxiliary instance that is used to transfer the	

Task	Description	Skills required
	data from Amazon S3 to S3 Glacier. You can use the default settings or configure each setting manually. To configure the settings manually: • Choose **Customize**. • For **EC2 instance type**, choose the instance type for the proxy appliance, based on your speed and cost requirements for transferring the backup files to the archive tier of your scale-out backup repository. • For **Amazon VPC**, choose the VPC for the target instance. • For **Subnet**, choose the subnet for the proxy appliance. • For **Security group**, choose the security group to associate with the proxy appliance. • For **Redirector port**, specify the TCP port for routing requests between the proxy appliance and backup infrastructure components.	

Task	Description	Skills required
	• Choose **OK** to confirm your settings. 7. At the **Summary** step of the wizard, review the configuration information, and then choose **Finish**.	

Add scale-out backup repositories

Task	Description	Skills required
Launch the New Scale-Out Backup Repository wizard.	1. On the Veeam console, open the **Backup Infrastru cture** view. 2. In the inventory pane, choose **Scale-out Repositories,** and then choose **Add Scale-out Repository**.	App owner, AWS systems administrator
Add a scale-out backup repository and configure capacity and archive tiers.	1. At the **Name** step of the wizard, specify the name and a brief description of the scale-out backup repository. 2. If needed, add performan ce extents. You can also use your existing Veeam local backup repository as your performance tier. Starting with Veeam version 12, you can add an S3 bucket as a performance extent for direct-to-object (DTO)	App owner, AWS systems administrator

Task	Description	Skills required
	backups, bypassing a local performance tier. 3. Choose **Advanced**, and specify additional options for the scale-out backup repository. • Choose **Use per-machine backup files** to create a separate backup file for each machine and write these files to the backup repository in multiple streams simultaneously. This option is recommended for better storage and compute resource utilization. • Choose **Perform full backup when required extent is offline** to create a full backup file in case an extent that contains restore points for an incremental backup goes offline. This option requires free space in the scale-out backup repository to host a full backup file. 4. At the **Policy** step of the wizard, specify the backup placement policy for the repository.	

Task	Description	Skills required
	• Choose **Data locality** to store full and incremental backup files that belong to the same chain together, to the same performance extent. You can store files that belong to a new backup chain to the same performance extent or to another one (unless you use a deduplicating storage appliance as a performance extent). • Choose **Performance** to store full and incremental backup files to different performance extents. This option requires a fast and reliable network connection. If you choose **Performance**, you can restrict the types of backup files to store on each performance extent. For example, you can store full backup files on one extent and incremental backup files on other extents. To choose file types: • Choose **Customize**.	

Task	Description	Skills required
	- In the **Backup Placement Settings** dialog box, choose a performance extent, and then choose **Edit**. - Choose the type of backup files you want to store on the extent. 5. At the **Capacity Tier** step of the wizard, configure the long-term storage tier that you want to attach to the scale-out backup repository. - Choose **Extend scale-out backup repository capacity with object storage**. For the object storage repository, choose the Amazon S3 storage for the capacity tier that you added in the previous epic. - Choose **Window** to select a time window for moving or copying data. - Choose **Copy backups to object storage as soon as they are created** to copy all or only recently created backup files to the capacity extent.	

Task	Description	Skills required
	• Choose **Move backups to object storage as they age out of the operational restores window** to transfer inactive backup chains to the capacity extent. In the **Move backup files older than X days** field, specify a duration after which backup files should be offloaded . (To offload inactive backup chains on the day they were created, specify 0 days.) You can also choose **Override** to move backup files sooner if the scale-out backup repository has reached a threshold that you specify. • Choose **Encrypt data uploaded to object storage** and specify a password to encrypt all data and their metadata for offloading. Choose **Add** or **Manage passwords** to specify a new password. 6. At the **Archive Tier** step of the wizard, configure the archive storage tier	

Task	Description	Skills required
	that you want to attach to the scale-out backup repository. (This step doesn't appear if you skipped adding Amazon S3 Glacier storage.) • Choose **Archive GFS full backups to object storage**. For the object storage repository, choose the Amazon S3 Glacier storage you added in the previous epic. • For **Archive GFS backups older than N days**, choose a time window for moving files to the archive extent. (To archive inactive backup chains on the day they were created, specify 0 days.) 7. At the **Summary** step of the wizard, review the configuration of the scale-out backup repository, and then choose **Finish**.	

Related resources

- Creating an IAM user in your AWS account (IAM documentation)
- Creating a bucket (Amazon S3 documentation)

- Blocking public access to your Amazon S3 storage (Amazon S3 documentation)
- Using S3 Object Lock (Amazon S3 documentation)
- Veeam technical documentation
- How to Create Secure IAM Policy for Connection to S3 Object Storage (Veeam documentation)

Additional information

The following sections provide sample IAM policies you can use when you create an IAM user in the Epics section of this pattern.

IAM policy for capacity tier

> **Note**
>
> Change the name of the S3 buckets in the example policy from `<yourbucketname>` to the name of the S3 bucket that you want to use for Veeam capacity tier backups. Also note that the policy should be restricted to the specific resources used for Veeam (indicated by the `Resource` specification in the following policy), and that the first part of the policy disables client-side encryption, as discussed in the AWS blog post Preventing unintended encryption of Amazon S3 objects.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::<your-bucket-name>/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "false"
                }
            }
        },
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObjectVersion",
                "s3:ListBucketVersions",
                "s3:ListBucket",
                "s3:PutObjectLegalHold",
                "s3:GetBucketVersioning",
                "s3:GetObjectLegalHold",
                "s3:GetBucketObjectLockConfiguration",
                "s3:PutObject*",
                "s3:GetObject*",
                "s3:GetEncryptionConfiguration",
                "s3:PutObjectRetention",
                "s3:PutBucketObjectLockConfiguration",
                "s3:DeleteObject*",
                "s3:DeleteObjectVersion",
                "s3:GetBucketLocation"

            ],
            "Resource": [
                "arn:aws:s3:::<yourbucketname>",
                "arn:aws:s3:::<yourbucketname>/*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

IAM policy for archive tier

Note

Change the name of the S3 buckets in the example policy from `<yourbucketname>` to the name of the S3 bucket that you want to use for Veeam archive tier backups.

To use your existing VPC, subnet, and security groups:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
                "s3:RestoreObject",
                "s3:ListBucket",
                "s3:AbortMultipartUpload",
                "s3:GetBucketVersioning",
                "s3:ListAllMyBuckets",
                "s3:GetBucketLocation",
                "s3:GetBucketObjectLockConfiguration",
                "s3:PutObjectRetention",
                "s3:GetObjectVersion",
                "s3:PutObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:DeleteObjectVersion",
                "s3:ListBucketVersions"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>",
                "arn:aws:s3:::<bucket-name>/*"
            ]
        }
    ]
}

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2Permissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:CreateKeyPair",
                "ec2:DescribeKeyPairs",
```

```
                "ec2:RunInstances",
                "ec2:DeleteKeyPair",
                "ec2:DescribeVpcAttribute",
                "ec2:CreateTags",
                "ec2:DescribeSubnets",
                "ec2:TerminateInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeImages",
                "ec2:DescribeVpcs"
            ],
            "Resource": "arn:aws:ec2:<region>:<account-id>:*"
        }
    ]
}
```

To create new VPC, subnet, and security groups:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Permissions",
            "Effect": "Allow",
            "Action": [
                "s3:DeleteObject",
                "s3:PutObject",
                "s3:GetObject",
                "s3:RestoreObject",
                "s3:ListBucket",
                "s3:AbortMultipartUpload",
                "s3:GetBucketVersioning",
                "s3:ListAllMyBuckets",
                "s3:GetBucketLocation",
                "s3:GetBucketObjectLockConfiguration",
                "s3:PutObjectRetention",
                "s3:GetObjectVersion",
                "s3:PutObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:DeleteObjectVersion",
                "s3:ListBucketVersions"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>",
```

```
                "arn:aws:s3:::<bucket-name>/*"
            ]
        }
    ]
}

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EC2Permissions",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:CreateKeyPair",
                "ec2:DescribeKeyPairs",
                "ec2:RunInstances",
                "ec2:DeleteKeyPair",
                "ec2:DescribeVpcAttribute",
                "ec2:CreateTags",
                "ec2:DescribeSubnets",
                "ec2:TerminateInstances",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeImages",
                "ec2:DescribeVpcs",
                "ec2:CreateVpc",
                "ec2:CreateSubnet",
                "ec2:DescribeAvailabilityZones",
                "ec2:CreateRoute",
                "ec2:CreateInternetGateway",
                "ec2:AttachInternetGateway",
                "ec2:ModifyVpcAttribute",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteSecurityGroup",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:DescribeRouteTables",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        }
    ]
}
```

Configure Veritas NetBackup for VMware Cloud on AWS

Shubham Salani, Amazon Web Services

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

Many enterprises use Veritas NetBackup as a backup and recovery solution for their on-premises VMware vSphere-based workloads. Once enterprises migrate their workloads to software-defined data centers (SDDCs) in the VMware Cloud on Amazon Web Services (AWS) infrastructure, there is no clear lift-and-shift procedure to integrate NetBackup. This pattern describes how you can set up Veritas NetBackup in your AWS account and configure it to back up the workloads in your VMware SDDCs.

This pattern does not include instructions for migrating your workloads. For more information, see [Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX](). When setting up your workloads to VMware Cloud on AWS, use a [stretched cluster]() (VMware documentation). In this configuration, your cluster spans two AWS Availability Zones within a single Region. This provides high availability and resiliency in the event that one of the Availability Zones becomes unavailable. [Elastic DRS]() and a [vSAN witness host]() (VMware documentation) seamlessly copy the data to a third Availability Zone, known as a *fault domain*. This parity solution can help you recover the data in the event of a failure. Because this approach requires three Availability Zones, when selecting an AWS Region for your VMware Cloud environment, make sure that it has three or more Availability Zones. For more information, see [Regions and Availability Zones]().

In this pattern, each SDDC has a backup host, which is a proxy server. Using Amazon Elastic Compute Cloud (Amazon EC2) instances, you set up the NetBackup master and media servers in a separate virtual private cloud (VPC), one for each SDDC. Because elastic network interfaces provide high bandwidth and low latency, you use them to configure connectivity between the backup hosts and their corresponding NetBackup master and media servers. The EC2 instances direct the backups to Amazon Elastic Block Store (Amazon EBS) volumes, which is the first point of backup. You can use AWS DataSync to keep your EBS volumes for the SDDCs synchronized.

You can also use AWS Transit Gateway and an interface VPC endpoint to connect the EBS volumes to another storage service, such as Amazon Simple Storage Service (Amazon S3). According to your retention policy, you can use S3 Intelligent-Tiering S3 Glacier storage classes to optimize your storage costs. For more information, see Using Amazon S3 storage classes (Amazon S3 documentation).

Prerequisites and limitations

Prerequisites

- Your VMware Cloud on AWS environment uses a stretched cluster that spans two Availability Zones.
- The backup host must reside on the VMware Cloud on AWS SDDC that has access to the datastore where the VMware Virtual Machine Disk File (VMDK) files are deployed.
- HotAdd transport mode must be enabled on the NetBackup client to back up and restore virtual machines (VMs), and it must permit restores from user-directed files and folders.

Limitations

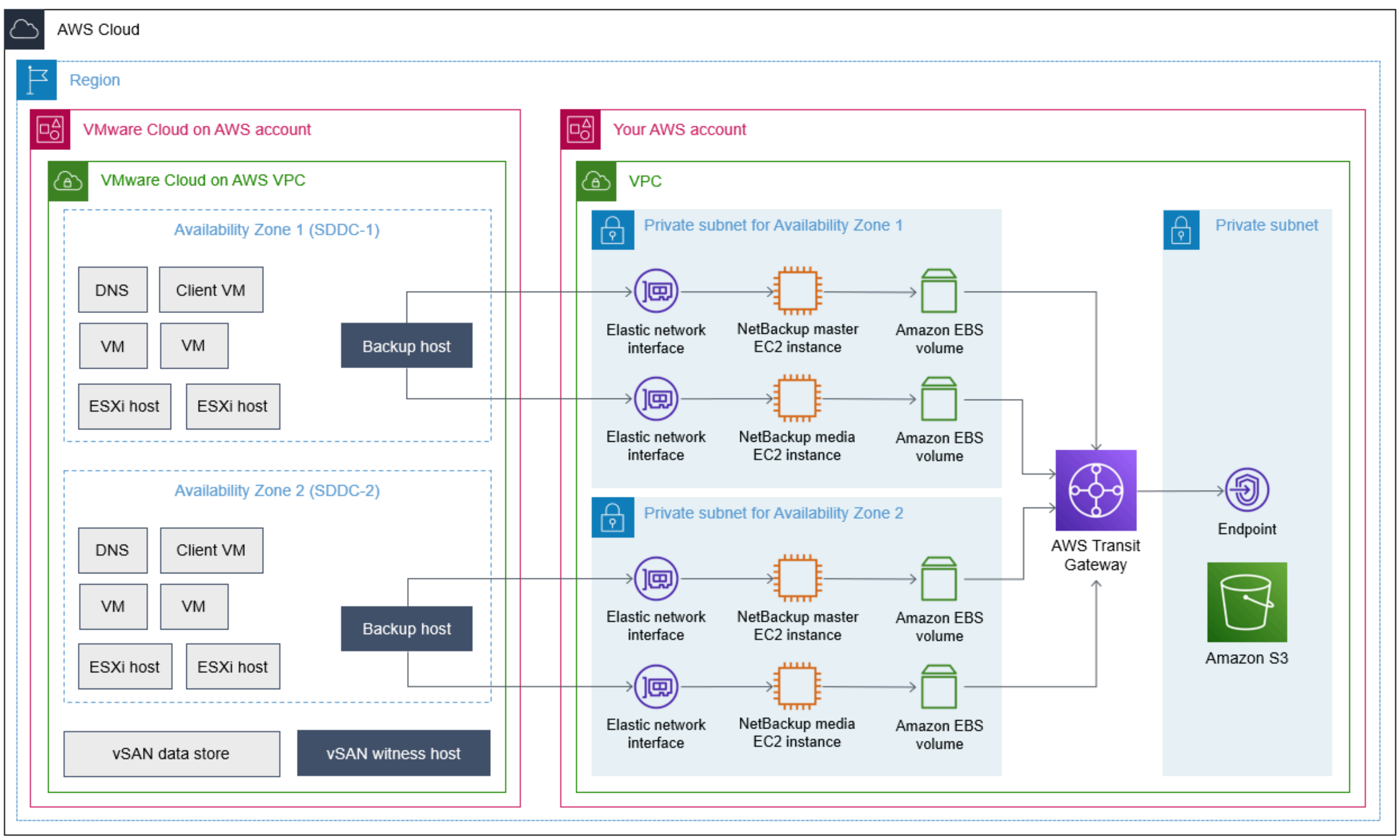
- The NetBackup master server must use DNS resolution to a private IP address for the vCenter backup host in the SDDC.
- The hosts files on the NetBackup master server and backup host should contain the following:
  - The private IP address and private DNS name of the master server
  - The private IP address and private DNS name of the backup host
- If you are configuring interface VPC endpoints to an S3 bucket, the SDDC Compute Gateway firewall must be configured to allow HTTPS from a Classless Inter-Domain Routing (CIDR) block source. For more information, see Access an S3 Bucket Using an S3 Endpoint (VMware documentation).
- VMware Cloud on AWS does not support the following features of NetBackup:
  - Backing up or restoring VM templates
  - Using NetBackup vSphere Client (HTML5 plug-in)
  - Locking and unlocking VMs for backups or restores
  - Backups cannot be stored in a vSAN datastore
  - Network block device (NBD), NBDSSL, and SAN transport modes

Product versions

- VMware Cloud on AWS SDDC version 1.0 or later
- Veritas NetBackup version 8.1.2 or later
- Linux version 6.8 or later
- VMware vSphere version 6.0 or later

Architecture

The following diagram shows the configuration of NetBackup for VMware Cloud on AWS. The NetBackup master and media servers are deployed in a separate VPC and are connected to the backup hosts in the SDDCs by elastic network interfaces. The NetBackup master and media servers store the backups in Amazon EBS volumes. You can optionally configure additional storage in Amazon S3 buckets by using AWS Transit Gateway and an AWS PrivateLink interface VPC endpoint.



Tools

AWS services and tools

- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances.
- AWS PrivateLink helps you create unidirectional, private connections from your virtual private clouds (VPCs) to services outside of the VPC.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other services

- VMware Cloud on AWS is an integrated cloud offering jointly developed by Amazon Web Services (AWS) and VMware.
- NetBackup for VMware backs up and restores the VMware virtual machines that run on VMware ESXi hosts.

Epics

Configure the NetBackup servers

Task	Description	Skills required
Update the firewall rules.	Update the firewall rules to establish connectivity between the VMware Cloud on AWS SDDC and the NetBackup master and media servers. Do the following: 1. Log in to VMware Cloud on AWS at https://vmc.vmware.com/ 2. On the **Networking and Security** tab, choose **Gateway Firewall**.	Network administrator, Cloud administrator

Task	Description	Skills required
	3. On the **Gateway Firewall** page, choose **Compute Gateway**. 4. Choose **ADD Rule**, and then create a new rule with the necessary firewall port settings. For more information, see NetBackup firewall port requirements (Veritas documentation).	

Task	Description	Skills required
Launch the NetBackup master and media servers.	1. Sign in to the AWS Management Console, and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/ 2. Launch an EC2 instance (Amazon EC2 documentation), and use the following configuration details: a. For the NetBackup master and media servers, select the `NBU-Linux-GA-8-1-2-Setup-f032d23e-881b-4dee-ba70-b9ca3e915910-ami-072509a7ffc156938.4` Amazon Machine Image (AMI). This preconfigured AMI is available through the AWS Marketplace. b. Select an instance type. NetBackup recommends `m5.2xlarge` for the master and media servers.	Cloud administrator, Backup administrator

Task	Description	Skills required
Configure the backup host for NetBackup.	1. Log in to VMware Cloud on AWS at https://vmc.vmware.com/ 2. Select the SDDC. 3. Choose the **Open VCENTER** tab. This opens the SDDC vCenter. 4. Note the fully qualified domain name (FQDN) of the backup host. 5. Log in to the NetBackup Administration Console. For more information, see [Logging in to the NetBackup Administration Console](#) (Veritas documentation). 6. Select the master and media servers, and then choose **VMware Access Hosts**. 7. Add the FQDN of the backup host. 8. Choose **Apply**, and then choose **OK**.	Cloud administrator, Backup administrator

(Optional) Set up Amazon S3 storage

Task	Description	Skills required
Configure storage in Amazon S3.	1. Review the [Amazon S3 cloud storage options](#) (Veritas documentation)	Cloud administrator, General AWS

Task	Description	Skills required
	and select the appropriate storage class for your requirements. 2. Configure NetBackup to use Amazon S3 for cloud storage according to the instructions in Configuring cloud storage in NetBackup (Veritas documentation).	

Related resources

AWS documentation

- Create an interface VPC endpoint (AWS PrivateLink documentation)

Veritas documentation

- NetBackup firewall port requirements

VMware documentation

- Deploy a VM from an OVF Template in a Content Library
- VMware Cloud on AWS data transfer charges: How it works? (VMware blog post)
- VMware Cloud on AWS: Stretched Clusters

Copy data from an S3 bucket to another account and Region by using the AWS CLI

Appasaheb Bagali and Purushotham G K, Amazon Web Services

Summary

This pattern describes how to migrate data from an Amazon Simple Storage Service (Amazon S3) bucket in an AWS source account to a destination S3 bucket in another AWS account, either in the same AWS Region or in a different Region.

The source S3 bucket allows AWS Identity and Access Management (IAM) access by using an attached resource policy. A user in the destination account has to assume a role that has `PutObject` and `GetObject` permissions for the source bucket. Finally, you run `copy` and `sync` commands to transfer data from the source S3 bucket to the destination S3 bucket.

Accounts own the objects that they upload to S3 buckets. If you copy objects across accounts and Regions, you grant the destination account ownership of the copied objects. You can change the ownership of an object by changing its [access control list (ACL)](#) to `bucket-owner-full-control`. However, we recommend that you grant programmatic cross-account permissions to the destination account because ACLs can be difficult to manage for multiple objects.

> Warning: This scenario requires IAM users with programmatic access and long-term credentia ls, which present a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed. Access keys can be updated if necessary. For more information, see [Updating access keys](#) in the *IAM User Guide*.

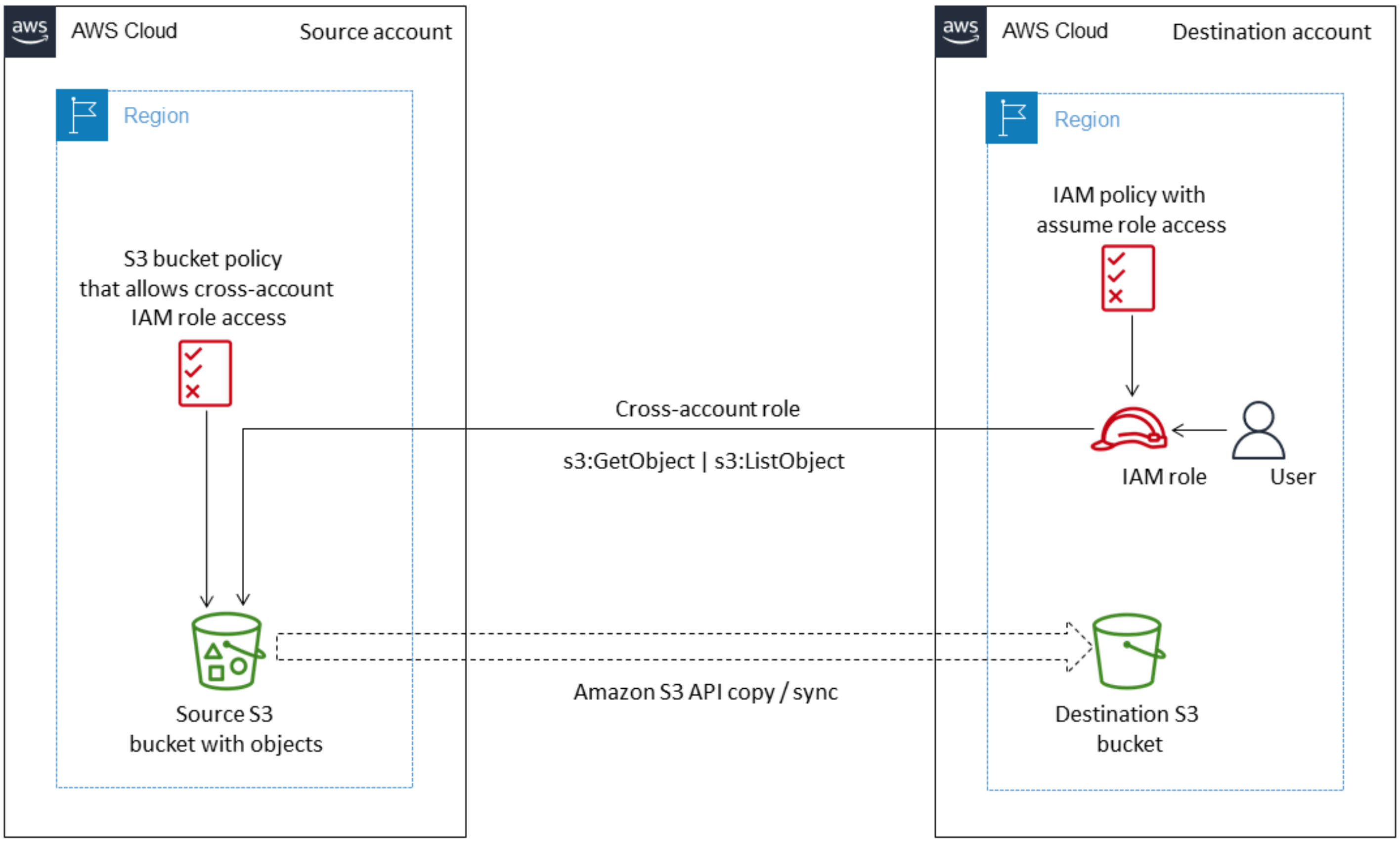
This pattern covers one-time migration. For scenarios that require continuous and automatic migration of new objects from a source bucket to a destination bucket, you can use S3 Batch Replication instead, as described in the pattern [Copy data from an S3 bucket to another account and Region by using S3 Batch Replication](#).

Prerequisites and limitations

- Two active AWS accounts in the same or different AWS Regions.
- An existing S3 bucket in the source account.

- If your source or destination Amazon S3 bucket has default encryption enabled, you must modify the AWS Key Management Service (AWS KMS) key permissions. For more information, see the AWS re:Post article on this topic.
- Familiarity with cross-account permissions.

Architecture



Tools

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Best practices

- Security best practices in IAM (IAM documentation)
- Applying least-privilege permissions (IAM documentation)

Epics

Create an IAM user and role in the destination AWS account

Task	Description	Skills required
Create an IAM user and get the access key.	1. Sign in to the AWS Management Console and create an IAM user that has programmatic access. For detailed steps, see Creating IAM users in the IAM documentation. There is no need to attach any policies for this user. 2. Generate an access key and secret key for this user. For instructions, see AWS Account and Access Keys in the AWS documentation.	AWS DevOps

Create an IAM identity-based policy.	Create an IAM Identity-based policy named `S3MigrationPolicy` by using the following permissions. For detailed steps, see Creating IAM policies in the IAM documentation.
	`{`
	`"Version":`
	`"2012-10-17",`
		AWS DevOps

Task	Description	Skills required

	<pre> "Statement": [
	        {
	            "Effect":
	 "Allow",
	            "Action": [
	
	 "s3:ListBucket",
	
	 "s3:GetObject",
	
	 "s3:GetObjectTaggi
	ng",
	
	 "s3:GetObjectVersi
	on",
	
	 "s3:GetObjectVersi
	onTagging"
	            ],
	            "Resource":
	 [
	
	 "arn:aws:s3:::amaz
	on-s3-demo-source-
	bucket",
	
	 "arn:aws:s3:::amazon-
	s3-demo-source-bucket/
	*"
	            ]
	        },
	        {
	            "Effect":
	 "Allow",
	            "Action": [
	
	 "s3:ListBucket",
	
	 "s3:PutObject",
	
	 "s3:PutObjectAcl",</pre>	

Task	Description	Skills required

	``
	 "s3:PutObjectTaggi
	ng",
	
	 "s3:GetObjectTaggi
	ng",
	
	 "s3:GetObjectVersi
	on",
	
	 "s3:GetObjectVersi
	onTagging"
	            ],
	            "Resource":
	 [
	
	 "arn:aws:s3:::amazon-
	s3-demo-destination-
	bucket",
	
	 "arn:aws:s3:::amazon-
	s3-demo-destination-
	bucket/*"
	            ]
	        }
	    ]
	}
	``
	
	Note
	
	Modify the source and destination bucket names according to your use case.
	
	This identity-based policy allows the user who is assuming this role to access	

Task	Description	Skills required
	the source bucket and destination bucket.	

Task	Description	Skills required
Create an IAM role.	Create an IAM role named `S3MigrationRole` by using the following trust policy, and then attach the previously created `S3MigrationPolicy`. For detailed steps, see [Creating a role to delegate permissions to an IAM user](#) in the IAM documentation. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": "arn:aws:iam::<destination_account>:user/<user_name>" }, "Action": "sts:AssumeRole", "Condition": {} } ] }` **Note** Modify the Amazon Resource Name (ARN) of the destination	AWS DevOps

Task	Description	Skills required
	IAM role or user name in the trust policy according to your use case. This trust policy allows the newly created IAM user to assume `S3MigrationRole` .	

Create and attach the S3 bucket policy in the source account

Task	Description	Skills required

Create and attach an S3 bucket policy.	Sign in to the AWS Management Console for your source account and open the Amazon S3 console. Choose your source S3 bucket and then choose **Permissions**. Under **Bucket policy**, choose **Edit** and then paste the following bucket policy. Choose **Save**.
	
	`{`
	`    "Version":`
	` "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Sid":`
	` "DelegateS3Access",`
	`            "Effect":`
	` "Allow",`
	`            "Principa`
	`l": {"AWS": "arn:aws:`
		Cloud administrator

Task	Description	Skills required

	``
	iam::<destination_
	account>:role/<Rol
	eName>"},
	            "Action":
	 ["s3:ListBucket",
	
	 "s3:GetObject",
	
	  "s3:GetObjectTaggi
	ng",
	
	  "s3:GetObjectVersi
	on",
	
	  "s3:GetObjectVersi
	onTagging"
	
	],
	            "Resource":
	 [
	
	 "arn:aws:s3:::amazon-
	s3-demo-source-bucket/
	*",
	
	 "arn:aws:s3:::amazon-
	s3-demo-source-bucket"
	            ]
	        }
	    ]
	}
	``
	
	Note
	
	Make sure that you include the AWS account ID for the destination account and configure the bucket policy	

Task	Description	Skills required
	template according to your requirements. This resource-based policy allows the destination role `S3MigrationRole` to access S3 objects in the source account.	

Configure the destination S3 bucket

Task	Description	Skills required
Create a destination S3 bucket.	Sign in to the AWS Management Console for your destination account, open the Amazon S3 console, and then choose **Create bucket**. Create an S3 bucket according to your requireme nts. For more information, see [Creating a bucket](#) in the Amazon S3 documentation.	Cloud administrator

Copy data to the destination S3 bucket

Task	Description	Skills required
Configure AWS CLI with the newly created user credentia ls.	1. Install the latest release of the AWS CLI. For instructi ons, see [Installing or updating the latest version](#)	AWS DevOps

Task	Description	Skills required
	of the AWS CLI in the AWS CLI documentation. 2. Run `$ aws configure` and update CLI with the AWS access key of the user you created. For more information, see Configuration and credential file settings in the AWS CLI documentation.	

Task	Description	Skills required
Assume the S3 migration role.	1. Use the AWS CLI to assume `S3MigrationRole` : `aws sts assume-role \ --role-arn "arn:aws:iam::<destination_account>:role/S3MigrationRole" \ --role-session-name AWSCLI-Session` This command outputs several pieces of information. Inside the credentials block you need the `AccessKeyId` , `SecretAccessKey` , and `SessionToken` . This example uses the environment variables `RoleAccessKeyID` , `RoleSecretKey` , and `RoleSessionToken` . Note that the timestamp of the expiration field is in the UTC time zone. The timestamp indicates when the temporary credentials of the IAM role expire. If the temporary credentials expire, you must call the `sts:AssumeRole` API again.	AWS administrator

Task	Description	Skills required

	2. Create three environment variables to assume the IAM role. These environment variables are filled out with the following output:
	
	`# Linux`
	`export AWS_ACCESS_KEY_ID=RoleAccessKeyID`
	`export AWS_SECRET_ACCESS_KEY=RoleSecretKey`
	`export AWS_SESSION_TOKEN=RoleSessionToken`
	`# Windows`
	`set AWS_ACCESS_KEY_ID=RoleAccessKeyID`
	`set AWS_SECRET_ACCESS_KEY=RoleSecretKey`
	`set AWS_SESSION_TOKEN=RoleSessionToken`
	
	3. Verify that you assumed the IAM role by running the following command:
	
	`aws sts get-caller-identity`
	
	For more information, see the AWS Knowledge Center.	

Task	Description	Skills required

Copy and synchronize data from the source S3 bucket to the destination S3 bucket.	When you have assumed the role `S3MigrationRole` you can copy the data using the copy (**cp**) or synchronize (**sync**) command.
	
	Copy (see the [AWS CLI Command Reference](#) for details):
	
	`aws s3 cp s3://amazon-s3-demo-source-bucket/ \ s3://amazon-s3-demo-destination-bucket/ \ --recursive --source-region SOURCE-REGION-NAME --region DESTINATION-REGION-NAME`
	
	Synchronize (see the [AWS CLI Command Reference](#) for details):
	
	`aws s3 sync s3://amazon-s3-demo-source-bucket/ \ s3://amazon-s3-demo-destination-bucket/ \ --source-region SOURCE-REGION-NAME --region DESTINATION-REGION-NAME`
		Cloud administrator

Troubleshooting

Issue	Solution
An error occurred (`AccessDenied`) when calling the `ListObjects` operation: Access Denied	• Make sure that you have assumed the role `S3MigrationRole` . • Run `aws sts get-caller-identity` to check the role used. If the output doesn't display the ARN for `S3MigrationRole` , assume the role again and retry.

Related resources

- Creating an S3 bucket (Amazon S3 documentation)
- Amazon S3 bucket policies and user policies (Amazon S3 documentation)
- IAM identities (users, groups, and roles) (IAM documentation)
- cp command (AWS CLI documentation)
- sync command (AWS CLI documentation)

Copy data from an S3 bucket to another account and Region by using S3 Batch Replication

Appasaheb Bagali, Lakshmikanth B D, Shubham Harsora, Purushotham G K, and Suman Rajotia, Amazon Web Services

Summary

This pattern explains how you can use Amazon Simple Storage Service (Amazon S3) Batch Replication to copy the contents of an S3 bucket to another S3 bucket automatically, without any manual intervention, after you set up the buckets. The source and destination buckets can be in the same or in different AWS accounts or Regions.

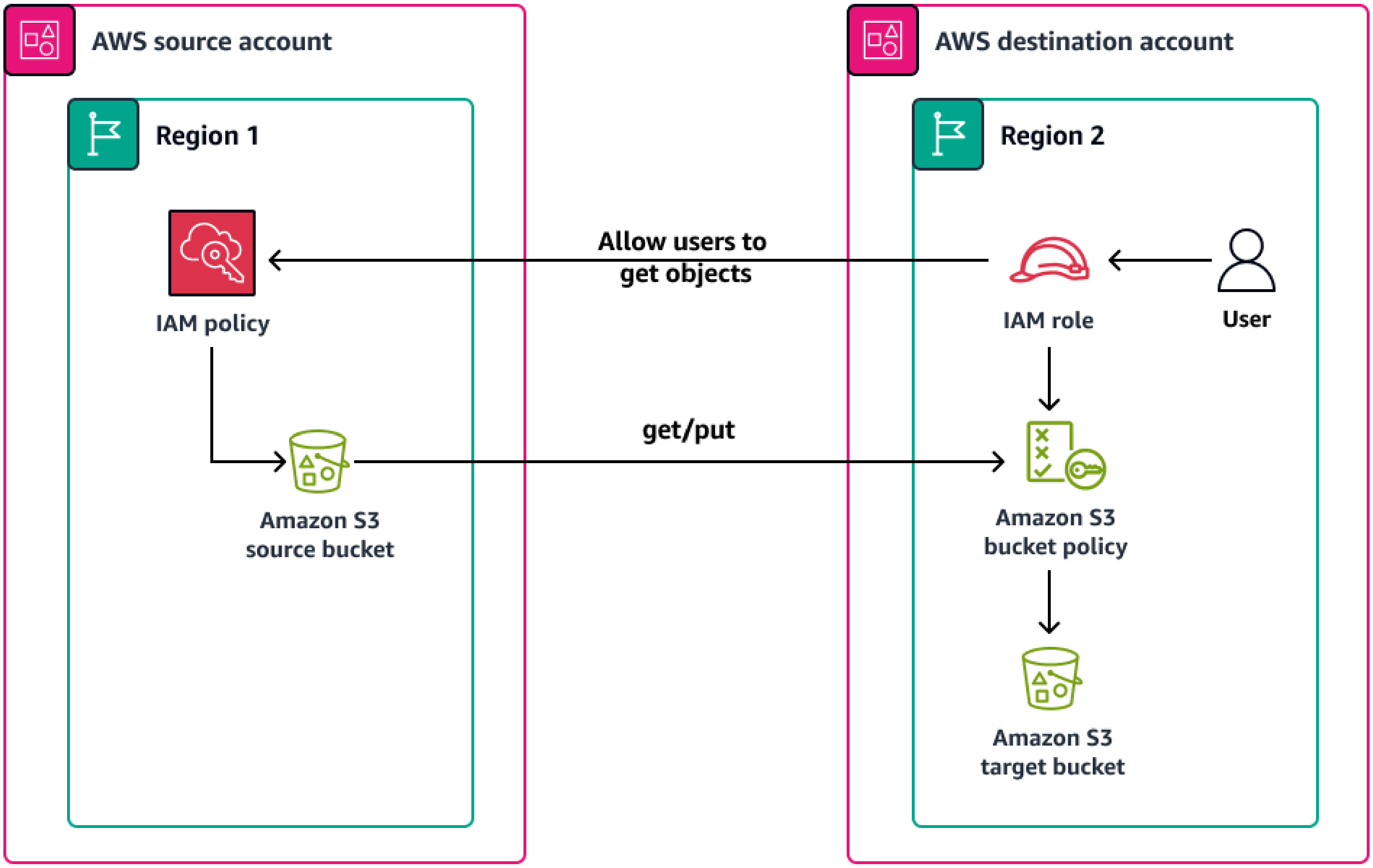
S3 Batch Replication gives you a way to replicate Amazon S3 objects that existed before a replication configuration was in place, objects that were previously replicated, and objects that failed replication. This method uses an S3 Batch Operations job. When the job finishes, you receive a completion report.

You can use S3 Batch Replication in scenarios that require continuous and automatic migration of new objects from a source bucket to a destination bucket. For one-time migration, you can use the AWS Command Line Interface (AWS CLI) instead, as described in the pattern Copy data from an S3 bucket to another account and Region by using the AWS CLI.

Prerequisites and limitations

- A source AWS account.
- A destination AWS account.
- An S3 bucket in the source account with a few objects (files or folders).
- One or more S3 buckets in the destination account.
- S3 Versioning enabled on the source and destination buckets.
- AWS Identity and Access Management (IAM) permissions to create an IAM policy, IAM role, and S3 bucket policy on the source and destination accounts.
- Amazon S3 Lifecycle rules disabled while the S3 Batch Replication job is active. This ensures parity between the source and destination buckets. Otherwise, the destination bucket might not be an exact replica of the source bucket.

Architecture



Tools

AWS services

- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [Amazon Simple Storage Service (Amazon S3)](#) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Best practices

The following video from AWS re:Invent 2022 discusses best practices for using Amazon S3 replication for regulatory compliance, data protection, and increased application performance.

https://www.youtube-nocookie.com/embed/hrJEbISBL04?controls=0

Epics

Create an IAM policy and role for cross-account replication in the source account

Task	Description	Skills required

Create an IAM policy for cross-account replication.	In the AWS source account:
	
	1. Open the [IAM console](https://console.aws.amazon.com/iam/).
	2. Create a new IAM policy.
	3. In the **Policy editor** section, choose **JSON**, and paste the following code.
	
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Sid": "GetSourceBucketConfiguration",`
	`            "Effect": "Allow",`
	`            "Action": [`
	`                "s3:ListBucket",`
	`                "s3:GetBucketLocation",`
	`                "s3:GetBucketAcl",`
	`                "s3:GetReplicationConfiguration",`
	`                "s3:GetObjectVersionForReplication",`
	`                "s3:GetObjectVersionAcl",`
		Cloud administrator, AWS administrator

Task	Description	Skills required

	``
	 "s3:GetObjectVersi
	onTagging"
	            ],
	            "Resource
	": [
	
	 "arn:aws:s3:::sour
	ce-bucket-name",
	
	 "arn:aws:s3:::sour
	ce-bucket-name/*"
	            ]
	        },
	        {
	            "Sid":
	 "ReplicateToDestin
	ationBuckets",
	            "Effect":
	 "Allow",
	            "Action":
	 [
	
	 "s3:List*",
	
	 "s3:*Object",
	
	 "s3:ReplicateObjec
	t",
	
	 "s3:ReplicateDelet
	e",
	
	 "s3:ReplicateTags"
	            ],
	            "Resource
	": [
	
	 "arn:aws:s3:::dest
	ination-bucket-nam
	e*",
	``	

Task	Description	Skills required

	<pre>"arn:aws:s3:::dest
	ination-bucket-nam
	e/*"
	            ]
	        },
	        {
	            "Sid":
	 "PermissionToOverr
	ideBucketOwner",
	            "Effect":
	 "Allow",
	            "Action":
	 [
	
	 "s3:ObjectOwnerOve
	rrideToBucketOwner"
	            ],
	            "Resource
	": [
	
	 "arn:aws:s3:::dest
	ination-bucket-nam
	e*",
	
	 "arn:aws:s3:::dest
	ination-bucket-nam
	e/*"
	            ]
	        }
	    ]
	}</pre>This policy includes three statements:
	
	• `GetSourceBucketConfiguration` provides access to the replication configuration and object	

Task	Description	Skills required
	version for replication on the source bucket. • `ReplicateToDestinationBuckets` provides access to replicate to the destination bucket. You can specify multiple destination buckets in the array. • `PermissionToOverrideBucketOwner` provides access to `ObjectOwnerOverrideToBucketOwner` so that the destination bucket can own the objects in the destination account that were replicated from the source account. 4. Choose **Next**, provide a policy name such as `cross-account-bucket-replication-policy` , and then choose **Create policy**. For more information, see [Creating IAM policies](#) in the IAM documentation.	

Task	Description	Skills required

Create an IAM role for cross-account replication.	In the AWS source account:
	
	1. On the [IAM console](https://console.aws.amazon.com/iam/), create an IAM role with the following information:
	  a. For **Trusted entity type**, choose **AWS service**.
	  b. For service, choose **S3**.
	  c. For use case, choose **S3 Batch Operations**.
	  d. Choose the policy you created in the previous step.
	2. Provide a role name such as cross-account-bucket-replication-role, and then choose **Create role**.
	
	For more information, see [Creating IAM roles](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html) in the IAM documentation.
		Cloud administrator, AWS administrator

Create a replication rule in the source account

Task	Description	Skills required
Create a replication rule against the source bucket in the source account.	In the AWS source account: 1. Open the [Amazon S3 console](https://console.aws.amazon.com/s3/). 2. Navigate to the source bucket, and choose the **Management** tab.	AWS administrator, Cloud administrator

Task	Description	Skills required
	3. Create a replication rule with the following configuration: a. Provide a rule name such as `s3-replication-rule`. b. For **Status**, choose **Enabled**. c. For rule scope, choose **Applies to all objects in the bucket**. d. For **Destination**, choose **Specify a bucket in another account**, and then enter the destination AWS account number and the bucket name. e. Choose the option to change object ownership to the destination bucket owner. f. For **IAM role**, choose the role you created earlier in the source account. g. For **Additional replication options**, select all available options. These provide the ability to replicate content quickly, monitor the progress of replication through Amazon CloudWatch metrics,	

Task	Description	Skills required
	replicate delete markers, and replicate metadata changes. h. Choose **Save**. 4. If you have multiple destination buckets, create additional replication rules. For more information, see [Configuring replication when source and destinati on buckets are owned by different accounts]() in the Amazon S3 documentation.	

Apply a bucket policy to the destination bucket

Task	Description	Skills required
Apply a bucket policy to the destination bucket.	This step has to be performed for each destination bucket individually in AWS destinati on accounts. In the AWS destination account: 1. Open the [Amazon S3 console](), navigate to the destination bucket, and choose the **Permissions** tab. 2. Edit the bucket policy by providing the following	AWS administrator, AWS systems administrator, Cloud administrator

Task	Description	Skills required

	JSON code, and save the policy:
	`{`
	`    "Version":`
	` "2012-10-17",`
	`    "Id": "PolicyFo`
	`rDestinationBucket",`
	`    "Statement": [`
	`        {`
	`            "Sid":`
	` "Permissions on`
	` objects and buckets",`
	`            "Effect":`
	` "Allow",`
	`            "Principa`
	`l": {`
	`                "AWS":`
	` "arn:aws:iam::Sour`
	`ceAWSAccountNumber`
	`:role/IAM-Role-cre`
	`ated-in-step1-in-s`
	`ource-account"`
	`            },`
	`            "Action": [`
	
	` "s3:List*",`
	
	` "s3:GetBucketVersi`
	`oning",`
	
	` "s3:PutBucketVersi`
	`oning",`
	
	` "s3:ReplicateDelete",`
	
	` "s3:ReplicateObject"`
	`            ],`
	`            "Resource":`
	` [`	

Task	Description	Skills required

	``
	 "arn:aws:s3:::dest
	ination-bucket",
	
	 "arn:aws:s3:::dest
	ination-bucket/*"
	            ]
	        },
	        {
	            "Sid":
	 "Permission to
	 override bucket owner",
	            "Effect":
	 "Allow",
	            "Principa
	l": {
	                "AWS":
	 "arn:aws:iam::Sour
	ceAWSAccountNumber
	:role/IAM-Role-cre
	ated-in-step1-in-s
	ource-account"
	            },
	            "Action":
	 "s3:ObjectOwnerOve
	rrideToBucketOwner",
	            "Resource
	": "arn:aws:s3:::dest
	ination-bucket/*"
	        }
	    ]
	}
	``
	
	This policy includes two statements:
	
	• `Permissions on objects and buckets` indicates that the destinati on bucket can replicate	

Task	Description	Skills required
	content based on the role defined in the source account. The role provides permissions to the source bucket. • `Permission to override bucket owner` indicates that the destinati on bucket has permissions to override the ownership from the source account.	

Test Amazon S3 cross-account replication

Task	Description	Skills required
Verify that replication works correctly.	1. Add an object to the source bucket. 2. Verify that the new object appears in the S3 buckets in the destination accounts. 3. View CloudWatch metrics: a. In the source bucket, choose the **Metrics** tab. b. In the **Replication metrics** section, select a replication rule. c. Choose **Display charts**. The charts reflect the state of replication by displaying the operation s that are pending replication, the replicati	AWS administrator, Cloud administrator

Task	Description	Skills required
	on latency, and the bytes pending replication. For more information, see [Monitoring metrics with Amazon CloudWatch] in the Amazon S3 documentation.	

Related resources

- When do I use IAM? (IAM documentation)
- How IAM works (IAM documentation)
- Creating IAM roles (IAM documentation)
- Creating IAM policies (IAM documentation)
- Overview of access management: Permissions and policies (IAM documentation)
- Creating, configuring, and working with Amazon S3 buckets (Amazon S3 documentation)
- Uploading, downloading, and working with objects in Amazon S3 (Amazon S3 documentation)
- Replicating objects (Amazon S3 documentation)

Migrate data from an on-premises Hadoop environment to Amazon S3 using DistCp with AWS PrivateLink for Amazon S3

Jason Owens, Andres Cantor, Jeff Klopfenstein, Bruno Rocha Oliveira, and Samuel Schmidt, Amazon Web Services

Summary

This pattern demonstrates how to migrate nearly any amount of data from an on-premises Apache Hadoop environment to the Amazon Web Services (AWS) Cloud by using the Apache open-source tool DistCp with AWS PrivateLink for Amazon Simple Storage Service (Amazon S3). Instead of using the public internet or a proxy solution to migrate data, you can use AWS PrivateLink for Amazon S3 to migrate data to Amazon S3 over a private network connection between your on-premises data center and an Amazon Virtual Private Cloud (Amazon VPC). If you use DNS entries in Amazon Route 53 or add entries in the **/etc/hosts** file in all nodes of your on-premises Hadoop cluster, then you are automatically directed to the correct interface endpoint.

This guide provides instructions for using DistCp for migrating data to the AWS Cloud. DistCp is the most commonly used tool, but other migration tools are available. For example, you can use offline AWS tools like AWS Snowball or AWS Snowmobile, or online AWS tools like AWS Storage Gateway or AWS DataSync. Additionally, you can use other open-source tools like Apache NiFi.

Prerequisites and limitations

Prerequisites

- An active AWS account with a private network connection between your on-premises data center and the AWS Cloud
- Hadoop, installed on premises with DistCp
- A Hadoop user with access to the migration data in the Hadoop Distributed File System (HDFS)
- AWS Command Line Interface (AWS CLI), installed and configured
- Permissions to put objects into an S3 bucket

Limitations

Virtual private cloud (VPC) limitations apply to AWS PrivateLink for Amazon S3. For more information, see Interface endpoint properties and limitations and AWS PrivateLink quotas (AWS PrivateLink documentation).

AWS PrivateLink for Amazon S3 doesn't support the following:

- Federal Information Processing Standard (FIPS) endpoints
- Website endpoints
- Legacy global endpoints

Architecture

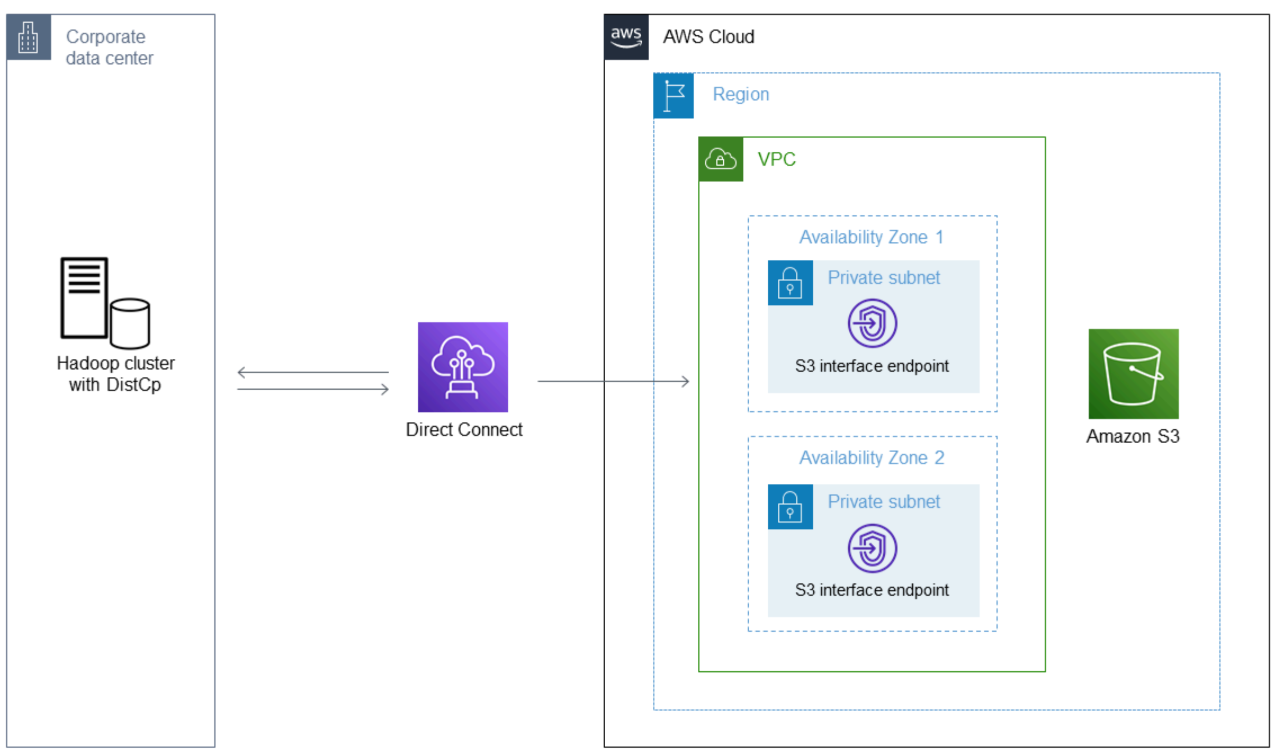
Source technology stack

- Hadoop cluster with DistCp installed

Target technology stack

- Amazon S3
- Amazon VPC

Target architecture



The diagram shows how the Hadoop administrator uses DistCp to copy data from an on-premises environment through a private network connection, such as AWS Direct Connect, to Amazon S3 through an Amazon S3 interface endpoint.

Tools

AWS services

- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- Apache Hadoop DistCp (distributed copy) is a tool used for copying large inter-clusters and intra-clusters. DistCp uses Apache MapReduce for distribution, error handling and recovery, and reporting.

Epics

Migrate data to the AWS Cloud

Task	Description	Skills required
Create an endpoint for AWS PrivateLink for Amazon S3.	1. Sign in to the AWS Management Console and open the Amazon VPC console. 2. On the navigation pane, choose **Endpoints**, and then choose **Create Endpoint**.	AWS administrator

Task	Description	Skills required
	3. For **Service category**, choose **AWS services**. 4. In the search box, enter **s3**, and then press **Enter**. 5. In the search results, choose the **com.amazonaws.<your-aws-region>.s3** service name where the value in the **Type** column is **Interface**. 6. For **VPC**, choose your VPC. For **Subnets**, choose your subnets. 7. For **Security group**, choose or create a security group that allows TCP 443. 8. Add tags based on your requirements and then choose **Create endpoint**.	

Task	Description	Skills required
Verify the endpoints and find the DNS entries.	1. Open the [Amazon VPC console](https://console.aws.amazon.com/vpc/), choose **Endpoints**, and then select the endpoint that you created earlier. 2. On the **Details** tab, find the first DNS entry for **DNS names**. This is the Regional DNS entry. When you use this DNS name, requests alternate between DNS entries specific to Availability Zones. 3. Choose the **Subnets** tab. You can find the address of the endpoint's elastic network interface in each Availability Zone.	AWS administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr>
<td>Check the firewall rules and routing configurations.</td>
<td>

To confirm that your firewall rules are open and that your networking configuration is correctly set up, use Telnet to test the endpoint on port 443. For example:

```
$ telnet vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com 443

Trying 10.104.88.6...

Connected to vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com.

...

$ telnet vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com 443

Trying 10.104.71.141...

Connected to vpce-<your-VPC-endpoint-ID>.s3.us-east-2.vpce.amazonaws.com.
```

Note

If you use the Regional entry, a successful test shows

</td>
<td>Network administrator, AWS administrator</td>
</tr>
</table>

Task	Description	Skills required
	that the DNS is alternating between the two IP addresses that you can see on the **Subnets** tab for your selected endpoint in the Amazon VPC console.	

Task	Description	Skills required
Configure the name resolutio n.	You must configure the name resolution to allow Hadoop to access the Amazon S3 interface endpoint. You can't use the endpoint name itself. Instead, you must resolve `<your-bucket-name> .s3.<your-aws-regi on>.amazonaws.com` or `*.s3.<your-aws-reg ion>.amazonaws.com` . For more information on this naming limitation, see [Introducing the Hadoop S3A client](#) (Hadoop website). Choose one of the following configuration options: • Use on-premises DNS to resolve the private IP address of the endpoint. You can override behavior for all buckets or selected buckets. For more informati on, see "Option 2: Access Amazon S3 using Domain Name System Response Policy Zones (DNS RPZ)" in [Secure hybrid access to Amazon S3 using AWS PrivateLink](#) (AWS blog post). • Configure on-premises DNS to conditionally forward	AWS administrator

Task	Description	Skills required
	traffic to the resolver inbound endpoints in the VPC. Traffic is forwarded to Route 53. For more information, see "Option 3: Forwarding DNS requests from on premises using Amazon Route 53 Resolver Inbound Endpoints" in [Secure hybrid access to Amazon S3 using AWS PrivateLink](#) (AWS blog post). • Edit the **/etc/hosts** file on all the nodes in your Hadoop cluster. This is a temporary solution for testing and isn't recommended for production. To edit the **/etc/hosts** file, add an entry for either `<your-bucket-name>.s3.<your-aws-region>.amazonaws.com` or `s3.<your-aws-region>.amazonaws.com`. The **/etc/hosts** file can't have multiple IP addresses for an entry. You must choose a single IP address from one of the Availability Zones, which then becomes a single point of failure.	

Task	Description	Skills required
Configure authentication for Amazon S3.	To authenticate to Amazon S3 through Hadoop, we recommend that you export temporary role credentials to the Hadoop environme nt. For more information, see [Authenticating with S3](#) (Hadoop website). For long-running jobs, you can create a user and assign a policy that has permissio ns to put data into an S3 bucket only. The access key and secret key can be stored on Hadoop, accessible only to the DistCp job itself and to the Hadoop administr ator. For more informati on on storing secrets, see [Storing secrets with Hadoop Credential Providers](#) (Hadoop website). For more informati on on other authentication methods, see [How to get credentials of an IAM role for use with CLI access to an AWS account](#) in the documentation for AWS IAM Identity Center (successor to AWS Single Sign-On). To use temporary credentials, add the temporary credentia ls to your credentials file, or run the following commands	AWS administrator

Task	Description	Skills required

	to export the credentials to your environment:
	`export AWS_SESSION_TOKEN=SECRET-SESSION-TOKEN`
	`export AWS_ACCESS_KEY_ID=SESSION-ACCESS-KEY`
	`export AWS_SECRET_ACCESS_KEY=SESSION-SECRET-KEY`
	
	If you have a traditional access key and secret key combination, run the following commands:
	`export AWS_ACCESS_KEY_ID=my.aws.key`
	`export AWS_SECRET_ACCESS_KEY=my.secret.key`
	
	Note
	If you use an access key and secret key combination, then change the credentials provider in the DistCp commands from `"org.apache.hadoop.fs.s3a.TemporaryAWSCredent`	

Task	Description	Skills required
	`ialsProvider"` to `"org.apache.hadoop.fs.s3a.SimpleAWSCredentialsProvider"` .	

Task	Description	Skills required
Transfer data by using DistCp.	To use DistCp to transfer data, run the following commands: `hadoop distcp -Dfs.s3a.aws.credentials.provider=\ "org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider" \ -Dfs.s3a.access.key="${AWS_ACCESS_KEY_ID}" \ -Dfs.s3a.secret.key="${AWS_SECRET_ACCESS_KEY}" \ -Dfs.s3a.session.token="${AWS_SESSION_TOKEN}" \ -Dfs.s3a.path.style.access=true \ -Dfs.s3a.connection.ssl.enabled=true \ -Dfs.s3a.endpoint=s3.<your-aws-region>.amazonaws.com \ hdfs:///user/root/ s3a://<your-bucket-name>` **Note** The AWS Region of the endpoint isn't automatically discovered when you use the DistCp	Migration engineer, AWS administrator

Task	Description	Skills required
	command with AWS PrivateLink for Amazon S3. Hadoop 3.3.2 and later versions resolve this issue by enabling the option to explicitly set the AWS Region of the S3 bucket. For more information, see [S3A to add option fs.s3a.endpoint.region to set AWS region](#) (Hadoop website). For more information on additional S3A providers, see [General S3A Client configuration](#) (Hadoop website). For example, if you use encryption, you can add the following option to the series of commands above depending on your type of encryption: `-Dfs.s3a.server-side-encryption-algorithm=AES-256 [or SSE-C or SSE-KMS]` **Note** To use the interface endpoint with S3A,	

Task	Description	Skills required
	you must create a DNS alias entry for the S3 Regional name (for example, `s3.<your-aws-region>.amazonaws.com`) to the interface endpoint. See the *Configure authentication for Amazon S3* section for instructions. This workaround is required for Hadoop 3.3.2 and earlier versions. Future versions of S3A won't require this workaround. If you have signature issues with Amazon S3, add an option to use Signature Version 4 (SigV4) signing: `-Dmapreduce.map.java.opts="-Dcom.amazonaws.services.s3.enableV4=true"`	

More patterns

- Access AWS services from IBM z/OS by installing the AWS CLI
- Automatically archive items to Amazon S3 using DynamoDB TTL
- Automatically back up SAP HANA databases using Systems Manager and EventBridge
- Back up and archive mainframe data to Amazon S3 using BMC AMI Cloud Data
- Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue
- Configure model invocation logging in Amazon Bedrock by using AWS CloudFormation
- Convert and unpack EBCDIC data to ASCII on AWS by using Python
- Convert VARCHAR2(1) data type for Oracle to Boolean data type for Amazon Aurora PostgreSQL
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS
- Deliver DynamoDB records to Amazon S3 using Kinesis Data Streams and Firehose with AWS CDK
- Deploy a Lustre file system for high-performance data processing by using Terraform and DRA
- Estimate storage costs for an Amazon DynamoDB table
- Identify public Amazon S3 buckets in AWS Organizations by using Security Hub
- Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP
- Migrate an Oracle partitioned table to PostgreSQL by using AWS DMS
- Migrate data from Microsoft Azure Blob to Amazon S3 by using Rclone
- Migrate Oracle CLOB values to individual rows in PostgreSQL on AWS
- Migrate shared file systems in an AWS large migration
- Migrate small sets of data from on premises to Amazon S3 using AWS SFTP
- Monitor Amazon Aurora for instances without encryption
- Move mainframe files directly to Amazon S3 using Transfer Family
- Remove Amazon EC2 entries in the same AWS account from AWS Managed Microsoft AD by using AWS Lambda automation
- Run stateful workloads with persistent data storage by using Amazon EFS on Amazon EKS with AWS Fargate
- Successfully import an S3 bucket as an AWS CloudFormation stack
- Synchronize data between Amazon EFS file systems in different AWS Regions by using AWS DataSync

- View EBS snapshot details for your AWS account or organization

Developer tools

Topics

- DevOps
- Infrastructure
- Web & mobile apps

DevOps

Topics

- Automate AWS infrastructure operations by using Amazon Bedrock
- Automate CloudFront updates when load balancer endpoints change by using Terraform
- Automate Amazon CodeGuru reviews for AWS CDK Python applications by using GitHub Actions
- Automate AWS resource assessment
- Install SAP systems automatically by using open-source tools
- Automate AWS Service Catalog portfolio and product deployment by using AWS CDK
- Automate deletion of AWS CloudFormation stacks and associated resources
- Automate dynamic pipeline management for deploying hotfix solutions in Gitflow environments by using AWS Service Catalog and AWS CodePipeline
- Automate ingestion and visualization of Amazon MWAA custom metrics on Amazon Managed Grafana by using Terraform
- Streamline Amazon Lex bot development and deployment by using an automated workflow
- Automate stack set deployment by using AWS CodePipeline and AWS CodeBuild
- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK
- Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK
- Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9
- Build and push Docker images to Amazon ECR using GitHub Actions and Terraform
- Build and test iOS apps with AWS CodeCommit, AWS CodePipeline, and AWS Device Farm
- Check AWS CDK applications or CloudFormation templates for best practices by using cdk-nag rule packs
- Configure mutual TLS authentication for applications running on Amazon EKS
- Automate the creation of AppStream 2.0 resources using AWS CloudFormation
- Create a custom log parser for Amazon ECS using a Firelens log router
- Create a pipeline and AMI using CodePipeline and HashiCorp Packer
- Create dynamic CI pipelines for Java and Python projects automatically
- Deploy CloudWatch Synthetics canaries by using Terraform
- Deploy a CI/CD pipeline for Java microservices on Amazon ECS

- Deploy a ChatOps solution to manage SAST scan results by using Amazon Q Developer in chat applications custom actions and AWS CloudFormation
- Deploy agentic systems on Amazon Bedrock with the CrewAI framework by using Terraform
- Deploy an AWS Glue job with an AWS CodePipeline CI/CD pipeline
- Deploy an Amazon EKS cluster from AWS Cloud9 using an EC2 instance profile
- Deploy code in multiple AWS Regions using AWS CodePipeline, AWS CodeCommit, and AWS CodeBuild
- Deploy artifact updates to on-premises Amazon EC2 instances using AWS CodePipeline
- Deploy workloads from Azure DevOps pipelines to private Amazon EKS clusters
- Execute Amazon Redshift SQL queries by using Terraform
- Export AWS Backup reports from across an organization in AWS Organizations as a CSV file
- Export tags for a list of Amazon EC2 instances to a CSV file
- Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere
- Give SageMaker notebook instances temporary access to a CodeCommit repository in another AWS account
- Implement a GitHub Flow branching strategy for multi-account DevOps environments
- Implement a Gitflow branching strategy for multi-account DevOps environments
- Implement a Trunk branching strategy for multi-account DevOps environments
- Implement centralized custom Checkov scanning to enforce policy before deploying AWS infrastructure
- Implement AI-powered Kubernetes diagnostics and troubleshooting with K8sGPT and Amazon Bedrock integration
- Automatically detect changes and initiate different CodePipeline pipelines for a monorepo in CodeCommit
- Integrate a Bitbucket repository with AWS Amplify using AWS CloudFormation
- Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function
- Manage Multi-AZ failover for EMR clusters by using Application Recovery Controller
- Manage blue/green deployments of microservices to multiple accounts and Regions by using AWS code services and AWS KMS multi-Region keys

- Monitor Amazon ECR repositories for wildcard permissions using AWS CloudFormation and AWS Config
- Optimize multi-account serverless deployments by using the AWS CDK and GitHub Actions workflows
- Provision AWS Service Catalog products based on AWS CloudFormation templates by using GitHub Actions
- Provision least-privilege IAM roles by deploying a role vending machine solution
- Publish Amazon CloudWatch metrics to a CSV file
- Remove Amazon EC2 entries across AWS accounts from AWS Managed Microsoft AD by using AWS Lambda automation
- Remove Amazon EC2 entries in the same AWS account from AWS Managed Microsoft AD by using AWS Lambda automation
- Run unit tests for Python ETL jobs in AWS Glue using the pytest framework
- Set up a Helm v3 chart repository in Amazon S3
- Set up a CI/CD pipeline by using AWS CodePipeline and AWS CDK
- Set up centralized logging at enterprise scale by using Terraform
- Set up end-to-end encryption for applications on Amazon EKS using cert-manager and Let's Encrypt
- Simplify Amazon EKS multi-tenant application deployment by using Flux
- Subscribe multiple email endpoints to an SNS topic by using a custom resource
- Coordinate resource dependency and task execution by using the AWS Fargate WaitCondition hook construct
- Use third-party Git source repositories in AWS CodePipeline
- Create a CI/CD pipeline to validate Terraform configurations by using AWS CodePipeline
- More patterns

Automate AWS infrastructure operations by using Amazon Bedrock

Ishwar Chauthaiwale and Anand Bukkapatnam Tirumala, Amazon Web Services

Summary

In cloud native solutions, automating common infrastructure operations play a vital role in maintaining efficient, secure, and cost-effective environments. Manually handling operations is time-consuming and prone to human error. Additionally, team members with varying levels of AWS expertise need to perform these tasks while ensuring compliance with security protocols. This pattern demonstrates how to use Amazon Bedrock to automate common AWS infrastructure operations through natural language processing (NLP).

This pattern can help organizations to develop reusable, modular, and secure code for deploying generative AI-based infrastructure across multiple environments. Through its focus on infrastructure as code (IaC) and automation, it delivers key DevOps benefits including version control, consistent deployments, reduced errors, faster provisioning, and improved collaboration.

The pattern implements a secure architecture that enables teams to manage operations related to key AWS services including:

- Amazon Simple Storage Service (Amazon S3) bucket versioning management
- Amazon Relational Database Service (Amazon RDS) snapshot creation
- Amazon Elastic Compute Cloud (Amazon EC2) instance management

The architecture employs Amazon Virtual Private Cloud (Amazon VPC) endpoints and private networking for secure communication, with AWS Lambda functions operating as task executors within private subnets. Amazon S3 provides data management and implements comprehensive AWS Identity and Access Management (IAM) roles and permissions to ensure proper access controls. This solution doesn’t include a chat history feature, and the chat isn’t stored.

Prerequisites and limitations

- An active AWS account.
- Proper access control measures should be in place to help secure and control access. Examples of access control include using AWS Systems Manager, foundation models access, an IAM role for deployment, and service-based roles, disabling public access to Amazon S3 buckets, and setting up a dead-letter queue.

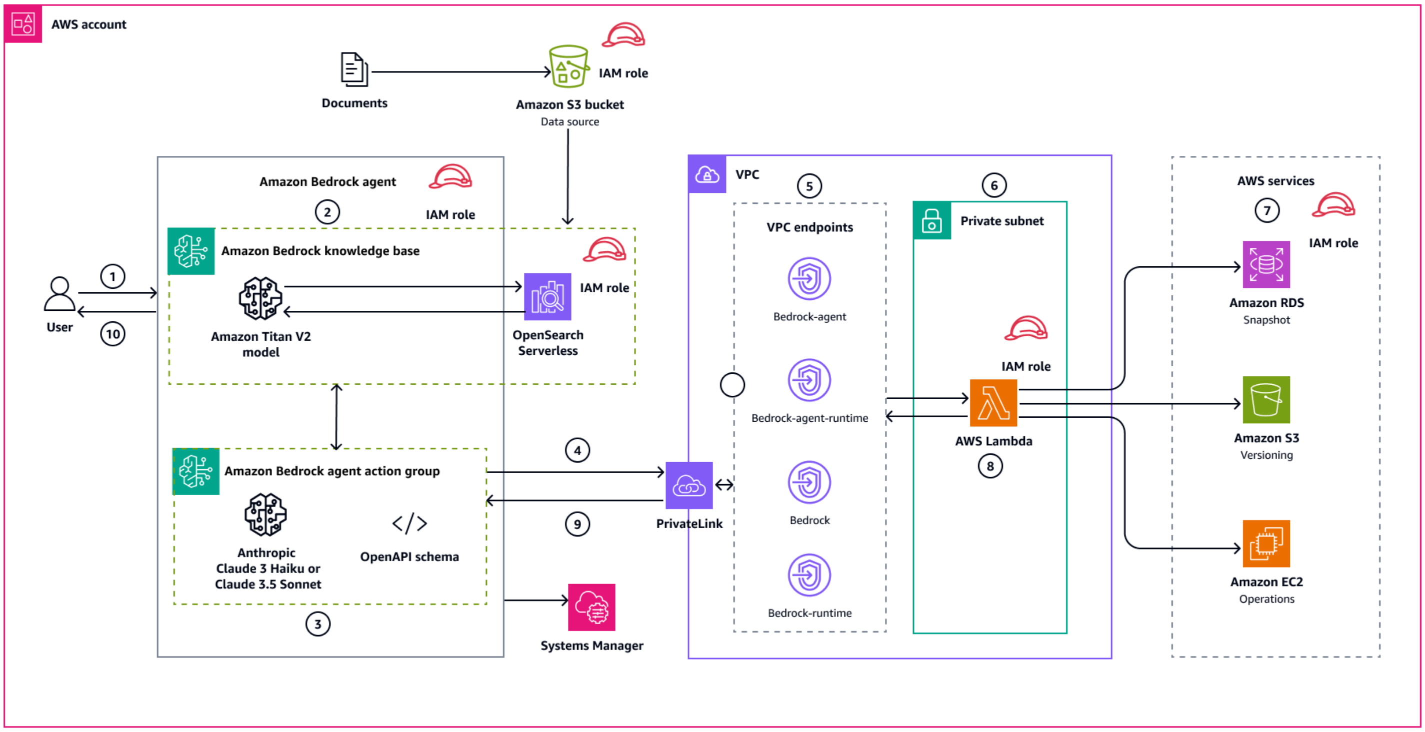
- An AWS Key Management Service (AWS KMS) customer managed key.
- AWS Command Line Interface (AWS CLI) version 2 or later, installed and configured on the deployment environment.
- Terraform AWS Provider version 4 or later installed and configured.
- Terraform version 1.5.7 or later installed and configured.
- Review and Define OpenAPI schemas for your agent's action groups in Amazon Bedrock to help protect against unauthorized access and maintain data integrity.
- Access enabled in your AWS account for the required Amazon Titan Text Embeddings v2 and either the Claude 3.5 Sonnet or Claude 3 Haiku foundation models. To avoid deployment failure, confirm that your target deployment AWS Region supports the required models.
- A configured virtual private cloud (VPC) that follows the AWS Well Architected Framework best practices.
- Completed review of the Amazon Responsible AI policy.

Product versions

- Amazon Titan Text Embeddings v2
- Anthropic Claude 3.5 Sonnet or Claude 3 Haiku
- Terraform AWS Provider version 4 or later
- Terraform version 1.5.7 or later

Architecture

The following diagram shows the workflow and architecture components for this pattern.



The solution architecture consists of multiple layers that work together to process natural language requests and execute corresponding AWS operations:

1. The user makes operations requests through the Amazon Bedrock chat console.
2. The chatbot uses Amazon Bedrock Knowledge Bases for request processing. It implements the Amazon Titan Text Embeddings v2 model for natural language processing.
3. If the user prompt includes an action request, the Amazon Bedrock action group uses either the Anthropic Claude 3 Haiku or the Claude 3.5 Sonnet model (depending on your choice) for execution logic and defines operations through an OpenAPI schema.
4. The action group reaches the Amazon VPC endpoints using AWS PrivateLink for secure service communication.
5. The AWS Lambda function is reached through Amazon VPC endpoints for Amazon Bedrock services.
6. The Lambda functions are the primary execution engine. Based on the request, the Lambda function calls the API to perform actions on the AWS services. The Lambda function also handles operation routing and execution.
7. The AWS services get the API request from the Lambda function and corresponding operations are performed.
8. The Lambda function computes an output payload that is understood by Amazon Bedrock.

9. This payload is sent to Amazon Bedrock by using PrivateLink for secure service communication. The large language model (LLM) used by Amazon Bedrock understands this payload and converts it into human understandable format.
10. The output is then shown to the user on the Amazon Bedrock chat console.

The solution enables the following primary operations:

- Amazon S3 – Enable bucket versioning for version control.
- Amazon RDS – Create database snapshots for backup.
- Amazon EC2 – List instances and control the start and stop of instances.

Tools

AWS services

- Amazon Bedrock is a fully managed service that makes high-performing foundation models (FMs) from leading AI startups and Amazon available for your use through a unified API.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon OpenSearch Serverless is an on-demand serverless configuration for Amazon OpenSearch Service.
- AWS PrivateLink helps you create unidirectional, private connections from your virtual private clouds (VPCs) to services outside of the VPC.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

- [AWS Systems Manager](#) helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.
- [Amazon Virtual Private Cloud (Amazon VPC)](#) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- [Git](#) is an open source, distributed version control system.
- [Terraform](#) is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub [aws-samples/infra-ops-orchestrator](#) repository.

Best practices

- Monitor Lambda execution logs regularly. For more information, see [Monitoring and troubleshooting Lambda functions](#). For more information about best practices, see [Best practices for working with AWS Lambda functions](#).
- Review security configurations periodically to ensure compliance with your organization's requirements. For more information, see [Security best practices](#).
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see [Grant least privilege](#) and [Security best practices](#) in the IAM documentation.

Epics

Deploy the solution

Task	Description	Skills required
Clone the repository.	To clone the repository on your local machine, run the following command:	AWS DevOps, DevOps engineer

Task	Description	Skills required
	`git clone "git@github.com:aws-samples/infra-ops-orchestrator.git"` `cd infra-ops-orchestrator`	
Edit the environment variables.	Edit the `terraform.tfvars` file. in the root directory of the cloned repository. Review the placeholders that are indicated by `[XXXXX]`, and update them according to your environment.	AWS DevOps, DevOps engineer

Create the infrastructure.	To create the infrastructure, run the following commands:
	`terraform init`
	`terraform plan`
	Review the execution plan carefully. If the planned changes are acceptable, then run the following command:
	`terraform apply --auto-approve`
		AWS DevOps, DevOps engineer

Access the solution

Task	Description	Skills required

Access the solution.	After successful deployment, follow these steps to use the chat-based interface:
	
	1. To access the Infrastructure Orchestrator Assistant, sign in to the AWS Managemen t Console using an [IAM role with Amazon Bedrock permissions](), and open the Amazon Bedrock console at [https://console.a ws.amazon.com/bedrock/](https://console.aws.amazon.com/bedrock/). Select **Agents** from the left navigation pane. Then, choose the Infrastructure Orchestrator Assistant in the **Agents** section.
	2. Make sure that target resources exist in your AWS environment for the following suggestions, and then try these example operations:
	  • Create a snapshot backup of an Amazon RDS instance by asking: 'Create a snapshot of RDS instance [instance-name]'
	  • Enable versioning on an Amazon S3 bucket by
		AWS DevOps, DevOps engineer

Task	Description	Skills required
	asking: 'Enable versioning for bucket [bucket-name]' • List Amazon EC2 instances by asking: 'List all EC2 instances' • Start or stop an Amazon EC2 instance by asking: 'Start EC2 instance [instance-id]' or 'Stop EC2 instance [instance-id]' Note: Replace the values in brackets with the actual resource names or IDs from your AWS environment.	

Clean up resources

Task	Description	Skills required

Delete the created resources.	To delete all infrastructure created by this pattern, run the following command:
	
	`terraform plan -destroy`
	
	Review the destruction plan carefully. If the planned deletions are acceptable, then run the following command:
	
	`terraform destroy`
		AWS DevOps, DevOps engineer

Task	Description	Skills required
	Note: This command will permanently delete all resources created by this pattern. The command will prompt for confirmat ion before removing any resources.	

Troubleshooting

Issue	Solution
Agent behavior	For information about this issue, see Test and troubleshoot agent behavior in the Amazon Bedrock documentation.
Lambda network issues	For information about these issues, see Troubleshoot networking issues in Lambda in the Lambda documentation.
IAM permissions	For information about these issues, see Troubleshoot IAM in the IAM documentation.

Related resources

- Creating a DB snapshot for a Single-AZ DB instance for Amazon RDS
- Define OpenAPI schemas for your agent's action groups in Amazon Bedrock
- Enabling versioning on buckets
- How Amazon Bedrock Agents works
- Retrieve data and generate AI responses with Amazon Bedrock Knowledge Bases
- Securely Access Services Over AWS PrivateLink
- Stop and start Amazon EC2 instances
- Use action groups to define actions for your agent to perform

Automate CloudFront updates when load balancer endpoints change by using Terraform

Tamilselvan P, Mohan Annam, and Naveen Suthar, Amazon Web Services

Summary

When users of Amazon Elastic Kubernetes Service (Amazon EKS) delete and re-install their ingress configuration through Helm charts, a new Application Load Balancer (ALB) is created. This creates a problem because Amazon CloudFront continues to reference the old ALB's DNS record. As a result, services destined to this endpoint will not be reachable. (For more details about this problematic workflow, see [Additional information](#).)

To solve this issue, this pattern describes using a custom AWS Lambda function that was developed with Python. This Lambda function automatically detects when a new ALB is created through Amazon EventBridge rules. Using the AWS SDK for Python (Boto3), the function then updates the CloudFront configuration with the new ALB's DNS address, ensuring that traffic is routed to the correct endpoint.

This automated solution maintains service continuity without additional routing or latency. The process helps to ensure that CloudFront always references the correct ALB DNS endpoint, even when the underlying infrastructure changes.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A sample web application for testing and validation that is deployed on Amazon EKS by using Helm. For more information, see Deploy applications with Helm on Amazon EKS in the Amazon EKS documentation.
- Configure CloudFront to route calls to an ALB that is created by a Helm ingress controller. For more information, see Install AWS Load Balancer Controller with Helm in the Amazon EKS documentation and Restrict access to Application Load Balancers in the CloudFront documentation.
- Terraform installed and configured in a local workspace.

Limitations

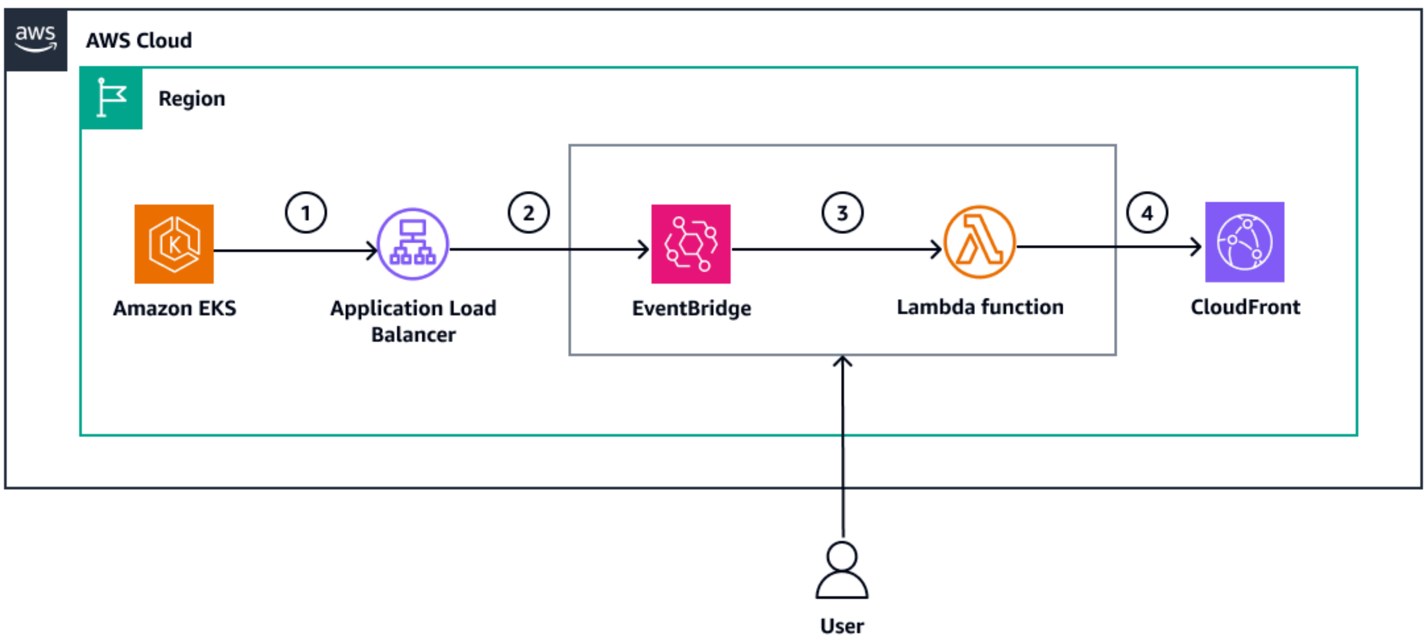
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- Terraform version 1.0.0 or later
- Terraform AWS Provider version 4.20 or later

Architecture

The following diagram shows the workflow and architecture components for this pattern.



This solution performs the following steps:

1. The Amazon EKS ingress controller creates a new Application Load Balancer (ALB) whenever there is a Helm restart or deployment.
2. EventBridge looks for ALB creation events.
3. The ALB creation event triggers the Lambda function.
4. The Lambda function has been deployed based on python 3.9 and uses boto3 API to call AWS services. The Lambda function updates the CloudFront entry with the latest load balancer DNS name, which is received from create load balancer events.

Tools

AWS services

- Amazon CloudFront speeds up distribution of your web content by delivering it through a worldwide network of data centers, which lowers latency and improves performance.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services.

Other tools

- Python is a general-purpose computer programming language.
- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub aws-cloudfront-automation-terraform-samples repository.

Epics

Set up local workstation

Task	Description	Skills required
Set up and configure the Git CLI.	To install and configure the Git command line interface (CLI) in your local workstati	DevOps engineer

Task	Description	Skills required
	on, follow the Getting Started – Installing Git instructions in the Git documentation.	

Task	Description	Skills required
Create the project folder and add the files.	1. Go to the pattern's [GitHub repository](), and choose the **Code** button. 2. On the **Clone** dialog, choose the **HTTPS** tab. In **Clone using the web URL**, copy the URL that's shown. 3. Create a folder on your local machine. Name it with your project name. 4. Open a terminal on your local machine, and navigate to this folder. 5. To clone this pattern's git repository, run the following command: `git clone https://github.com/aws-samples/aws-cloudfront-automation-terraform-samples` 6. After the repository has been cloned, use the following command to go to the cloned directory: `cd <directory name>/cloudfront-update` Open this project in an Integrated Development Environment (IDE) of your choice.	DevOps engineer

Provision the target architecture using the Terraform configuration

Task	Description	Skills required

Deploy the solution.	To deploy resources in the target AWS account, use the following steps:
	1. Go to the `cloudfront-update` folder.
	2. Update the `terraform .tfvars` file with the `cloudfront_distribution_id` .
	3. To set the AWS Region for your AWS profile, run the following command:
	`export AWS_REGION={{ REGION }}`
	4. To initialize Terraform, run the following command:
	`terraform init`
	5. To validate Terraform, run the following command:
	`terraform validate`
	6. To create a Terraform execution plan, run the following command:
	`terraform plan`
		DevOps engineer

Task	Description	Skills required
	7. To apply the actions from `terraform plan`, run the following command: `terraform apply`	

Verify the deployment

Task	Description	Skills required
Validate the deployment.	1. Sign in to the AWS Management Console and open the Amazon CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home. 2. In the left navigation pane, choose **Distributions** and then open the CloudFront distribution. 3. On the **Origins** tab, verify that the origin name and origin mapping have the updated ALB DNS record.	DevOps engineer

Clean up infrastructure

Task	Description	Skills required
Clean up the infrastructure.	To clean up the infrastructure that you created earlier, use the following steps:	DevOps engineer

Task	Description	Skills required
	1. Run the following command: `terraform destroy` 2. To confirm the destroy command, enter `yes`.	

Troubleshooting

Issue	Solution
Error validating provider credentials	When you run the Terraform `apply` or `destroy` commands from your local machine, you might encounter an error similar to the following: `Error: configuring Terraform AWS Provider: error validating provider credentials: error calling sts:GetCallerIdentity: operation error STS: GetCallerIdentity, https response error StatusCode: 403, RequestID: 123456a9-fbc1-40ed-b8d8-513d0133ba7f, api error InvalidClientTokenId: The security token included in the request is invalid.` This error is caused by the expiration of the security token for the credentials used in your local machine's configuration. To resolve the error, see [Set and view configuration settings](#) in the AWS Command Line Interface (AWS CLI) documentation.

Related resources

AWS resources

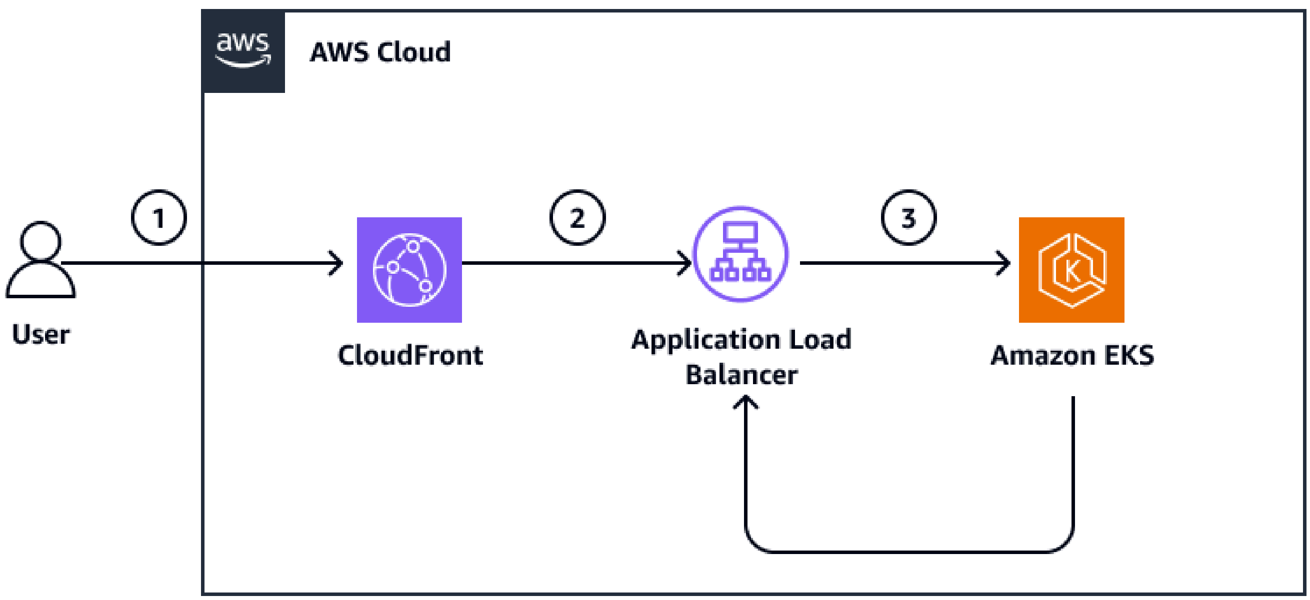
- Restrict access to Application Load Balancers
- Route internet traffic with AWS Load Balancer Controller

Terraform documentation

- AWS Provider
- Install Terraform
- Remote State

Additional information

Problematic workflow



The diagram shows the following workflow:

1. When the user accesses the application, the call goes to CloudFront.
2. CloudFront routes the calls to the respective Application Load Balancer (ALB).
3. The ALB includes the target IP addresses which are the application pod's IP addresses. From there, the ALB provides the expected results to the user.

However, this workflow demonstrates a problem. The application deployments are happening through Helm charts. Whenever there is a deployment or if someone restarts Helm, the respective ingress is also re-created. As a result, the external load balancer controller re-creates the ALB. Also, during each re-creation, the ALB is re-created with a different DNS name. Because of this, CloudFront will have a stale entry in the origin settings. Because of this stale entry, the application will not be reachable for the user. This issue results in downtime for users.

Alternative solution

Another possible solution is to create an external DNS for the ALB and then point it to the Amazon Route 53 private hosted zone endpoint in CloudFront. However, this approach adds another hop in the application flow, which might cause application latency. This pattern's Lambda function solution doesn't disrupt current flow.

Automate Amazon CodeGuru reviews for AWS CDK Python applications by using GitHub Actions

Vanitha Dontireddy and Sarat Chandra Pothula, Amazon Web Services

Summary

This pattern showcases the integration of Amazon CodeGuru automated code reviews for AWS Cloud Development Kit (AWS CDK) Python applications, orchestrated through GitHub Actions. The solution deploys a serverless architecture defined in AWS CDK Python. By automating expert code analysis within the development pipeline, this approach can do the following for AWS CDK Python projects:

- Enhance code quality.
- Streamline workflows.
- Maximize the benefits of serverless computing.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2.9.11 or later, installed and configured.
- An active GitHub account and a GitHub repository with read and write workflow permissions and creation of pull requests (PR) by GitHub Actions to ensure the PR workflow operates correctly.
- An OpenID Connect (OIDC) role in GitHub Actions to deploy the solution in the AWS account. To create the role, use the AWS CDK construct.

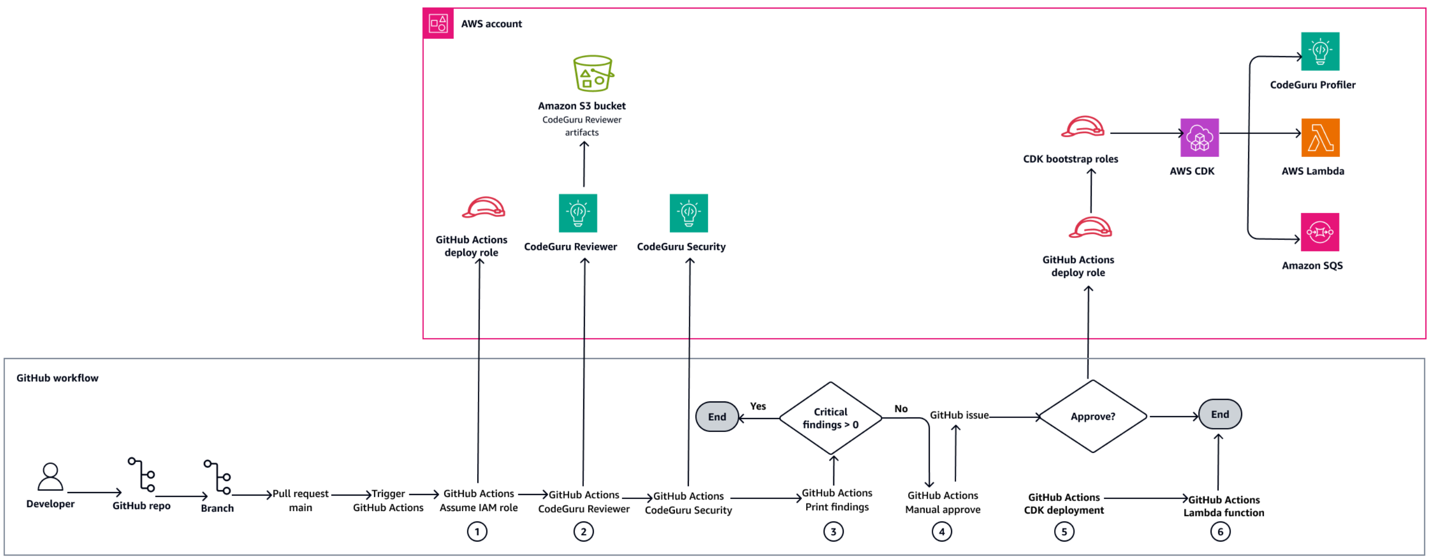
Limitations

- Amazon CodeGuru Profiler supports applications written in all Java virtual machine (JVM) languages (such as Scala and Kotlin) and runtimes and Python 3.6 or later.
- Amazon CodeGuru Reviewer supports associations with Java and Python code repositories only from the following source providers: AWS CodeCommit, Bitbucket, GitHub, GitHub Enterprise Cloud, and GitHub Enterprise Server. In addition, Amazon Simple Storage Service (Amazon S3) repositories are only supported through GitHub Actions.

- There isn’t an automated way to print the findings during the continuous integration and continuous deployment (CI/CD) pipeline. Instead, this pattern uses GitHub Actions as an alternative method to handle and display the findings.
- Some AWS services aren’t available in all AWS Regions. For Region availability, see [AWS services by Region](). For specific endpoints, see [Service endpoints and quotas](), and choose the link for the service.

Architecture

The following diagram shows the architecture for this solution.



As shown in the diagram, when a developer creates a pull request (PR) for review, GitHub Actions triggers the following steps:

1. IAM role assumption – The pipeline uses the IAM role that’s specified in GitHub Secrets to perform deployment tasks.
2. Code analysis
   - CodeGuru Reviewer analyzes the code stored in the Amazon S3 bucket. It identifies defects and provides recommendations for fixes and optimizations.
   - CodeGuru Security scans for policy violations and vulnerabilities.
3. Findings review
   - The pipeline prints a link to the findings dashboard in the console output.
   - If critical findings are detected, the pipeline fails immediately.

- For high, normal, or low severity findings, the pipeline continues to the next step.

4. PR approval
   - A reviewer must manually approve the PR.
   - If the PR is denied, the pipeline fails and halts further deployment steps.
5. CDK deployment – Upon PR approval, the CDK deployment process begins. It sets up the following AWS services and resources:
   - CodeGuru Profiler
   - AWS Lambda function
   - Amazon Simple Queue Service (Amazon SQS) queue
6. Profiling data generation – To generate sufficient profiling data for CodeGuru Profiler:
   - The pipeline invokes the Lambda function multiple times by sending messages to the Amazon SQS queue periodically.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- CDK Toolkit is a command line cloud development kit that helps you interact with your AWS CDK app.
- Amazon CodeGuru Profiler collects runtime performance data from your live applications, and provides recommendations that can help you fine-tune your application performance.
- Amazon CodeGuru Reviewer uses program analysis and machine learning to detect potential defects that are difficult for developers to find. Then, CodeGuru Profiler offers suggestions for improving your Java and Python code.
- Amazon CodeGuru Security is a static application security tool that uses machine learning to detect security policy violations and vulnerabilities. It provides suggestions for addressing security risks and generates metrics so you can track the security posture of your applications.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that's tightly integrated with GitHub repositories. You can use GitHub Actions to automate your build, test, and deployment pipeline.

Code repository

The code for this pattern is available in the GitHub amazon-codeguru-suite-cdk-python repository.

Best practices

- Adhere to the Best practices for developing and deploying cloud infrastructure with the AWS CDK.
- Follow Security best practices in IAM when using AWS services in GitHub Actions workflows, including:
  - Do not store credentials in your repository code.
  - Assume an IAM role to receive temporary credentials, and use temporary credentials when possible.
  - Grant least privilege to the IAM role used in GitHub Actions workflows. Grant only the permissions that are required to perform the actions in your GitHub Actions workflows.
  - Monitor the activity of the IAM role that's used in GitHub Actions workflows.
  - Periodically rotate any long-term credentials that you use.

Epics

Set up your environment

Task	Description	Skills required

Set up AWS credentials.	To export the variables that define the AWS account and AWS Region where you're deploying the stack, run the following commands:
	`export CDK_DEFAULT_ACCOUNT=<12-digit AWS account number>`
	`export CDK_DEFAULT_REGION=<AWS Region>`
	The AWS credentials for the AWS CDK are provided through environment variables.
		AWS DevOps, DevOps engineer

Clone the repository.	To clone the repository on your local machine, run the following command: `git clone https://github.com/aws-samples/amazon-codeguru-suite-cdk-python.git`	AWS DevOps, DevOps engineer
Install the CDK Toolkit.	To confirm that the CDK Toolkit is installed and to check the version, run the following command:	AWS DevOps, DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>cdk --version</code></pre>If the CDK Toolkit version is earlier than 2.27.0, enter the following command to update it to version 2.27.0:<pre><code>npm install -g aws-cdk@2
.27.0</code></pre>If the CDK Toolkit is not installed, run the following command to install it:<pre><code>npm install -g aws-cdk@2
.27.0 --force</code></pre></td><td></td></tr>
<tr><td>Install the required dependencies.</td><td>To install the required project dependencies, run the following command:<pre><code>python -m pip install --
upgrade pip
pip install -r requireme
nts.txt</code></pre></td><td>AWS DevOps, DevOps engineer</td></tr>
</table>

Task	Description	Skills required

Bootstrap the CDK environme nt.	To [bootstrap](#) an AWS CDK environment, run the following commands:
	`npm install`
	`npm run cdk bootstrap "aws://${ACCOUNT_NUMBER}/${AWS_REGION}"`
	After you successfully bootstrap the environment, the following output should be displayed:
	`#  Bootstrapping environment aws://{account}/{region}...`
	`#  Environment aws://{account}/{region} bootstrapped`
		AWS DevOps, DevOps engineer

Deploy the CDK app

Task	Description	Skills required
Synthesize the AWS CDK app.	To synthesize an AWS CDK app, run the following command: `cdk synth` For more information about this command, see [cdk synthesize](#) in the AWS CDK documentation.	AWS DevOps, DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy the resources.</td><td>To deploy the resources, run the following command:<pre><code>cdk deploy --require-approval never</code></pre>Note
The <code>--require-approval never</code> flag means that the CDK will approve and execute all changes automatically. This includes changes that the CDK would normally flag as needing manual review (such as IAM policy changes or removal of resources). Make sure that your CDK code and CI/CD pipeline are well-tested and secure before you use the <code>--require-approval never</code> flag in production environments.</td><td>AWS DevOps, DevOps engineer</td></tr>
</table>

Create GitHub secrets and personal access token

Task	Description	Skills required
Create the required secrets in GitHub.	To allow GitHub Actions workflows to access AWS resources securely without exposing sensitive information in your repository's code, create secrets. To create the secrets in GitHub for `ROLE_TO_ASSUME`, `CodeGuruReviewArtifactBucketName`, and `AWS_ACCOUNT_ID`, follow the instructions in [Creating secrets for a repository](#) in the GitHub Actions documentation. Following is more information about the variables: • `AWS_ACCOUNT_ID` – The AWS account ID where the pipeline is executed. • `CodeGuruReviewArtifactBucketName` – The name of the S3 bucket where CodeGuru Reviewer artifacts are stored. This pattern uses the bucket name `codeguru-reviewer-build-artifacts-<ACCOUNT_ID>-<REGION>`.	AWS DevOps, DevOps engineer

Task	Description	Skills required
	• `AWS_REGION` – The AWS Region where the resources are located. • `ROLE_TO_ASSUME` – The name of the IAM role that the pipeline assumes. This pattern uses the role name `githubActionsDeployRole`.	
Create a GitHub personal access token.	To set up a secure way for your GitHub Actions workflows to authenticate and interact with GitHub, do the following: 1. To create a GitHub personal access token that has read and write access to your repository, follow the instructions in [Managing your personal access tokens](#) in the GitHub documentation. 2. To save this token as a repository secret for GitHub Actions, follow the instructions in [Creating secrets for a repository](#) in the GitHub Actions documentation.	AWS DevOps, DevOps engineer

Clean up

Task	Description	Skills required
Clean up resources.	To clean up your AWS CDK Python app, run the following command: `cdk destroy --all`	DevOps engineer

Troubleshooting

Issue	Solution
Display link to the dashboard findings.	There is no way to print the findings during the CI/CD pipeline. Instead, this pattern uses GitHub Actions as an alternative method to handle and display the findings.

Related resources

AWS resources

- AWS Cloud Development Kit
- Amazon CodeGuru Documentation
- Amazon S3
- AWS Identity and Access Management
- Amazon Simple Queue Service
- What is AWS Lambda?

GitHub documentation

- Configuring OpenID Connect in Amazon Web Services
- GitHub Actions

- Reusing workflows
- Triggering a workflow

Automate AWS resource assessment

Naveen Suthar, Arun Bagal, Manish Garg, and Sandeep Gawande, Amazon Web Services

Summary

This pattern describes an automated approach for setting up resource assessment capabilities by using the AWS Cloud Development Kit (AWS CDK). By using this pattern, operations teams gather resource auditing details in an automated manner and view the details of all resources deployed in an AWS account on a single dashboard. This is helpful in the following use cases:

- Identifying infrastructure as code (IaC) tools and isolating resources created by different IaC solutions such as HashiCorp Terraform, AWS CloudFormation, AWS CDK, and AWS Command Line Interface (AWS CLI)
- Fetching resource-auditing information

This solution will also help the leadership team obtain insights about the resources and activities in an AWS account from a single dashboard.

Note: Amazon QuickSight is a paid service. Before running it to analyze data and create a dashboard, review the Amazon QuickSight pricing.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Identity and Access Management (IAM) roles and permissions with access to provision resources
- An Amazon QuickSight account created with access to Amazon Simple Storage Service (Amazon S3) and Amazon Athena
- AWS CDK version 2.55.1 or later installed
- Python version 3.9 or later installed

Limitations

- This solution is deployed to a single AWS account.
- The solution will not track the events that happened before its deployment unless AWS CloudTrail was already set up and storing data in an S3 bucket.

Product versions

- AWS CDK version 2.55.1 or later
- Python version 3.9 or later

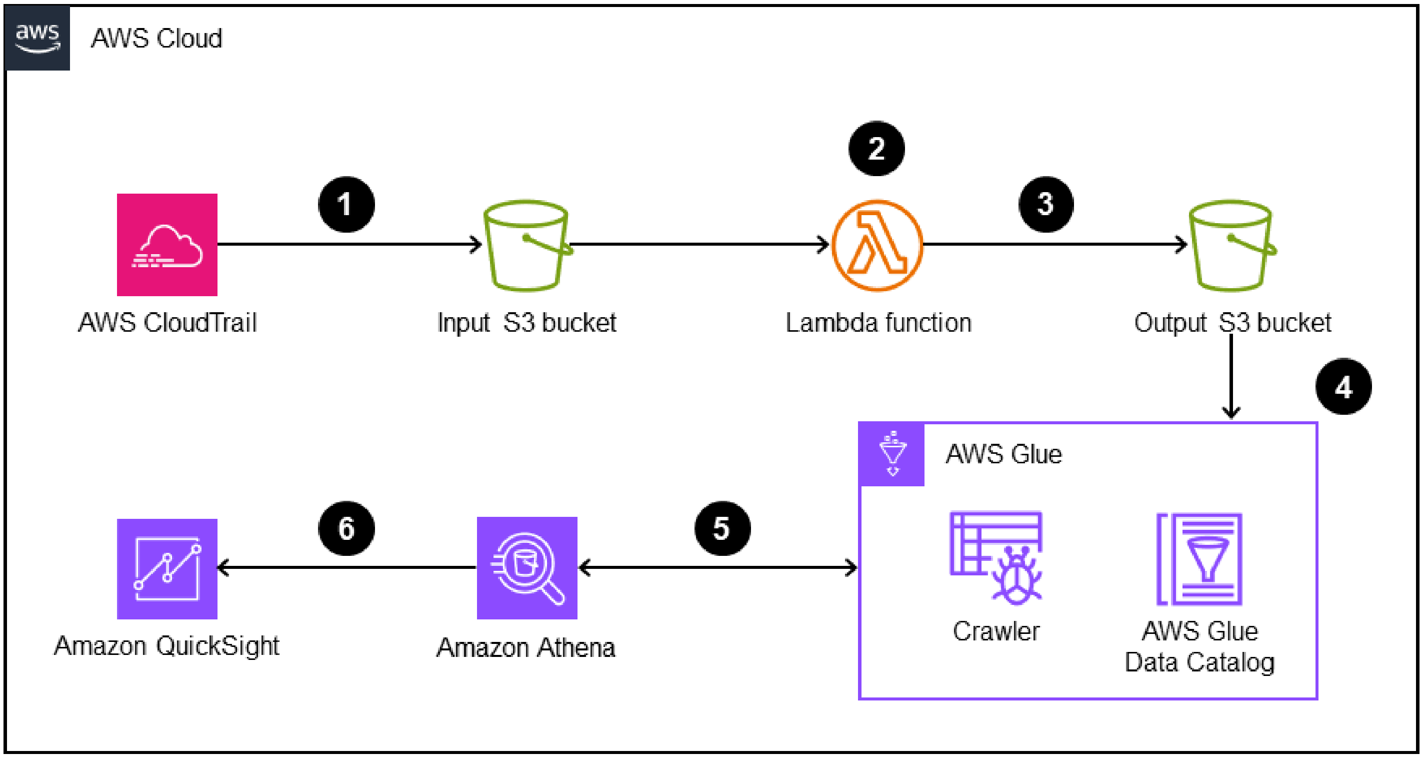
Architecture

Target technology stack

- Amazon Athena
- AWS CloudTrail
- AWS Glue
- AWS Lambda
- Amazon QuickSight
- Amazon S3

Target architecture

The AWS CDK code will deploy all the resources that are required to set up resource-assessment capabilities in an AWS account. The following diagram shows the process of sending CloudTrail logs to AWS Glue, Amazon Athena, and QuickSight.



1. CloudTrail sends logs to an S3 bucket for storage.
2. An event notification invokes a Lambda function that processes the logs and generates filtered data.
3. The filtered data is stored in another S3 bucket.
4. An AWS Glue crawler is set up on the filtered data that is in the S3 bucket to create a schema in the AWS Glue Data Catalog table.
5. The filtered data is ready to be queried by Amazon Athena.
6. The queried data is accessed by QuickSight for visualization.

Automation and scale

- This solution can be scaled from one AWS account to multiple AWS accounts if there is an organization-wide CloudTrail trail in AWS Organizations. By deploying CloudTrail at the organizational level, you can also use this solution to fetch resource-auditing details for all the required resources.
- This pattern uses AWS serverless resources to deploy the solution.

Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- AWS CloudTrail helps you audit the governance, compliance, and operational risk of your AWS account.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams. This pattern uses an AWS Glue crawler and an AWS Glue Data Catalog table.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data in a single dashboard.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code repository

The code for this pattern is available in the GitHub infrastructure-assessment-iac-automation repository.

The code repository contains the following files and folders:

- `lib` folder – The AWS CDK construct Python files used to create AWS resources
- `src/lambda_code` – The Python code that is run in the Lambda function
- `requirements.txt` – The list of all Python dependencies that must be installed
- `cdk.json` – The input file to provide values required to spin up resources

Best practices

Set up monitoring and alerting for the Lambda function. For more information, see Monitoring and troubleshooting Lambda functions. For general best practices when working with Lambda functions, see the AWS documentation.

Epics

Set up your environment

Task	Description	Skills required
Clone the repo on your local machine.	To clone the repository, run the command `git clone https://github.com/aws-samples/infrastructure-assessment-iac-automation.git .`	AWS DevOps, DevOps engineer

Set up the Python virtual environment and install required dependencies.	To set up the Python virtual environment, run the following commands.
	`cd infrastructure-assessment-iac-automation`
	`python3 -m venv .venv`
	`source .venv/bin/activate`
	To set up the required dependencies, run the command `pip install -r requirements.txt .`
		AWS DevOps, DevOps engineer

Set up the AWS CDK environment and synthesize the AWS CDK code.	1. To set up the AWS CDK environment in your AWS account, run the command	AWS DevOps, DevOps engineer

Task	Description	Skills required
	`cdk bootstrap aws://ACCOUNT-NUMBER/REGION .` 2. To convert the code to an AWS CloudFormation stack configuration, run the command `cdk synth`.	

Set up AWS credentials on your local machine

Task	Description	Skills required

Export variables for the account and Region where the stack will be deployed.	To provide AWS credentials for AWS CDK by using environment variables, run the following commands.
	`export CDK_DEFAULT_ACCOUNT=<12 Digit AWS Account Number>`
	`export CDK_DEFAULT_REGION=<region>`
		AWS DevOps, DevOps engineer

Set up the AWS CLI profile.	To set up the AWS CLI profile for the account, follow the instructions in the [AWS documentation].	AWS DevOps, DevOps engineer

Configure and deploy the resource-assessment tool

Task	Description	Skills required
Deploy resources in the account.	To deploy resources in the AWS account by using AWS CDK, do the following: 1. In the root of the cloned repository, in the `cdk.json` file, provide inputs for the following parameters: • `s3_context` • `ct_context` • `kms_context` • `lambda_context` • `glue_context` • `qs_context` These values define resource configurations and nomenclature. Default values are set and can be changed if required. **Note** To avoid an error saying that the S3 bucket already exists, make sure to provide unique names for `s3_contex`	AWS DevOps

Task	Description	Skills required
	`t` in the `ct` and `output` sections. 2. To deploy resources, run the command `cdk deploy`. The `cdk deploy` command creates a CloudTrail resource to log events and save the log file in the input S3 bucket. The trail's log files will be processed by the Lambda function. The filtered results are stored in the output S3 bucket and are ready to be consumed by Amazon Athena and Amazon QuickSight.	

Task	Description	Skills required
Run the AWS Glue crawler and create the Data Catalog table.	An [AWS Glue crawler](#) is used to keep the data schema dynamic. The solution creates and updates partitions in the [AWS Glue Data Catalog table](#) by running the crawler periodically as defined by the AWS Glue crawler scheduler . After the data is available in the output S3 bucket, use the following steps to run the AWS Glue crawler and create the Data Catalog table schema for testing: 1. Sign in to the AWS Management Console and navigate to the AWS Glue console. 2. In the navigation pane, under **Data Catalog**, choose **Crawlers**. 3. Select the `iac-tool-qa-resource-iac-json-crawler` crawler. 4. Run the crawler. 5. After the crawler runs successfully, it creates an AWS Glue Data Catalog table. AWS QuickSight will use the table to visualize the data.	AWS DevOps, DevOps engineer

Task	Description	Skills required
	Note The AWS CDK code configures the AWS Glue crawler to run at a particular time, but you can also run it on demand.	
Deploy the QuickSight construct.	1. To deploy the QuickSight construct, uncomment the code between `#QuickSight setup - start` and `#QuickSight setup – ends` in `resource_iac_tool_stack.py` . 2. After you uncomment , run the `cdk deploy` command to create `QuickSight DataSource` and `QuickSight DataSet` in the QuickSight account.	AWS DevOps, DevOps engineer

Task	Description	Skills required
Create the QuickSight dashboard.	To create the example QuickSight dashboard and analysis, do the following: 1. Navigate to the QuickSight console and select the AWS Region where resources are deployed. 2. In the navigation pane, choose **Datasets**, and validate that a dataset named `ct-operations-iac-ds` has been created in the Amazon QuickSight dataset. If you don't see the dataset, redeploy the QuickSight construct. 3. Select the `ct-operations-iac-ds` dataset, and choose **USE IN ANALYSIS**. 4. Select the default sheet. 5. Select the respective columns from the field list on the left side. 6. After selecting the required columns, select the appropriate visual type to view the data. For more information, see [Starting an analysis in](#)	AWS DevOps, DevOps engineer

Task	Description	Skills required
	Amazon QuickSight and Visual types in Amazon QuickSight.	

Clean up all AWS resources in the solution

Task	Description	Skills required
Remove the AWS resources.	1. To remove AWS resources deployed by the solution, run the command `cdk destroy`. 2. Delete all objects from the two S3 buckets, and then remove the buckets. For more information, see Deleting a bucket.	AWS DevOps, DevOps engineer

Set up additional features on top of the AWS resource-assessment tool automation

Task	Description	Skills required
Monitor and clean up manually created resources.	(Optional) If your organizat ion has compliance requireme nts to create resources using IaC tools, you can achieve compliance by using AWS resource-assessment tool automation to fetch manually provisioned resources. You can also use the tool to import the resources to an IaC tool or to re-create them. To	AWS DevOps, DevOps engineer

Task	Description	Skills required
	monitor manually provision ed resources, perform the following high-level tasks: 1. Deploy AWS resource-assessment tool automatio n. 2. Set up a Lambda function to query the Athena tables on a daily basis, find the relevant data about manually provisioned resources, and export it to a comma-separated values (CSV) file. 3. After the Lambda function runs, a notification with the required data can be sent to respective stakeholders. 4. For longer retention, the .csv file can be stored in the S3 bucket. 5. Based on the information in the .csv file, delete the manually created resources or import them to an existing IaC solution.	

Troubleshooting

Issue	Solution
AWS CDK returns errors.	For help with AWS CDK issues, see Troublesh ooting common AWS CDK issues.

Related resources

- Building Lambda functions with Python
- Get started with AWS CDK
- Working with AWS CDK in Python
- Creating a CloudTrail log trail
- Get Started with Amazon QuickSight

Additional information

Multiple accounts

To set up the AWS CLI credential for multiple accounts, use AWS profiles. For more information, see the *Configure multiple profiles* section in Set up the AWS CLI.

AWS CDK commands

When working with AWS CDK, keep in mind the following useful commands:

- Lists all stacks in the app

```
cdk ls
```

- Emits the synthesized AWS CloudFormation template

```
cdk synth
```

- Deploys the stack to your default AWS account and Region

```
cdk deploy
```

- Compares the deployed stack with the current state

```
cdk diff
```

- Opens the AWS CDK documentation

```
cdk docs
```

Install SAP systems automatically by using open-source tools

Guilherme Sesterheim, Amazon Web Services

Summary

This pattern shows how to automate SAP systems installation by using open-source tools to create the following resources:

- An SAP S/4HANA 1909 database
- An SAP ABAP Central Services (ASCS) instance
- An SAP Primary Application Server (PAS) instance

HashiCorp Terraform creates the SAP system's infrastructure and Ansible configures the operating system (OS) and installs SAP applications. Jenkins runs the installation.

This setup turns SAP systems installation into a repeatable process, which can help increase deployment efficiency and quality.

Note

The example code provided in this pattern works for both high-availability (HA) systems and non-HA systems.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Simple Storage Service (Amazon S3) bucket that contains all of your SAP media files
- An AWS Identity and Access Management (IAM) principal with an access key and secret key, and that has the following permissions:
  - **Read only permissions:** Amazon Route 53, AWS Key Management Service (AWS KMS)
  - **Read and write permissions:** Amazon S3, Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic File System (Amazon EFS), IAM, Amazon CloudWatch, Amazon DynamoDB
- A Route 53 private hosted zone

- A subscription to the Red Hat Enterprise Linux for SAP with HA and Update Services 8.2 Amazon Machine Image (AMI) in Amazon Marketplace
- An AWS KMS customer managed key
- A Secure Shell (SSH) key pair
- An Amazon EC2 security group that allows SSH connection on port 22 from the hostname where you install Jenkins (the hostname is most likely **localhost**)
- Vagrant by HashiCorp installed and configured
- VirtualBox by Oracle installed and configured
- Familiarity with Git, Terraform, Ansible, and Jenkins

Limitations

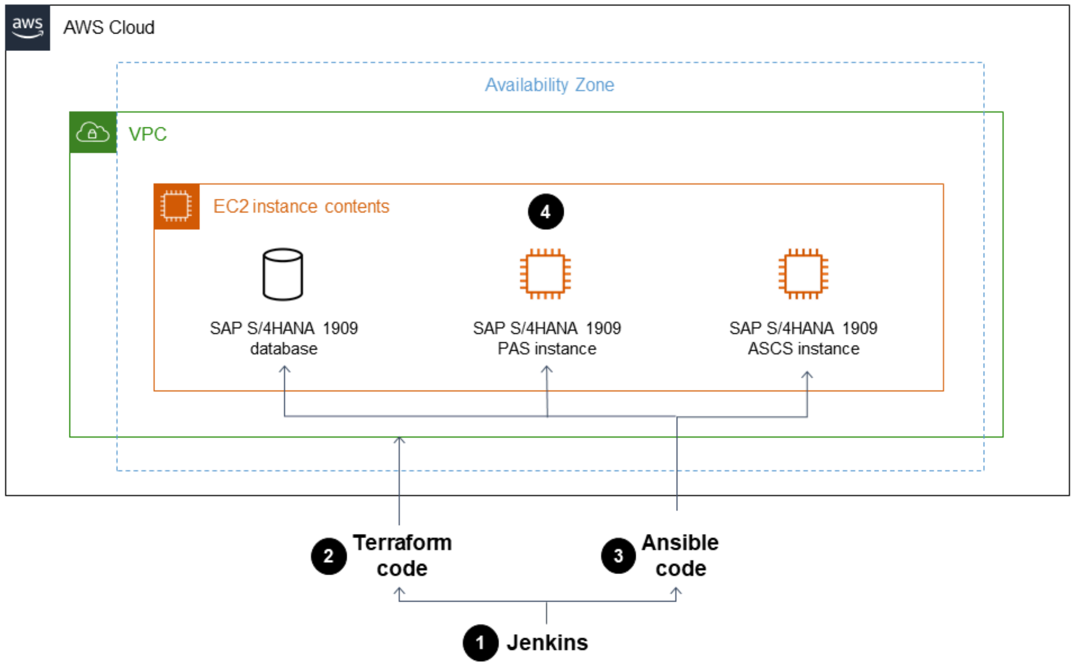
- Only SAP S/4HANA 1909 is fully tested for this specific scenario. The example Ansible code in this pattern requires modification if you use another version of SAP HANA.
- The example procedure in this pattern works for Mac OS and Linux operating systems. Some of the commands can be run only in Unix-based terminals. However, you can achieve a similar result by using different commands and a Windows OS.

Product versions

- SAP S/4HANA 1909
- Red Hat Enterprise Linux (RHEL) 8.2 or higher versions

Architecture

The following diagram shows an example workflow that uses open-source tools to automate SAP systems installation in an AWS account:



The diagram shows the following workflow:

1. Jenkins orchestrates running the SAP system installation by running Terraform and Ansible code.
2. Terraform code builds the SAP system's infrastructure.
3. Ansible code configures the OS and installs SAP applications.
4. An SAP S/4HANA 1909 database, an ASCS instance, and PAS instance that include all defined prerequisites are installed on an Amazon EC2 instance.

> **Note**
>
> The example setup in this pattern automatically creates an Amazon S3 bucket in your AWS account to store the Terraform state file.

Technology stack

- Terraform

- Ansible
- Jenkins
- An SAP S/4HANA 1909 database
- An SAP ASCS instance
- An SAP PAS instance
- Amazon EC2

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need, and quickly scale them up or down.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to protect your data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- HashiCorp Terraform is a command-line interface application that helps you use code to provision and manage cloud infrastructure and resources.
- Ansible is an open-source configuration as code (CaC) tool that helps automate applications, configurations, and IT infrastructure.
- Jenkins is an open-source automation server that enables developers to build, test, and deploy their software.

Code

The code for this pattern is available in the GitHub aws-install-sap-with-jenkins-ansible repository.

Epics

Configure the prerequisites

Task	Description	Skills required
Add your SAP media files to an Amazon S3 bucket.	Create an Amazon S3 bucket that contains all of your SAP media files. **Important** Make sure that you follow the AWS Launch Wizard's folder hierarchy for **S/4HANA** in the Launch Wizard documentation.	Cloud administrator
Install VirtualBox.	Install and configure VirtualBox by Oracle.	DevOps engineer
Install Vagrant.	Install and configure Vagrant by HashiCorp.	DevOps engineer
Configure your AWS account.	1. Verify that you have an IAM principal with an access key and secret key, and that has the following permissions: • **Read only permissions:** Amazon Route 53, AWS Key Management Service (AWS KMS) • **Read and write permissions:** Amazon	General AWS

Task	Description	Skills required
	S3, Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic File System (Amazon EFS), IAM, Amazon CloudWatch, Amazon DynamoDB 2. Save the IAM principal's access key and secret key for reference later. 3. Create a Route 53 private hosted zone, if you don't have one already. Save the zone name (for example, **sapteam.net**) for reference later. 4. Subscribe to the Red Hat Enterprise Linux for SAP with HA and Update Services 8.2 AMI in Amazon Marketplace. Save the **AMI ID** (for example, **ami-0000000**) for reference later. 5. Create an AWS KMS customer managed key. Save the KMS key's Amazon Resource Name (ARN) for reference later. **Note** The following is an example AWS KMS customer	

Task	Description	Skills required
	managed key ARN: **arn:aws:kms:us-east-1:123412341234:key/uuid** 6. Create an SSH key pair. Save the key pair's name and .pem file for reference later. 7. Create an Amazon EC2 security group that allows SSH connection on port 22 from the hostname where you install Jenkins. Save the **security group ID** for reference later. **Note** The hostname is most likely **localhost**.	

Build and run your SAP installation

Task	Description	Skills required
Clone the code repository from GitHub.	Clone the aws-install-sap-with-jenkins-ansible repository on GitHub.	DevOps engineer
Start the Jenkins service.	Open the Linux terminal. Then, navigate to the local folder that contains the	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>cloned code repository folder and run the following command:
<pre>sudo vagrant up</pre>
Note
The Jenkins startup takes about 20 minutes. The command returns a Service is up and running message when successful.</td><td></td></tr>
<tr><td>Open Jenkins in a web browser and log in.</td><td>1. In a web browser, enter http://localhost:5555. Jenkins opens.
2. Log in to Jenkins by using admin for the username and my_secret_pass_from_vault for the password.</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
Configure your SAP system installation parameters.	1. In Jenkins, choose **Manage Jenkins**. Then, choose **Manage Credentials**. A list of credential variables that you can configure appears. 2. Configure all of the following credential variables: • For **AWS_ACCOUNT_CREDENTIALS**, enter your IAM principal's access key ID and secret access key ID. • For **AMI_ID**, enter the Red Hat Enterprise Linux for SAP with HA and Update Services 8.2 AMI's AMI ID. • For **KMS_KEY_ARN**, enter your AWS KMS customer managed key's ARN. • For **SSH_KEYPAIR_NAME**, enter the name of your SSH key pair, without entering the **.pem** file type. • For **SSH_KEYPAIR_FILE**, enter the full name of your key pair's .pem file (for example, **mykeypair.pem**). Make sure that you also upload the key pairs' .pem file to Jenkins.	AWS systems administrator, DevOps engineer

Task	Description	Skills required
	• For **S3_ROOT_FOLDER_INSTALL_FILES**, enter the name of the Amazon S3 bucket—and folder, if applicable—(for example, **s3://my-media-bucket/S4H1909**) that contains your SAP media files. • For **PRIVATE_DNS_ZONE_NAME**, enter the name of your Route 53 private hosted zone (for example, **myprivatecompanyurl.net**). • For **VPC_ID**, enter the VPC ID (for example, **vpc-12345**) of the Amazon VPC that you're creating the SAP resources in. • For **SUBNET_IDS**, enter two public subnet IDs if you're working in a test environment (for future HA capabilities). If you're working in a production environment, it's a best practice to use two private subnets with a bastion host. • For **SECURITY_GROUP_ID**, enter the ID of the Amazon EC2 security group that allows SSH connection on port 22 from the hostname where you installed Jenkins.	

Task	Description	Skills required
	Note You can configure the other nonrequired parameters as needed, based on your use case. For example, you can change the SAP system ID (SID) of the instances, default password, names, and tags for your SAP system. All of the required variables have **(Required)** at the beginning of their names.	

Task	Description	Skills required
Run you SAP system installat ion.	1. In Jenkins, choose **Jenkins Home**. Then, choose **SAP Hana+ASCS+PAS 3 Instances**. 2. Choose **Spin up and install**. Then, choose **Main**. 3. Choose **Build now**. For information on the pipeline steps, see the **Understanding the pipeline steps** section of Automating SAP installation with open-source tools on the AWS Blog. **Note** If an error occurs, move your cursor over the red error box that appears and choose **Logs**. The logs for the pipeline step that errored out appear. Most errors occur because of incorrect parameter settings.	DevOps engineer, AWS systems administrator

Related resources

- DevOps for SAP – SAP Installation: From 2 Months to 2 Hours (DevOps Enterprise Summit Video Library)

Automate AWS Service Catalog portfolio and product deployment by using AWS CDK

Sandeep Gawande, Viyoma Sachdeva, and RAJNEESH TYAGI, Amazon Web Services

Summary

AWS Service Catalog helps you centrally manage catalogs of IT services, or *products*, that are approved for use in your organization's AWS environment. A collection of products is called a *portfolio*, and a portfolio also contains configuration information. With AWS Service Catalog, you can create a customized portfolio for each type of user in your organization and then grant access to the appropriate portfolio. Those users can then quickly deploy any product they need from within the portfolio.

If you have a complex networking infrastructure, such as multi-Region and multi-account architectures, it is recommended that you create and manage Service Catalog portfolios in a single, central account. This pattern describes how to use AWS Cloud Development Kit (AWS CDK) to automate creation of Service Catalog portfolios in a central account, grant end users access to them, and then, optionally, provision products in one or more target AWS accounts. This ready-to-use solution creates the Service Catalog portfolios in the source account. It also, optionally, provisions products in target accounts by using AWS CloudFormation stacks and helps you configure TagOptions for the products:

- **AWS CloudFormation StackSets –** You can use StackSets to launch Service Catalog products across multiple AWS Regions and accounts. In this solution, you have the option to automatically provision products when you deploy this solution. For more information, see Using AWS CloudFormation StackSets (Service Catalog documentation) and StackSets concepts (CloudFormation documentation).
- **TagOption library –** You can manage tags on provisioned products by using TagOption library. A *TagOption* is a key-value pair managed in AWS Service Catalog. It is not an AWS tag, but it serves as a template for creating an AWS tag based on the TagOption. For more information, see TagOption library (Service Catalog documentation).

Prerequisites and limitations

Prerequisites

- An active AWS account that you want to use as the source account for administering Service Catalog portfolios.
- If you are using this solution to provision products in one or more target accounts, the target account must already exist and be active.
- AWS Identity and Access Management (IAM) permissions to access AWS Service Catalog, AWS CloudFormation, and AWS IAM.

Product versions

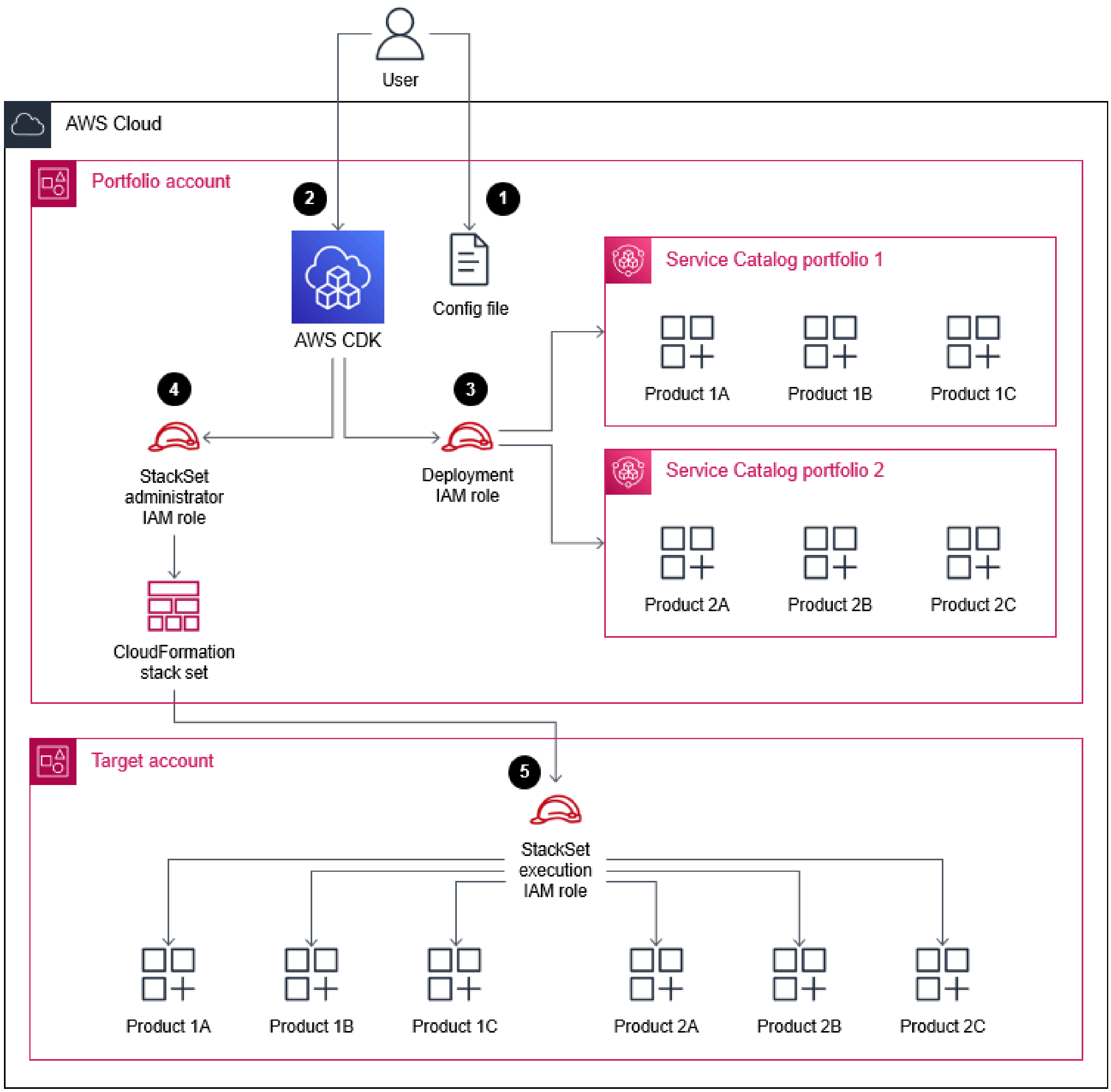
- AWS CDK version 2.27.0

Architecture

Target technology stack

- Service Catalog portfolios in a centralized AWS account
- Service Catalog products deployed in target account

Target architecture



1. In the portfolio (or *source*) account, you update the **config.json** file with the AWS account, AWS Region, IAM role, portfolio, and product information for your use case.
2. You deploy the AWS CDK application.
3. The AWS CDK application assumes the deployment IAM role and creates the Service Catalog portfolios and products defined in the **config.json** file.

If you configured StackSets to deploy products in a target account, the process continues. If you didn't configure StackSets to provision any products, then the process is complete.

4. The AWS CDK application assumes the **StackSet administrator** role and deploys the AWS CloudFormation stack set you defined in the **config.json** file.
5. In the target account, StackSets assumes the **StackSet execution** role and provisions the products.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CDK Toolkit is a command line cloud development kit that helps you interact with your AWS CDK app.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Service Catalog helps you centrally manage catalogs of IT services that are approved for AWS. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.

Code repository

The code for this pattern is available on GitHub, in the aws-cdk-servicecatalog-automation repository. The code repository contains the following files and folders:

- **cdk-sevicecatalog-app** – This folder contains the AWS CDK application for this solution.
- **config** – This folder contains the **config.json** file and the CloudFormation template for deploying the products in the Service Catalog portfolio.
- **config/config.json** – This file contains all of the configuration information. You update this file to customize this solution for your use case.
- **config/templates** – This folder contains the CloudFormation templates for the Service Center products.

- **setup.sh** – This script deploys the solution.
- **uninstall.sh** – This script deletes the stack and all of the AWS resources created when deploying this solution.

To use the sample code, follow the instructions in the Epics section.

Best practices

- IAM roles used to deploy this solution should adhere to the principle of least-privilege (IAM documentation).
- Adhere to the Best practices for developing cloud applications with AWS CDK (AWS blog post).
- Adhere to the AWS CloudFormation best practices (CloudFormation documentation).

Epics

Set up your environment

Task	Description	Skills required

Install the AWS CDK Toolkit.	Make sure you have AWS CDK Toolkit installed. Enter the following command to confirm whether it is installed and check the version.
	`cdk --version`
	If AWS CDK Toolkit is not installed, then enter the following command to install it.
	`npm install -g aws-cdk@2.27.0`
	If AWS CDK Toolkit version is earlier than 2.27.0, then enter
		AWS DevOps, DevOps engineer

Task	Description	Skills required
	the following command to update it to version 2.27.0. `npm install -g aws-cdk@2.27.0 --force`	

Clone the repository.	Enter the following command. In *Clone the repository* in the [Additional information](#) section, you can copy the full command containing the URL for the repository. This clones the [aws-cdk-servicecatalog-automation](#) repository from GitHub.
	`git clone <repository-URL>.git`
	This creates a `cd aws-cdk-servicecatalog-automation` folder in the target directory. Enter the following command to navigate into this folder.
	`cd aws-cdk-servicecatalog-automation`
		AWS DevOps, DevOps engineer

Task	Description	Skills required

Set up AWS credentials.	Enter the following commands. These export the following variables, which define the AWS account and Region where you are deploying the stack.
	
	`export CDK_DEFAULT_ACCOUNT=<12-digit AWS account number>`
	
	`export CDK_DEFAULT_REGION=<AWS Region>`
	
	AWS credentials for AWS CDK are provided through environment variables.
		AWS DevOps, DevOps engineer

Configure permissions for end user IAM roles.	If you are going to use IAM roles to grant access to the portfolio and the products in it, the roles must have permissions to be assumed by the **servicecatalog.amazonaws.com** service principal. For instructions about how to grant these permissions, see [Enabling trusted access with Service Catalog](#) (AWS Organizations documentation).	AWS DevOps, DevOps engineer

Task	Description	Skills required
Configure IAM roles required by StackSets.	If you are using StackSets to automatically provision products in target accounts, you need to configure the IAM roles that administer and run the stack set. 1. In the source account, confirm whether the `AWSCloudFormationStackSetAdministrationRole` already exists. In the target accounts, confirm whether `AWSCloudFormationStackSetExecutionRole` already exists. If these roles already exist, you can skip to the next epic. 2. Follow the instructions in [Grant self-managed permissions](#) (IAM documentation) to create the stack set administration role in the portfolio account and create the execution role in each target account.	AWS DevOps, DevOps engineer

Customize and deploy the solution

Task	Description	Skills required
Create the CloudFormation templates.	In the `config/templates` folder, create CloudFormation templates for any products that you want to include in your portfolios. For more information, see [Working with AWS CloudFormation templates](#) (CloudFormation documentation).	App developer, AWS DevOps, DevOps engineer
Customize the config file.	In the `config` folder, open the **config.json** file and define the parameters as appropriate for your use case. The following parameters are required unless otherwise noted: In the `portfolios` section, define the following parameters to create one or more Service Catalog portfolios: • `portfolioName` – The name of the portfolio. • `providerName` – The name of the person, team, or organization that manages the portfolio. • `description` – A brief description of the portfolio.	App developer, DevOps engineer, AWS DevOps

Task	Description	Skills required
	`roles` – (Optional) Names of any IAM roles that should have access to this portfolio. Users who have this role can access the products in this portfolio. `users` – (Optional) Names of any IAM users who should have access to this portfolio and its products. `groups` – (Optional) Names of any IAM user groups that should have access to this portfolio and its products. **Warning** IAM users have long-term credentials, which presents a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed.	

Task	Description	Skills required
	Important `roles`, `users`, and `groups` are all optional parameters, but if you do not define one of these parameters, then no one can view the portfolio products in the Service Catalog console. Define at least one of these parameters. For more information, see [Grant permissions to Service Catalog end users](#) (Service Catalog documentation). • (Optional) In the `tagOption` section, define TagOptions for the products: 　• `key` – Name of the TagOption key 　• `value` – Allowed string values for the TagOption For more information, see [TagOption library](#) (Service Catalog documentation).	

Task	Description	Skills required
	• In the `products` section, define the following parameters for the products: • `portfolioName` – The name of the portfolio where you want to add the product. You can specify only one portfolio. • `productName` – The name of the product. • `owner` – The owner of the product. • `productVersionName` – The name of the product version in string value, such as `v1`. • `templatePath` – The file path for the CloudFormation template for the product. • `deployWithStackSets` – (Optional) Specify one or more accounts and Regions where you want to use StackSets to automatically provision products in the portfolios. If you use this deployment option, all of the following parameter	

Task	Description	Skills required
	s in this section are required: • `accounts` – The target accounts. • `regions` – The target Regions. • `stackSetAdministrationRoleName` – The name of the IAM role used to administer the StackSets configuration. Do not change this value. This role must have this exact name. • `stackSetExecutionRoleName` – The name of the IAM role in the target account that deploys the stack instances. Do not change this value. This role must have this exact name. For an example of a completed config file, see *Sample config file* in the [Additional information](#) section.	

Task	Description	Skills required
Deploy the solution.	Enter the following command. This deploys the AWS CDK app and provisions the Service Catalog portfolios and products as specified in the **config.json** file. `sh +x setup.sh`	App developer, DevOps engineer, AWS DevOps

Task	Description	Skills required
Verify the deployment.	Verify successful deployment by doing the following: 1. Sign in to the AWS Management Console with credentials that can access one or more of the portfolios you defined in the config file. 2. Open the Service Catalog console at https://console.aws.amazon.com/servicecatalog/. 3. In the navigation pane, under **Provisioning**, choose **Products**. Verify that you see a list of products that you specified for the portfolio. 4. Follow the instructi ons in Launching a product (Service Catalog documentation) to launch one of the available products. Confirm that the available product versions and tags match the values you provided in the config file. 5. If you chose to automatic ally provision products in one or more target accounts by using	General AWS

Task	Description	Skills required
	StackSets, do the following : a. Sign in with credentials that give you permissions to view the provisioned products in one of the target accounts. b. In the Service Catalog console, in the navigation pane, under **Provisioning**, choose **Provisioned products**. c. Confirm that the expected products appear in the list.	

Task	Description	Skills required
(Optional) Update the portfolios and products.	If you want to use this solution to update the portfolios or products or to provision new products: 1. Make the required changes in the **config.json** file. 2. Add or modify any CloudFormation templates as needed in the `config/template` folder. 3. Redeploy the solution. For example, you can add additional portfolios or provision more resources. The AWS CDK app implement s only the changes. If there are no changes to previousl y deployed portfolios or products, the redeployment doesn't affect them.	App developer, DevOps engineer, General AWS

Clean up the solution

Task	Description	Skills required
(Optional) Remove AWS resources deployed by this solution.	If you want to delete a provisioned product, follow the instructions in [Deleting provisioned products](#) (Service Catalog documentation).	AWS DevOps, DevOps engineer, App developer

Task	Description	Skills required
	If you want to delete all the resources created by this solution, enter the following command. `sh uninstall.sh`	

Related resources

- AWS Service Catalog Construct Library (AWS API Reference)
- StackSets concepts (CloudFormation documentation)
- AWS Service Catalog (AWS marketing)
- Using Service Catalog with the AWS CDK (AWS workshop)

Additional information

Clone the repository

Enter the following command to clone the repository from GitHub.

```
git clone https://github.com/aws-samples/aws-cdk-servicecatalog-automation.git
```

Sample config file

The following is a sample **config.json** file with example values.

```
{
    "portfolios": [
        {
            "displayName": "EC2 Product Portfolio",
            "providerName": "User1",
            "description": "Test1",
            "roles": [
                "<Names of IAM roles that can access the products>"
            ],
            "users": [
```

```
                "<Names of IAM users who can access the products>"
            ],
            "groups": [
                "<Names of IAM user groups that can access the products>"
            ]
        },
        {
            "displayName": "Autoscaling Product Portfolio",
            "providerName": "User2",
            "description": "Test2",
            "roles": [
                "<Name of IAM role>"
            ]
        }
    ],
    "tagOption": [
        {
            "key": "Group",
            "value": [
                "finance",
                "engineering",
                "marketing",
                "research"
            ]
        },
        {
            "key": "CostCenter",
            "value": [
                "01",
                "02",
                "03",
                "04"
            ]
        },
        {
            "key": "Environment",
            "value": [
                "dev",
                "prod",
                "stage"
            ]
        }
    ],
    "products": [
```

```
        {
            "portfolioName": "EC2 Product Profile",
            "productName": "Ec2",
            "owner": "owner1",
            "productVersionName": "v1",
            "templatePath": "../../config/templates/template1.json"
        },
        {
            "portfolioName": "Autoscaling Product Profile",
            "productName": "autoscaling",
            "owner": "owner1",
            "productVersionName": "v1",
            "templatePath": "../../config/templates/template2.json",
            "deployWithStackSets": {
                "accounts": [
                    "012345678901",
                ],
                "regions": [
                    "us-west-2"
                ],
                "stackSetAdministrationRoleName": 
 "AWSCloudFormationStackSetAdministrationRole",
                "stackSetExecutionRoleName": "AWSCloudFormationStackSetExecutionRole"
            }
        }
    ]
}
```

Automate deletion of AWS CloudFormation stacks and associated resources

SANDEEP SINGH and James Jacob, Amazon Web Services

Summary

AWS CloudFormation is a widely-used service for managing cloud infrastructure as code (IaC). When you use CloudFormation, you manage related resources as a single unit called a *stack*. You create, update, and delete a collection of resources by creating, updating, and deleting stacks.

Sometimes, you no longer need the resources in a CloudFormation stack. Depending on the resources and their configurations, it can be complicated to delete a stack and its associated resources. In real-world production systems, deletions sometimes fail or take a long time due to conflicting conditions or restrictions that CloudFormation cannot override. It can require careful planning and execution to make sure that all resources are properly deleted in an efficient and consistent manner. This pattern describes how to set up a framework that helps you manage the deletion of CloudFormation stacks that involve the following complexities:

- **Resources with delete protection –** Some resources might have delete protection enabled. Common examples are Amazon DynamoDB tables and Amazon Simple Storage Service (Amazon S3) buckets. Delete protection prevents automated deletion, such as deletion through CloudFormation. If you want to delete these resources, you must manually or programmatically override or temporarily disable the delete protection. You should carefully consider the implication of deleting these resources before proceeding.
- **Resources with retention policies –** Certain resources, such as AWS Key Management Service (AWS KMS) keys and Amazon S3 buckets, might have retention policies that specify how long they should be retained after deletion is requested. You should account for these policies in the cleanup strategy to maintain compliance with organizational policies and regulatory requirements.
- **Delayed deletion of Lambda functions that are attached to a VPC –** Deleting an AWS Lambda function that is attached to a virtual private cloud (VPC) can take 5–40 minutes, depending the multiple interconnected dependencies involved in the process. If you detach the function from the VPC before deleting the stack, you can reduce this delay to under 1 minute.
- **Resources not directly created by CloudFormation –** In certain application designs, resources might be created outside of the original CloudFormation stack, either by the application itself or by resources provisioned through the stack. The following are two examples:

  - CloudFormation might provision an Amazon Elastic Compute Cloud (Amazon EC2) instance that runs a user data script. Then, this script might create an AWS Systems Manager parameter to store application-related data. This parameter is not managed through CloudFormation.
  - CloudFormation might provision a Lambda function that automatically generates an Amazon CloudWatch Logs group for storing logs. This log group is not managed through CloudFormation.

  Even though these resources aren't directly managed by CloudFormation, they often need to be cleaned up when the stack is deleted. If left unmanaged, they can become orphaned and lead to unnecessary resource consumption.

Although these guardrails can cause complexity, they are intentional and critical. Allowing CloudFormation to override all constraints and indiscriminately delete resources could lead to detrimental and unforeseen consequences in many scenarios. However, as a DevOps or cloud engineer who is responsible for managing the environment, there are times when overriding these constraints might be necessary, particularly in development, testing, or staging environments.

Targeted business outcomes

By implementing this framework, you can achieve the following benefits:

- **Cost management** – Regular and efficient cleanup of temporary environments, such as end-to-end or user-acceptance testing environments, helps prevent resources from running longer than necessary. This can significantly reduce costs.
- **Security** – Automated cleanup of outdated or unused resources reduces the attack surface and helps maintain a secure AWS environment.
- **Operational efficiency** – Regular and automated cleanup can provide the following operational benefits:
  - Automated scripts that remove old log groups or empty Amazon S3 buckets can improve operational efficiency by keeping the environment clean and manageable.
  - Quickly deleting and recreating stacks supports rapid iteration for design and implementation, which can lead to a more robust and resilient architecture.
  - Regularly deleting and rebuilding environments can help you identify and fix potential issues. This can help you make sure that the infrastructure can withstand real-world scenarios.

Prerequisites and limitations

Prerequisites

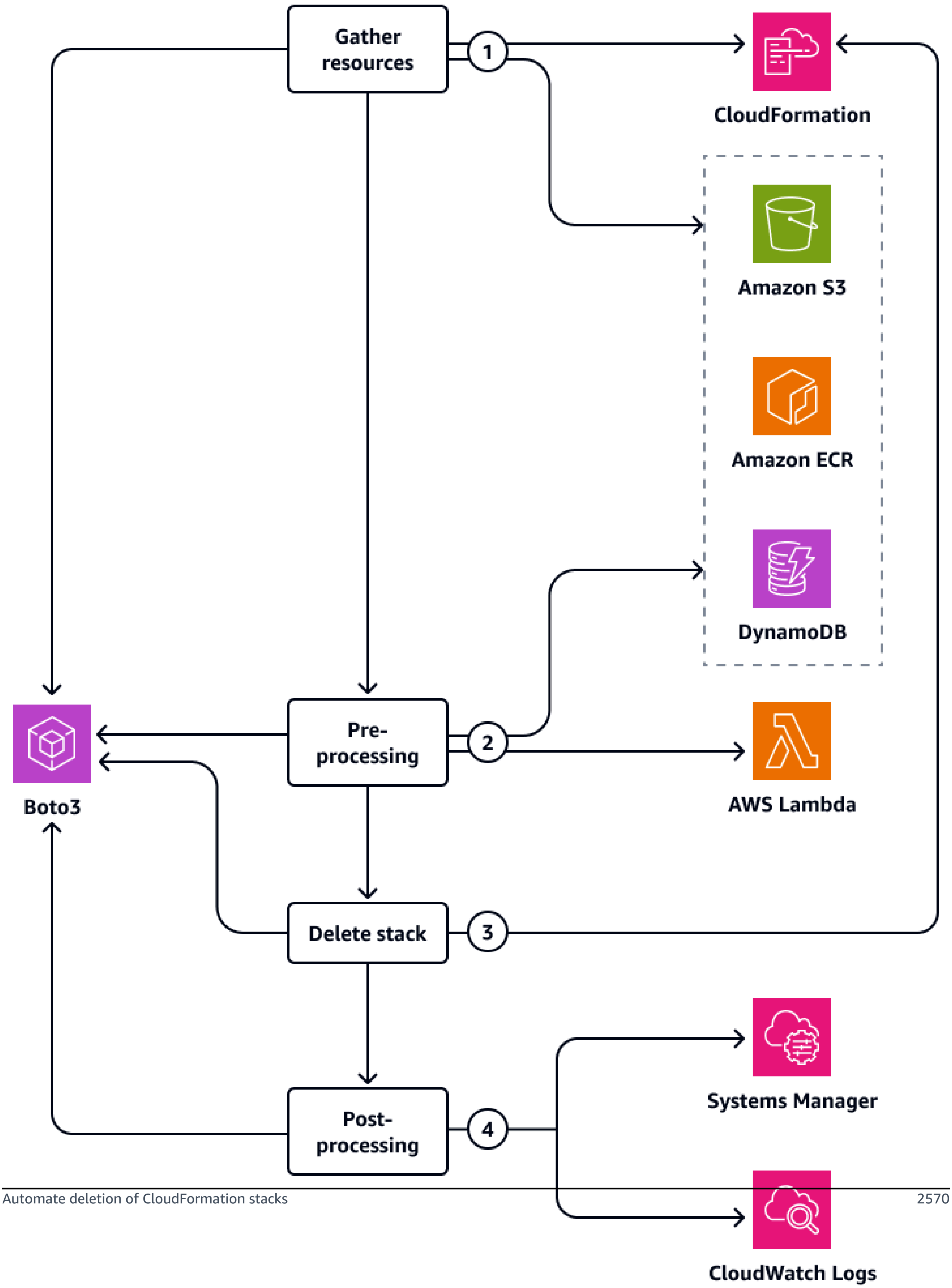
- An active AWS account
- Python version 3.6 or later, installed
- AWS Command Line Interface (AWS CLI), installed and configured

Limitations

- A naming convention is used to identify the resources that should be deleted. The sample code in this pattern uses a prefix for the resource name, but you can define your own naming convention. Resources that do not use this naming convention will not be identified or subsequently deleted.

Architecture

The following diagram shows how this framework identifies the target CloudFormation stack and the additional resources associated with it.


Gather resources
1
CloudFormation
Amazon S3
Amazon ECR
DynamoDB
Boto3
Pre-processing
2
AWS Lambda
Delete stack
3
Systems Manager
Post-processing
4
CloudWatch Logs

The diagram shows the following workflow:

1. **Gather resources** – The automation framework uses a naming convention to return all relevant CloudFormation stacks, Amazon Elastic Container Registry (Amazon ECR) repositories, DynamoDB tables, and Amazon S3 buckets.

   > **Note**
   >
   > The functions for this stage use paginators, a feature in Boto3 that abstracts the process of iterating over a truncated API result set. This makes sure that all resources are processed. To further optimize performance, consider applying server-side filtering or consider using JMESPath to perform client-side filtering.

2. **Pre-processing** – The automation framework identifies and addresses the service constraints that must be overridden in order to allow CloudFormation to delete the resources. For example, it changes the `DeletionProtectionEnabled` setting for DynamoDB tables to `False`. In the command-line interface, for each resource, you receive a prompt asking if you want to override the constraint.
3. **Delete stack** – The automation framework deletes the CloudFormation stack. In the command-line interface, you receive a prompt asking if you want to delete the stack.
4. **Post-processing** – The automation framework deletes any related resources that were not directly provisioned through CloudFormation as part of the stack. Examples of these resource types include Systems Manager parameters and CloudWatch log groups. Separate functions gather these resources, pre-process them, and then delete them. In the command-line interface, for each resource, you receive a prompt asking if you want to delete the resource.

   > **Note**
   >
   > The functions for this stage use paginators, a feature in Boto3 that abstracts the process of iterating over a truncated API result set. This makes sure that all resources are processed. To further optimize performance, consider applying server-side filtering or consider using JMESPath to perform client-side filtering.

Automation and scale

If your CloudFormation stack includes other resources that are not included in the sample code, or if the stack has a constraint that has not been addressed in this pattern, then you can adapt the automation framework for your use case. Follow the same methodology of gathering resources, pre-processing, deleting the stack, and then post-processing.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- CloudFormation Command Line Interface (CFN-CLI) is an open source tool that helps you develop and test AWS and third-party extensions and then register them for use in CloudFormation.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services.

Other tools

- Click is a Python tool that helps you create command line interfaces.
- Poetry is a tool for dependency management and packaging in Python.
- Pyenv is a tool that helps you manage and switch between versions of Python.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub cloudformation-stack-cleanup repository.

Best practices

- **Tag resources for easy identification –** Implement a tagging strategy to identify resources that are created for different environments and purposes. Tags can simplify the cleanup process by helping you filter resources based on their tags.
- **Set up resource life cycles –** Define resource life cycles in order to automatically delete resources after a certain period. This practice helps you make sure that temporary environments do not become permanent cost liabilities.

Epics

Install tools

Task	Description	Skills required
Clone the repository.	1. Create a folder in your virtual environment. Name it with your project name. 2. Open a terminal on your local machine, and navigate to this folder. 3. Enter the following command to clone the [cloudformation-stack-cleanup](https://github.com/aws-samples/cloudformation-stack-cleanup) repository to your project directory: `git clone https://github.com/aws-samples/cloudformation-stack-cleanup.git`	DevOps engineer
Install Poetry.	Follow the [instructions](https://python-poetry.org/docs/) (Poetry documentation) to install Poetry in the target virtual environment.	DevOps engineer
Install dependencies.	1. Enter the following command to navigate to the project directory: `cd cloudformation-stack-cleanup` 2. Enter the following command:	DevOps engineer

Task	Description	Skills required
	`poetry install` This installs all of the required dependencies, such as Boto3, click, and the source code for the CloudFormation CLI.	
(Optional) Install Pyenv.	Follow the instructions (GitHub) to install Pyenv.	DevOps engineer

(Optional) Customize the framework

Task	Description	Skills required
Create functions that gather, pre-process, and delete the target resources.	1. In the cloned repository, enter the following command to navigate to the `cli` directory: `cd cfncli/cli` 2. Open the **cleanup_enviornment.py** file. 3. Create a new Python function that gathers the type of resource that you want to modify. For an example, see the `gather_ddb_tables` function in this file. 4. Create a new Python function that overrides the service constraints	DevOps engineer, Python

Task	Description	Skills required
	for the target resource. For an example, see the `remove_ddb_deletion_protection` function in this file. 5. Create a new Python function that collects unmanaged target resources. For an example, see the `gather_log_groups` function in this file. 6. Create a new Python function that deletes unmanaged target resources. For an example, see the `delete_log_group` function in this file. 7. Save and close the **cleanup_enviornment.py** file.	

Create sample resources

Task	Description	Skills required
Create a CloudFormation stack.	1. Navigate to the project directory. 2. Enter the following command to create a CloudFormation stack that provisions a DynamoDB	AWS DevOps

Task	Description	Skills required
	table and a security group. Update the value for <VPCID>: `aws cloudformation create-stack \ --stack-name sampleforcleanup-Stack \ --template-body file://samples/sample-cfn-stack.yaml \ --parameters ParameterKey=VpcId,ParameterValue=<VPCID> \ --region us-east-1`	
Create a Systems Manager parameter.	Enter the following command to create a Systems Manager parameter that isn't provisioned through CloudFormation: `aws ssm put-parameter \ --name "/sampleforcleanup/database/password" \ --value "your_db_password" \ --type "SecureString" \ --description "Database password for my app" \ --tier "Standard" \ --region "us-east-1"`	AWS DevOps

Task	Description	Skills required
Create an Amazon S3 bucket.	Enter the following command to create an Amazon S3 bucket that isn't provisioned through CloudFormation: `aws s3api create-bucket \ --bucket samplesorcleanup-unmanagedbucket-<UniqueIdentifier> \ --region us-east-1 \ --create-bucket-configuration LocationConstraint=us-east-1`	AWS DevOps

Delete the sample resources

Task	Description	Skills required
Delete the CloudFormation stack.	1. Enter the following command to delete the sample CloudFormation stack, Systems Manager parameter, and Amazon S3 bucket that you created: `cfncli --region us-east-1 \ dev cleanup-env \ --prefix-list sampleforcleanup` 2. When prompted, enter Y to continue.	AWS DevOps

Task	Description	Skills required
Validate resource deletion.	In the output, confirm that all of the sample resources have been deleted. For a sample output, see the Additiona l resources section of this pattern.	AWS DevOps

Related resources

- Delete a stack (CloudFormation documentation)
- Troubleshooting CloudFormation (CloudFormation documentation)
- Giving Lambda functions access to resources in an Amazon VPC (Lambda documentation)
- How do I delete an AWS CloudFormation stack that's stuck in DELETE_FAILED status? (AWS Knowledge Center)

Additional information

The following is a sample output from the `cfncli` command:

```
cfncli --region aus-east-1  dev cleanup-env --prefix-list sampleforcleanup


https://sts.us-east-1.amazonaws.com
Cleaning up: ['sampleforcleanup'] in xxxxxxxxxx:us-east-1
Do you want to proceed? [Y/n]: Y
No S3 buckets
No ECR repositories
No Lambda functions in VPC
The following DynamoDB tables will have their deletion protection removed:
sampleforcleanup-MyDynamoDBTable
Do you want to proceed with removing deletion protection from these tables? [Y/n]: Y
Deletion protection disabled for DynamoDB table 'sampleforcleanup-MyDynamoDBTable'.
The following CloudFormation stacks will be deleted:
sampleforcleanup-Stack
Do you want to proceed with deleting these CloudFormation stacks? [Y/n]: Y
Initiated deletion of CloudFormation stack: `sampleforcleanup-Stack`
```

```
Waiting for stack `sampleforcleanup-Stack` to be deleted...
CloudFormation stack `sampleforcleanup-Stack` deleted successfully.
The following ssm_params will be deleted:
/sampleforcleanup/database/password
Do you want to proceed with deleting these ssm_params? [Y/n]: Y
Deleted SSM Parameter: /sampleforcleanup/database/password
Cleaned up: ['sampleforcleanup']
```

Automate dynamic pipeline management for deploying hotfix solutions in Gitflow environments by using AWS Service Catalog and AWS CodePipeline

Balaji Vedagiri, Faisal Shahdad, Shanmugam Shanker, and Vivek Thangamuthu, Amazon Web Services

Summary

> **Note**
>
> AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern addresses a scenario of managing a dynamic hotfix pipeline that's dedicated solely to deploying hotfix solutions to a production environment securely. The solution is implemented and managed by using an AWS Service Catalog portfolio and product. An Amazon EventBridge rule is used for event automation. Restrictions are enforced by using Service Catalog portfolio constraints and AWS Identity and Access Management (IAM) roles for developers. Only an AWS Lambda function is allowed to launch the Service Catalog product, triggered by the EventBridge rule. This pattern is designed for environments with a specific Gitflow setup, which is described in Additional information.

Typically, a hotfix is deployed to address critical or security issues reported in a live environment, such as Production. Hotfixes should be deployed directly to Staging and Production environments only. The Staging and Production pipelines are used extensively for regular development requests. These pipelines can't be used to deploy hotfixes because there are ongoing features in quality assurance that can't be promoted to Production. To release hotfixes, this pattern describes a dynamic, short-lived pipeline with the following security features:

- **Automatic creation** – A hotfix pipeline is automatically created whenever a hotfix branch is created in an AWS CodeCommit repository.
- **Access restrictions** – Developers don't have access to create this pipeline outside of the hotfix process.
- **Controlled stage** – The pipeline has a controlled stage with a special access token, ensuring that a pull request (PR) can only be created once.

- **Approval stages –** Approval stages are included in the pipeline to get necessary approvals from relevant stakeholders.
- **Automatic deletion –** The hotfix pipeline is automatically deleted whenever a `hotfix` branch is deleted in the CodeCommit repository after it's merged with a PR.

Prerequisites and limitations

Prerequisites

- Three active AWS accounts are required as follows:
  - Tools account - For continuous integration and continuous delivery (CI/CD) setup.
  - Stage account - For user acceptance testing.
  - Production account - For a business end user.
  - (Optional) Add an AWS account to act as a QA account. This account is required if you want both a main pipeline setup, including QA, and a hotfix pipeline solution for testing.
- An AWS CloudFormation stack with an optional condition to deploy in the QA account using the main pipeline, if needed. The pattern can still be tested without the main pipeline setup by creating and deleting a `hotfix` branch.
- An Amazon Simple Storage Service (Amazon S3) bucket to store the CloudFormation templates that are used to create Service Catalog products.
- Create PR approval rules for the CodeCommit repository in accordance with the compliance requirements (after creating the repository).
- Restrict the IAM permissions of developers and team leads to deny the execution of the prcreation-lambda Lambda function because it should be invoked only from the pipeline.

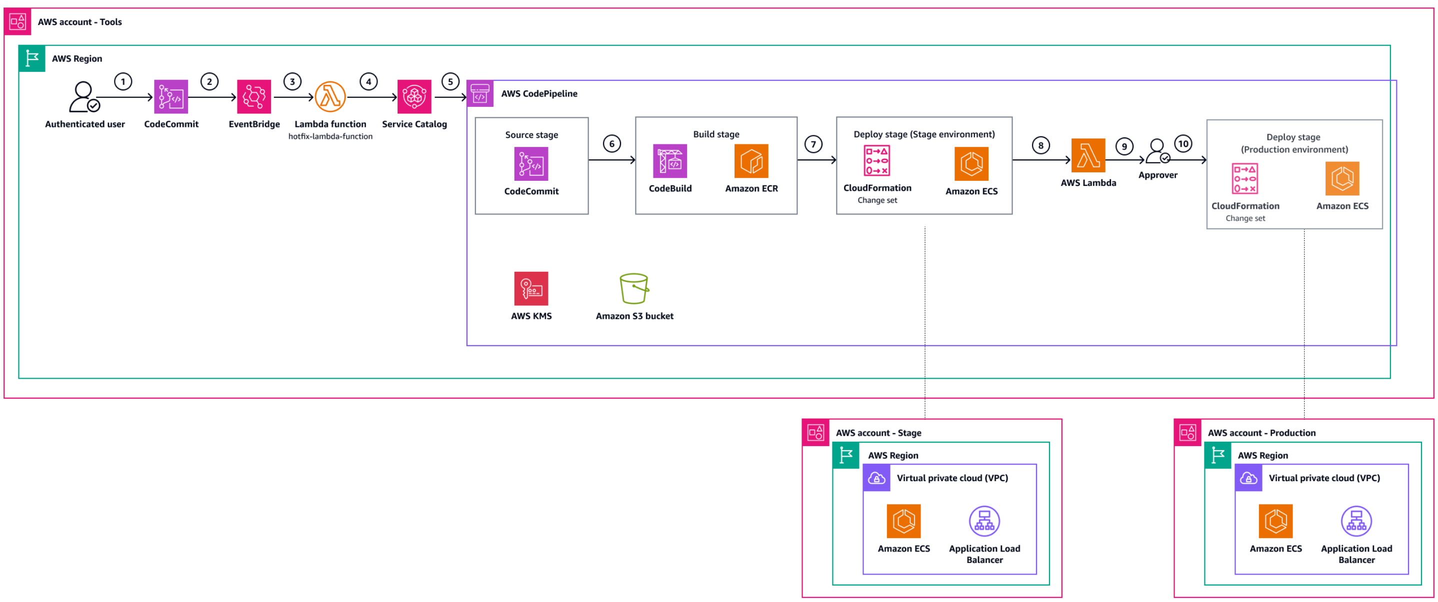
Limitations

- The CloudFormation provider is used in the deploy stage, and the application is deployed using a CloudFormation change set. If you want to use a different deployment option, modify the CodePipeline stack as required.
- This pattern uses AWS CodeBuild and other configuration files to deploy a sample microservice. If you have a different workload type (for example, serverless workloads), you must update all relevant configurations.

- This pattern deploys the application in a single AWS Region (for example, US East (N. Virginia) us-east-1) across AWS accounts. To deploy across multiple Regions, change the Region reference in commands and stacks.
- Some AWS services aren't available in all AWS Regions. For Region availability, see [AWS services by Region](). For specific endpoints, see [Service endpoints and quotas](), and choose the link for the service.

Architecture

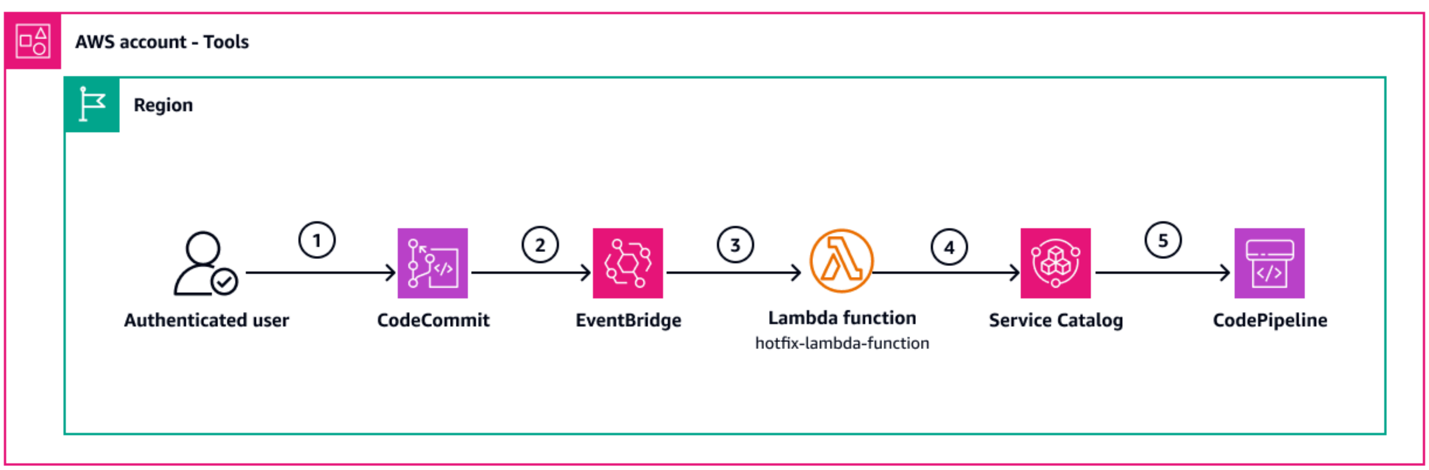
The diagrams in this section provide workflows for a create lifecycle event and for a delete lifecycle event.



The preceding diagram for creating a lifecycle event shows the following:

1. The developer creates a `hotfix-*` branch in the CodeCommit repository to develop a hotfix-related solution.
2. The `hotfix-*` branch creation event is captured through the EventBridge rule. The event details include the repository name and branch name.
3. The EventBridge rule invokes the AWS Lambda function `hotfix-lambda-function`. The EventBridge rule passes the event information to the Lambda function as input.
4. The Lambda function processes the input to retrieve the repository name and branch name. It launches the Service Catalog product with values retrieved from the processed input.

5. The Service Catalog product includes a pipeline setup that will deploy the solution to the Stage and Production environments. The pipeline block includes source, build, and deploy stages. Also, there is a manual approval stage to promote the deployment for the Production environment.
6. The source stage retrieves the code from the repository and `hotfix-*` branch that was created in the first step. The code is passed to the build stage through an Amazon S3 bucket for artifacts. In the build stage, a container image is created that includes the hotfix that is developed in the `hotfix-*` branch and pushed into Amazon Elastic Container Registry (Amazon ECR).
7. The deploy stage to the stage environment updates Amazon Elastic Container Service (Amazon ECS) with the latest container image that includes the hotfix. The hotfix is deployed by creating and executing a CloudFormation change set.
8. The `prcreation-lambda` Lambda function is invoked after successful deployment in the Stage environment. This Lambda function creates a PR from the `hotfix-*` branch to the `develop` and `main` branches of the repository. The Lambda function ensures that the fix developed in the `hotfix-*` branch is backmerged and included in subsequent deployments.
9. A manual approval stage helps to ensure that the necessary stakeholders review the fix and give approval to deploy in Production.
10. The deploy stage to the production environment updates Amazon ECS with the latest container image that includes the hotfix. The hotfix is deployed by creating and executing a CloudFormation change set.



The preceding diagram for deleting a lifecycle event shows the following:

1. The developer deletes the `hotfix-*` branch after successful deployment of the hotfix to the production environment.
2. The `hotfix-*` branch deletion event is captured through an EventBridge rule. The event details include the repository name and branch name.
3. The EventBridge rule invokes the Lambda function. The EventBridge rule passes the event information to the Lambda function as input.
4. The Lambda function processes the input to retrieve the repository name and branch name. The Lambda function determines the respective Service Catalog product from the passed input and then terminates the product.
5. The Service Catalog provisioned product termination deletes the pipeline and relevant resources that were created earlier in that product.

Automation and scale

- The pattern includes an EventBridge rule and a Lambda function, which can handle multiple hotfix branch creation requests in parallel. The Lambda function provisions the Service Catalog product for the matching event rule.
- The pipeline setup is handled by using the Service Catalog product, which provides version control capabilities. The solution also scales automatically to handle multiple hotfix developments for the same application in parallel.
- The prcreation-lambda function ensures that these hotfix changes are also merged back into the `main` and the `develop` branches through an automatic pull request creation. This approach is essential to keep the `main` and the `develop` branches up to date with all fixes and avoid potential code regressions. This process helps to maintain consistency across branches and prevent code regressions by ensuring that all long-lived branches have the latest fixes.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.

- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system. AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. For more information, see How to migrate your AWS CodeCommit repository to another Git provider.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Service Catalog helps you centrally manage catalogs of IT services that are approved for AWS. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- AWS CloudFormation Linter (cfn-lint) is a linter that checks CloudFormation YAML or JSON templates against the CloudFormation resource specification. It also performs other checks, such as checking for valid values for resource properties and adherence to best practices.
- cfn-nag is an open source tool that that identifies potential security issues in CloudFormation templates by searching for patterns.
- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers. This pattern uses Docker to build and test container images locally.
- Git is an open-source, distributed version control system.

Code repository

The code for this pattern is available in the GitHub dynamic_hotfix_codepipeline repository.

Best practices

Review and adjust IAM roles and service control policies (SCP) in your environment to ensure that they restrict access appropriately. This is crucial to prevent any actions that could override the security measures included in this pattern. Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see [Grant least privilege](#) and [Security best practices](#) in the IAM documentation.

Epics

Set up the work environment

Task	Description	Skills required
Clone the repository.	To clone the sample [repository](#) into a new directory in your work location, run the following command: `git clone git@github.com:aws-samples/dynamic_hotfix_codepipeline.git`	AWS DevOps
Export environment variables for CloudFormation stack deployment.	Define the following environment variables that will be used as input to the CloudFormation stacks later in this pattern. • `ApplicationName` – This variable is used to name resources created for an application, making it easier to track them. Use the following command, replacing `Applicati`	AWS DevOps

Task	Description	Skills required

	onname with your actual application name:
	`export ApplicationName=<Applicationname>`
	• `BucketStartName` – This variable is for naming an Amazon S3 bucket. S3 bucket names must be globally unique across all AWS accounts. Use the following command, replacing `BucketName` with a unique name for your S3 bucket:
	`export BucketStartName=<BucketName>`
	• **Account numbers and Regions** – These variables store AWS account numbers for different environments and the deployment Region. Use the following commands, replacing the placeholders (such as `prodaccountnumber` and `region`) with your actual AWS account numbers and the AWS Region that you're using.	

Task	Description	Skills required

	Note
	`QAAccount` is optional. If you want to use `QAAccount` , set it up by using the parameters for the main pipeline stack.
	
	`export ProdAccount=<prodaccountnumber>`
	`export StageAccount=<stage/preprodaccountnumber>`
	`export QAAccount=<qaccountnumber>`
	`export ToolsAccount=<toolsaccountnumber>`
	`export DepRegion=<region>`	

Set up prerequisites required in AWS accounts

Task	Description	Skills required
Create resources required for CI/CD in the tools account.	To deploy the CloudFormation stack in the tools account, use the following commands. (Remove the `QAAccount` parameter	AWS DevOps

Task	Description	Skills required
	if you're not using the QA account for setup.) `#InToolsAccount aws cloudformation deploy \ --template-file pre-requisites/pre-reqs.yaml \ --stack-name prereqs \ --parameter-overrides BucketStartName=${BucketStartName} \ ApplicationName=${ApplicationName} ProdAccount=${ProdAccount} \ StageAccount=${StageAccount} ToolsAccount=${ToolsAccount} \ QAAccount=${QAAccount} \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM --region ${DepRegion}` Make a note of the resources that the CodeCommit repository and Amazon ECR created from the preceding stack. These parameters are necessary to set up the `main` branch of the pipeline in upcoming steps.	

Task	Description	Skills required

Create resources required for CI/CD in the workload accounts.	1. To package the CloudFormation template in each workload account (stage, production, and optional QA), use the following commands. In the following command, replace `S3bucketpackage` with the Amazon S3 bucket name that you want to use for packaging.
	
	`#InStageAccount aws cloudformation package \ --template-file pre-requisites/infrastack.yaml \ --s3-bucket <S3bucketpackage> \ --s3-prefix infraStack \ --region ${DepRegion} \ --output-template-file pre-requisites/infrastructure_stage.template`
	
	`#InProdAccount aws cloudformation package \ --template-file pre-requisites/infrastack.yaml \ --s3-bucket <S3bucketpackage> \`
		AWS DevOps

Task	Description	Skills required

	``
	    --s3-prefix
	 infraStack \
	    --region ${DepRegi
	on} \
	    --output-
	template-file
	 pre-requisites/inf
	rastructure_prod.t
	emplate
	``
	2. To deploy the CloudFormation template in each workload account, use the following commands:
	``
	#InStageAccount
	aws cloudformation
	 deploy --stack-name
	 inframainstack \
	--parameter-over
	rides Applicati
	onName=${Applicati
	onName} ToolsAcco
	unt=${ToolsAccount
	} DepRegion=${DepReg
	ion} \
	--template-file
	 pre-requisites/inf
	rastructure_stage.
	template --region
	 ${DepRegion}
	 --capabilities
	 CAPABILITY_NAMED_I
	AM
	
	#InProdAccount
	aws cloudformation
	 deploy --stack-name
	 inframainstack \
	--parameter-over
	rides Applicati
	``	

Task	Description	Skills required
	```onName=${ApplicationName} ToolsAccount=${ToolsAccount} DepRegion=${DepRegion} \ --template-file pre-requisites/infrastructure_prod.template --region ${DepRegion} --capabilities CAPABILITY_NAMED_IAM```	

Task	Description	Skills required

Update the S3 artifact bucket policy to allow access for workload accounts.	To update the CloudFormation stack prerequisites in the tools account, use the following commands to add all required permissions for the Stage and Production workload accounts. (Remove the `QAAccount` parameter if you're not using it for setup.)
	
	`#InToolsAccount`
	`aws cloudformation deploy \`
	`    --template-file pre-requisites/pre-reqs.yaml \`
	`    --stack-name prereqs \`
	`    --parameter-overrides BucketStartName=${BucketStartName} \`
	`    ApplicationName=${ApplicationName} ProdAccount=${ProdAccount} \`
	`    StageAccount=${StageAccount} ToolsAccount=${ToolsAccount} \`
	`    QAAccount=${QAAccount} PutPolicy=true \`
	`    --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM --region ${DepRegion}`
		AWS DevOps

Set up Lambda function and Service Catalog resources in tools account

Task	Description	Skills required

Set up the Service Catalog portfolio and products.	To set up the Service Catalog portfolio and products, do the following:
	
	1. Upload the templates [pipeline-main.yaml]() and [pipeline-hotfix.yaml]() from the repository in the CodePipeline directory to an existing Amazon S3 bucket (`Bucketname`) in the Region that you want to deploy to (`DepRegion`).
	
	`#InToolsAccount`
	`aws s3 cp ./codepipeline/pipeline-main.yaml s3://<Bucketname>/pipeline-main.yaml`
	`aws s3 cp ./codepipeline/pipeline-hotfix.yaml s3://<Bucketname>/pipeline-hotfix.yaml`
	
	2. Set up the Service Catalog portfolio and product that will manage the pipeline for the `main` and `hotfix` branches. Make a note of the details from the `Outputs` section for `MainProductId` and
		AWS DevOps

Task	Description	Skills required
	`MainProductArtifactId` from the following stack. The information is required in later steps during the pipeline setup for the `main` branch. `#InToolsAccount aws cloudformation deploy \ --template-file pre-requisites/servicecatalogsetup.yaml \ --stack-name servicecatalogsetup \ --parameter-overrides TemplateBucket=<Bucketname> \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM --region ${DepRegion}` 3. Provide access for an IAM role that will deploy resources in the tools account to the Service Catalog portfolio main pipeline portfolio. Use this portfolio to deploy the application by using the `main` branch. For more information about how to provide access, see [Granting Access to Users](#)	

Task	Description	Skills required
	in the Service Catalog documentation.	

Task	Description	Skills required
Set up Lambda functions.	This solution uses the following Lambda functions to manage hotfix workflows: • `hotfix-lambda-function` handles Service Catalog product provisioning when a `hotfix` branch is created. • `hotfix-cleanup-lambda-function` manages product termination when a `hotfix` branch is deleted. • `prcreation-lambda` creates pull requests from the `hotfix` branch to the `develop` and `main` branches. To enable the Lambda functions to provision and terminate Service Catalog products when `hotfix` branches are created or deleted through the associated EventBridge rule, use the following steps: 1. To create the IAM roles and permissions for the Lambda functions, use the following command to	AWS DevOps

Task	Description	Skills required
	deploy a CloudFormation stack: `#InToolsAccount aws cloudformation deploy \ --template-file pre-requisites/lambdasetup.yaml \ --stack-name prsclambdasetup \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM --region ${DepRegion}` 2. After the stack deployment, grant the `hotfix-lambda-execution-role` access to the Service Catalog portfolio hotfix pipeline portfolio by using the [AWS Management Console](https://docs.aws.amazon.com/servicecatalog/latest/adminguide/catalogs_portfolios_users.html). This access allows the Lambda function to launch or terminate the Service Catalog product for hotfix branches.	

Create pipeline for main branch and deploy application in workload accounts

Task	Description	Skills required
Set up pipeline for `main` branch.	To set up the pipeline for the main branch, run the following command in the tools account. Replace the	AWS DevOps

Task	Description	Skills required

	parameters for `MainProductId` and `MainProductArtifactId` with values from `servicecatalogsetup` stack outputs.
	`#InToolsAccount`
	`aws servicecatalog provision-product \`
	`    --product-id <MainProductId> \`
	`    --provisioning-artifact-id <MainProductArtifactId> \`
	`    --provisioned-product-name "${ApplicationName}-main-pipeline" \`
	`    --provisioning-parameters Key=CodeCommitRepoName,Value="${ApplicationName}-repository" Key=ECRRepository,Value="${ApplicationName}-app" \`
	`    --region=${DepRegion}`	

Task	Description	Skills required
Deploy application using the `main` branch.	1. To clone the CodeCommit repository that was created in the *prereqs*, use the `git clone` command. For more information, see [Connect to the CodeCommit repository by cloning the repository](#) as described in the CodeCommit documentation. 2. Copy all the application files from the `repotemplates` [directory](#) available in the repository into your local repository clone (`${ApplicationName}-repository`). Modify the following files to update the `ToolsAccount` ID. In each file, locate the `RegistryAccountid` parameter and set its value to your `ToolsAccount` ID. Commit the changes to the CodeCommit repository, and push the files to both the `main` and `develop` branches. • [ecs-configuration-prod.yaml](#) • [ecs-configuration-qa.yaml](#)	AWS DevOps

Task	Description	Skills required
	• [ecs-configuration-stage.yaml] 3. To verify the application deployment, monitor the CodePipeline execution by using the AWS Managemen t Console. After deploymen t is complete, access the application by using the Application Load Balancer FQDN in the stage environment. Confirm that the application works as expected. 4. To approve the deploymen t to production, use the CodePipeline console to locate the pipeline for your application. Find the `ApprovalToStart` stage. Review the changes and, if satisfactory, provide manual approval to proceed with the productio n deployment.	

Create the pipeline for a hotfix-* branch and deploy the hotfix

Task	Description	Skills required
Create a `hotfix-*` branch and commit changes.	To create a pipeline for the `hotfix-*` branch and deploy the hotfix to the workload accounts, do the following:	AWS DevOps

Task	Description	Skills required
	1. Create a branch using a name that starts with the keyword `hotfix`. For example, this pattern uses the `hotfix-check1` branch in the CodeCommit application repository (`${ApplicationName}-repository`). For more detailed steps, see [Connect to an AWS CodeCommit repository](#) and [Basic Git commands](#) in the CodeCommit documentation. 2. Verify that the Service Catalog product `Hotfix CICD Pipeline` is successfully provisioned dynamically for the `hotfix-check1` branch. The provisioned product name is named after this hotfix branch name and the application's CodeCommit repository name. 3. Commit some minor changes in the [index.html](#) file, and push them to the CodeCommit repository. 4. Verify that the CodePipeline execution is successful in the stage environme	

Task	Description	Skills required
	nt. To deploy in the production environment, provide manual approval in CodePipeline. 5. Confirm the changes are visible in the application home page by using the Application Load Balancer fully qualified domain name (FQDN). The FQDN is available in the `Outputs` section of `inframainstack-ALBStack-*` .	
Delete the `hotfix-check1` branch.	To delete the `hotfix-check1` branch created earlier, do the following: 1. Delete the `hotfix-check1` branch that's in the CodeCommit application repository. 2. Verify that the Service Catalog product that was provisioned for the `hotfix-check1` branch terminated successfully.	AWS DevOps

Clean up resources

Task	Description	Skills required

Clean up the deployed resources.	To clean up the resources that were deployed earlier, do the following:
	1. To scale down the Amazon ECS service to zero replicas in the workload accounts, use the following command:
	`aws ecs update-service --cluster ${ApplicationName}-Cluster --service ${ApplicationName}-Service-stage --desired-count 0 --region ${DepRegion}`
	`aws ecs update-service --cluster ${ApplicationName}-Cluster --service ${ApplicationName}-Service-prod --desired-count 0 --region ${DepRegion}`
	2. Terminate the Service Catalog product that was provisioned for the `main` branch.
	3. Clean up objects created in the Amazon S3 buckets in the tools account. Delete all Docker images
		AWS DevOps

Task	Description	Skills required

	in Amazon ECR before deleting the registry itself.
	4. Remove IAM roles in the access granted section of the Service Catalog portfolios before you delete the Service Catalog portfolio.
	5. Delete the CloudFormation stacks deployed in the tools account and the workload accounts.
	
	`##In Tools Account## aws cloudformation delete-stack --stack-name servicecatalogsetup --region ${DepRegion} aws cloudformation delete-stack --stack-name prlambdasetup --region ${DepRegion} aws cloudformation delete-stack --stack-name prereqs --region ${DepRegion}`
	
	`##In Workload Accounts## aws cloudformation delete-stack --stack-name inframainstack --region ${DepRegion}`	

Task	Description	Skills required
	For more information, see Deleting provisioned products in the Service Catalog documentation.	

Troubleshooting

Issue	Solution
Changes that you committed to the CodeCommit repository aren't getting deployed.	Check the CodeBuild logs for errors in the Docker build action. For more information, see the CodeBuild documentation.
The Service Catalog product isn't being provisioned.	Review the related CloudFormation stacks for failed events. For more information, see the CloudFormation documentation.

Related resources

- Basic Git commands
- Configure an IAM policy to limit pushes and merges to a branch
- Connect to an AWS CodeCommit repository
- Granting access to users
- Pushing a Docker image to an Amazon ECR private repository
- Troubleshooting AWS CodeBuild
- What is AWS CodePipeline?

Additional information

This pattern is designed for environments with a Gitflow setup that is adopted for the development workflow in the CI/CD process. The pipelines follow the deployment cycle that starts from development and moves through quality assurance (QA), stage, and production environments. The CI/CD setup includes two git branches with promotional deployments to environments as follows:

- The `develop` branch deploys to the development environment.
- The `main` branch deploys to the QA, stage, and production environments.

In this setup, it's a challenge to apply a hotfix or a security patch faster than the usual deployment cycle while the active development of new features is ongoing. A dedicated process is necessary to address hotfix or security requests, ensuring that live environments remain properly functioning and secure.

However, you can use other available options without requiring a dedicated deployment process if:

- The CI/CD process is well-equipped with automated testing, such as functional and end-to-end tests, which eliminate the need for manual testing and prevent delays in deployments to production. However, if automated testing isn't well integrated into the CI/CD process, pushing a small fix to the production environment can become complex and cumbersome for developers. This is because there might be new features waiting in the QA environment for approval and sign-off. A hotfix or security fix can't be pushed into production in a straightforward manner simultaneously.
- Development teams continuously deploy new features into the production environment, integrating hotfixes or security patches into the scheduled deployment of each new feature. In other words, the next feature update to the production environment comprises two components: The addition of a new feature and the inclusion of the hotfix or security patch. However, if the deployment cycle isn't continuous, there can be multiple new features already awaiting approval in the QA environment. Managing different versions and ensuring the correct changes are reapplied can then become complex and error-prone.

> **Note**
>
> If you're using [version 2](#) of AWS CodePipeline with proper triggers set up on the `hotfix` branch, you still require a dedicated process to address unscheduled requests. In version 2, you can set up triggers for either push or pull requests. The execution will either be queued or executed immediately, depending on the previous state of the pipeline. However, with a dedicated pipeline, the fixes are applied immediately to the production environment, ensuring that urgent issues are resolved without delay.

Automate ingestion and visualization of Amazon MWAA custom metrics on Amazon Managed Grafana by using Terraform

Faisal Abdullah and Satya Vajrapu, Amazon Web Services

Summary

This pattern discusses how to use Amazon Managed Grafana to create and monitor custom metrics that are ingested by Amazon Managed Workflows for Apache Airflow (Amazon MWAA). Amazon MWAA serves as the orchestrator for workflows, employing Directed Acyclic Graphs (DAGs) that are scripted in Python. This pattern centers on the monitoring of custom metrics, including the total number of DAGs running within the last hour, the count of passed and failed DAGs each hour, and the average duration of these processes. This analysis shows how Amazon Managed Grafana integrates with Amazon MWAA to enable comprehensive monitoring and insights into the orchestration of workflows within this environment.

Prerequisites and limitations

Prerequisites

- An active AWS account with the necessary user permissions to create and manage the following AWS services:
  - AWS Identity and Access Management (IAM) roles and policies
  - AWS Lambda
  - Amazon Managed Grafana
  - Amazon Managed Workflows for Apache Airflow (Amazon MWAA)
  - Amazon Simple Storage Service (Amazon S3)
  - Amazon Timestream
- Access to a shell environment which can be a terminal on your local machine or AWS CloudShell.
- A shell environment with Git installed and the latest version of the AWS Command Line Interface (AWS CLI) installed and configured. For more information, see Installing or updating to the latest version of the AWS CLI in the AWS CLI documentation.
- The following Terraform version installed: `required_version = ">= 1.6.1, < 2.0.0"` You can use tfswitch to switch between different versions of Terraform.
- Configured identity source in AWS IAM Identity Center for your AWS account. For more information, see Confirm your identity sources in IAM Identity Center in the IAM Identity Center

documentation. You can choose from the default IAM Identity Center directory, Active Directory, or an external Identity provider (IdP) such as Okta. For more information, see Related resources.

Limitations

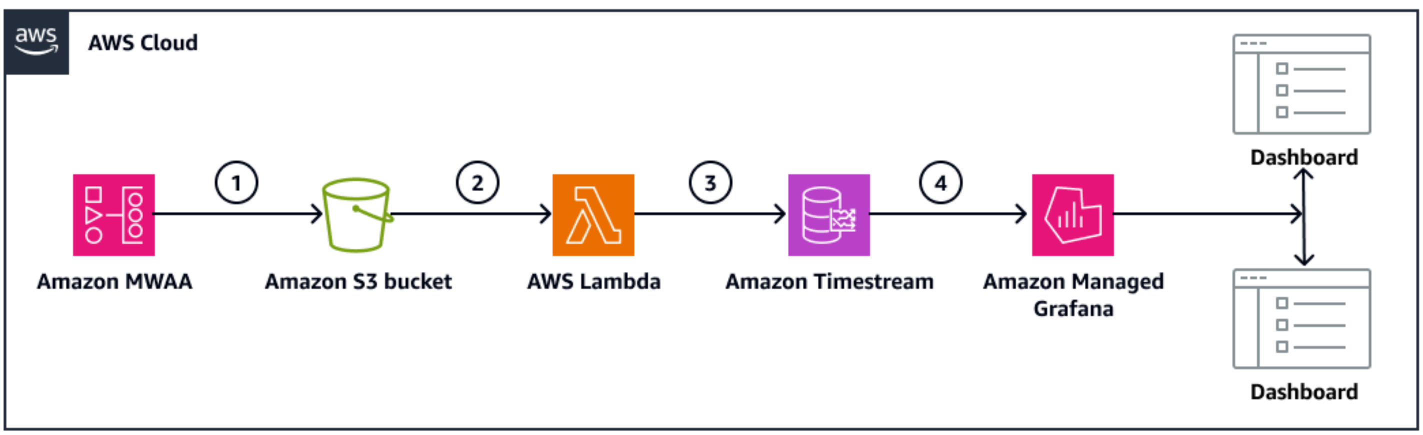
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- Terraform `required_version = ">= 1.6.1, < 2.0.0"`
- Amazon Managed Grafana version 9.4 or later. This pattern was tested on version 9.4.

Architecture

The following architecture diagram highlights the AWS services used in the solution.



The preceding diagram steps through the following workflow:

1. Custom metrics within Amazon MWAA originate from DAGs that are executing within the environment. The metrics upload to the Amazon S3 bucket in a CSV file format. The following DAGs use the database querying capabilities of Amazon MWAA:
   - `run-example-dag` – This DAG contains sample Python code that defines one or more tasks. It runs every 7 minutes and prints the date. After printing the date, the DAG includes a task to sleep, or pause, execution for a specific duration.

- `other-sample-dag` – This DAG runs every 10 mins and prints the date. After printing the date, the DAG includes a task to sleep, or pause, execution for a specific duration.
- `data-extract` – This DAG runs every hour and queries the Amazon MWAA database and collects metrics. After the metrics are collected, this DAG writes them to an Amazon S3 bucket for further processing and analysis.

2. To streamline data processing, Lambda functions run when they're triggered by Amazon S3 events, which facilitates the loading of metrics into Timestream.
3. Timestream is integrated as a data source within Amazon Managed Grafana where all the custom metrics from Amazon MWAA are stored.
4. Users can query the data and construct custom dashboards to visualize key performance indicators and gain insights into the orchestration of workflows within Amazon MWAA.

Tools

AWS services

- AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to all of your AWS accounts and cloud applications.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use. In this pattern, AWS Lambda runs the Python code in response to Amazon S3 events and manages the compute resources automatically.
- Amazon Managed Grafana is a fully managed data visualization service that you can use to query, correlate, and visualize, and alert on your metrics, logs, and traces. This pattern uses Amazon Managed Grafana to create a dashboard for metrics visualization and alerts.
- Amazon Managed Workflows for Apache Airflow (Amazon MWAA) is a managed orchestration service for Apache Airflow that you can use to set up and operate data pipelines in the cloud at scale. Apache Airflow is an open source tool used to programmatically author, schedule, and monitor sequences of processes and tasks referred to as workflows. In this pattern, sample DAGs and a metrics extractor DAG are deployed in Amazon MWAA.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data. In this pattern, Amazon S3 is used to store DAGs, scripts, and custom metrics in CSV format.
- Amazon Timestream for LiveAnalytics is is a fast, scalable, fully managed, purpose-built time series database that makes it easy to store and analyze trillions of time series data points

per day. Timestream for LiveAnalytics also integrates with commonly used services for data collection, visualization, and machine learning. In this pattern, it's used to ingest the generated Amazon MWAA custom metrics.

Other tools

- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources. This pattern uses a Terraform module to automate the provisioning of infrastructure in AWS.

Code repository

The code for this pattern is available on GitHub in the visualize-amazon-mwaa-custom-metrics-grafana repository. The `stacks/Infra` folder contains the following:

- Terraform configuration files for all AWS resources
- Grafana dashboard .json file in the `grafana` folder
- Amazon Managed Workflows for Apache Airflow DAGs in the `mwaa/dags` folder
- Lambda code to parse the .csv file and store metrics in the Timestream database in the `src` folder
- IAM policy .json files in the `templates` folder

Best practices

Terraform must store state about your managed infrastructure and configuration so that it can map real-world resources to your configuration. By default, Terraform stores state locally in a file named `terraform.tfstate`. It's crucial to ensure the safety and integrity of your Terraform state file because it maintains the current state of your infrastructure. For more information, see Remote State in the Terraform documentation.

Epics

Deploy the infrastructure using Terraform

Task	Description	Skills required

Deploy the infrastructure.	To deploy the solution infrastructure, do the following:
	1. Open a terminal or command prompt on your local computer or by using AWS CloudShell.
	2. Navigate to the directory where you want to clone the repository.
	3. To clone the repository, run the following command:
	`git clone https://github.com/aws-samples/visualize-amazon-mwaa-custom-metrics-grafana`
	4. After the cloning process is finished, run the following command to navigate into the cloned repository directory:
	`cd visualize-amazon-mwaa-custom-metrics-grafana/stacks/infra`
	5. To download and initialize the required providers, run the following command:
		AWS DevOps

Task	Description	Skills required

	`terraform init`
	
	6. To get a comprehensive view of all the resources that Terraform will create, run the following command:
	
	`terraform plan`
	
	Terraform provisions the following resources:
	• Amazon Virtual Private Cloud (Amazon VPC) and associated networking components
	• Amazon S3 resources
	• AWS Lambda functions
	• Amazon Managed Grafana resources (workspace, dashboards, data source)
	• Supporting IAM resources (roles and policies)
	
	7. To create the AWS resources from the plan output, run the following command:
	
	`terraform apply -auto-approve`	

Task	Description	Skills required
	The infrastructure provisioning completes in approximately 20 minutes. 8. To create the specified AWS resources according to the configuration defined in your Terraform files, run the following command: `terraform apply`	

Validate the deployed infrastructure resources

Task	Description	Skills required
Validate the Amazon MWAA environment.	To validate the Amazon MWAA environment, do the following: 1. Sign in to the AWS Management Console, navigate to the [Amazon MWAA dashboard console](#) and select **Open Airflow UI**. 2. You should see the following three DAGs in **Active** status: • data-extract • run-example-dag • other-sample-dag	AWS DevOps, Data engineer

Task	Description	Skills required
	3. If a DAG isn't active, you can activate it by enabling the toggle switch next to the DAG name.	
Verify the DAG schedules.	To view each DAG schedule, go to the **Schedule** tab in the **Airflow UI**. Each of the following DAGs has a pre-configured schedule, which runs in the Amazon MWAA environment and generates custom metrics: • **run-example-dag** - Runs every 7 minutes • **other-sample-dag** - Runs every 10 minutes • **data-extract** - Runs every hour You can also see the successful runs of each DAG under the **Runs** column.	Data engineer, AWS DevOps

Configure the Amazon Managed Grafana environment

Task	Description	Skills required
Configure access to the Amazon Managed Grafana workspace.	The Terraform scripts created the required Amazon Managed Grafana workspace	AWS DevOps

Task	Description	Skills required
	, dashboards, and metrics page. To configure access so that you can view them, do the following: 1. Open the [Amazon Managed Grafana](#) console. 2. In **Workspaces**, select the workspace `grafana-ws-dev`, and navigate to the **Authentication** tab in the lower pane. 3. Choose the **Assign new user or group** button. 4. Add either your group in the **Groups** tab or a user in the **Users** tab, and then choose the **Assign user and groups** button. 5. After the user (or group) is added, make this user (or group) an admin. Select the user in **Assigned users or group** in the **Assigned user group** tab and choose **Make admin** from the dropdown menu. For more information, see [Use AWS IAM Identity Center with your Amazon Managed Grafana workspace](#) in the Amazon Managed Grafana documentation.	

Task	Description	Skills required
	6. Navigate to **Workspaces**, and then choose the **Grafana workspace URL**. To sign in to Amazon Managed Grafana as an admin, choose **Sign in with AWS IAM Identity Center**.	

Task	Description	Skills required
Install the Amazon Timestream plugin.	Amazon MWAA custom metrics are loaded into the Timestream database. You use the Timestream plugin to visualize the metrics with Amazon Managed Grafana dashboards. To install the Timestream plugin, do the following: 1. In the Amazon Managed Grafana console, expand the menu in the left navigation pane and go to **Administration, Plugins**. 2. Search for and then install latest version of the Amazon Timestream plugin. 3. After the plugin is installed, go to **Administration**, **Data sources** to see the Timestream data source. If the data source isn't listed, refresh the page. For more information, see Extend your workspace with plugins in the Amazon Managed Grafana documentation.	AWS DevOps, DevOps engineer

Visualize the custom metrics in the Amazon Managed Grafana dashboard

Task	Description	Skills required
View the Amazon Managed Grafana dashboard.	To view the metrics that were ingested into the Amazon Managed Grafana workspace, do the following: 1. In the Amazon Managed Grafana console, choose **Dashboards** in the left navigation pane. 2. To view the metrics, choose **MWAA events dashboards** and then select **mwaa_metrics**. The dashboard metrics page shows the following informati on: • Total DAG runs in the last one hour • Total successful, failed, and running DAG runs in the last one hour • Average duration for all, successful, and failed DAG runs	AWS DevOps
Customize the Amazon Managed Grafana dashboard.	To customize the dashboards for further future enhanceme nts, do the following: 1. On the Amazon Managed Grafana dashboard	AWS DevOps

Task	Description	Skills required
	`mwaa_metrics` page, choose the **Dashboard settings** icon. 2. To view the data structure that defines the dashboard , choose **JSON model**. You can customize the dashboard by making edits to this JSON model directly in the console. Alternatively, the source code for this dashboard is available in the `dashboard .json` file in the `stacks/ infra/grafana` folder in the [GitHub repository]().	

Clean up AWS resources

Task	Description	Skills required
Pause the Amazon MWAA DAG runs.	To pause the DAG runs, do the following: 1. In the Amazon MWAA console, navigate to [Airflow environments]() and choose **Open Airflow UI**. 2. To pause the DAG, use the toggle switch next to each DAG. 3. Refresh the **Airflow UI** page, which should list	AWS DevOps, Data engineer

Task	Description	Skills required
	three DAGs in the **Paused** section.	
Delete the objects in the Amazon S3 buckets.	To delete the Amazon S3 buckets **mwaa-events-bucket-*** and **mwaa-metrics-bucket-***, follow the instructions for using the Amazon S3 console in [Deleting a bucket](#) in the Amazon S3 documentation.	AWS DevOps

Task	Description	Skills required
Destroy the resources created by Terraform.	To destroy the resources created by Terraform and the associated local Terraform state file, do the following: 1. (Optional) Before deleting the resources, you can preview the changes that Terraform will make. To generate a plan, run the following command: `terraform plan -destroy` The command output shows that the `destroy` command will delete all the AWS resources that were created earlier. 2. `terraform destroy -auto-approve` This command takes approximately 20 minutes to destroy the infrastru cture. **Note** To destroy all resources managed by Terraform, run the following command. : The -	AWS DevOps

Task	Description	Skills required

	`auto-approve` tag doesn't wait for user confirmation to start destroying the resources.
	3. To delete the local Terraform state file, run the following commands:
	`rm .terraform.lock.hcl`
	`rm -rf .terraform`
	`rm terraform.tfstate*`	

Troubleshooting

Issue	Solution
`null_resource.plugin_mgmt (local-exec): aws: error: argument operation: Invalid choice, valid choices are:`	Upgrade your AWS CLI to the latest version.
Loading data sources error - `Fetch error: 404 Not Found Instantiating…`	The error is intermittent. Wait a few minutes, and then refresh your data sources to view the listed Timestream data source.

Related resources

AWS documentation

- Amazon Managed Grafana for dashboarding and visualization

- Configure Amazon Managed Grafana to use Okta
- Use AWS IAM Identity Center with your Amazon Managed Grafana workspace
- Working with DAGs on Amazon MWAA

AWS videos

- Configure IAM Identity Center with Amazon Managed Grafana for authentication, as shown in the following video.

https://www.youtube-nocookie.com/embed/XX2Xcz-Ps9U?controls=0

- If IAM Identity Center isn't available, you can also integrate the Amazon Managed Grafana authentication by using an external Identity provider (IdP) such as Okta, as shown in the following video.

https://www.youtube-nocookie.com/embed/Z4JHxl2xpOg?controls=0

Additional information

You can create a comprehensive monitoring and alerting solution for your Amazon MWAA environment, enabling proactive management and rapid response to potential issues or anomalies. Amazon Managed Grafana includes the following capabilities:

Alerting – You can configure alerts in Amazon Managed Grafana based on predefined thresholds or conditions. Set up email notifications to alert relevant stakeholders when certain metrics exceed or fall below specified thresholds. For more information, see Grafana alerting in the Amazon Managed Grafana documentation.

Integration – You can integrate Amazon Managed Grafana with various third-party tools such as OpsGenie, PagerDuty, or Slack for enhanced notification capabilities. For example, you can set up webhooks or integrate with APIs to trigger incidents and notifications in these platforms based on alerts generated in Amazon Managed Grafana. In addition, this pattern provides a GitHub repository to create AWS resources. You can further integrate this code with your infrastructure deployment workflows.

Streamline Amazon Lex bot development and deployment by using an automated workflow

Balaji Panneerselvam, Attila Dancso, Pavan Dusanapudi, Anand Jumnani, and James O'Hara, Amazon Web Services

Summary

Developing and deploying Amazon Lex conversational bots can be challenging when you're trying to manage multiple features, developers, and environments. An automated workflow using infrastructure as code (IaC) principles can help streamline the process. This pattern can help improve the productivity of Amazon Lex developers and enable efficient bot lifecycle management in the following ways:

- **Enable concurrent development of multiple features** - With an automated workflow, developers can work on different features in parallel in separate branches. Changes can then be merged and deployed without blocking other work.
- **Use the Amazon Lex console UI** - Developers can use the user-friendly Amazon Lex console to build and test bots. The bots are then described in infrastructure code for deployment.
- **Promote bots across environments** - The workflow automates promoting bot versions from lower environments like development and test up to production. This approach reduces the risk and overhead of manual promotions.
- **Maintain version control** - Managing bot definitions in Git rather than solely through the Amazon Lex service provides you with version control and an audit trail. Changes are tracked to individual developers, unlike when only using the AWS Management Console or APIs to modify bots stored in AWS.

By automating the Amazon Lex bot release process, teams can deliver features faster with reduced risk and effort. Bots remain under version control rather than isolated in the Amazon Lex console.

Prerequisites and limitations

Prerequisites

- The workflow involves multiple AWS accounts for different environments (development, production, and DevOps), which requires account management and cross-account access configurations.

- Python 3.9 available in your deployment environment or pipeline.
- Git installed and configured on a local workstation for source control.
- AWS Command Line Interface (AWS CLI) installed and configured to authenticate by using the command line or Python.

Limitations

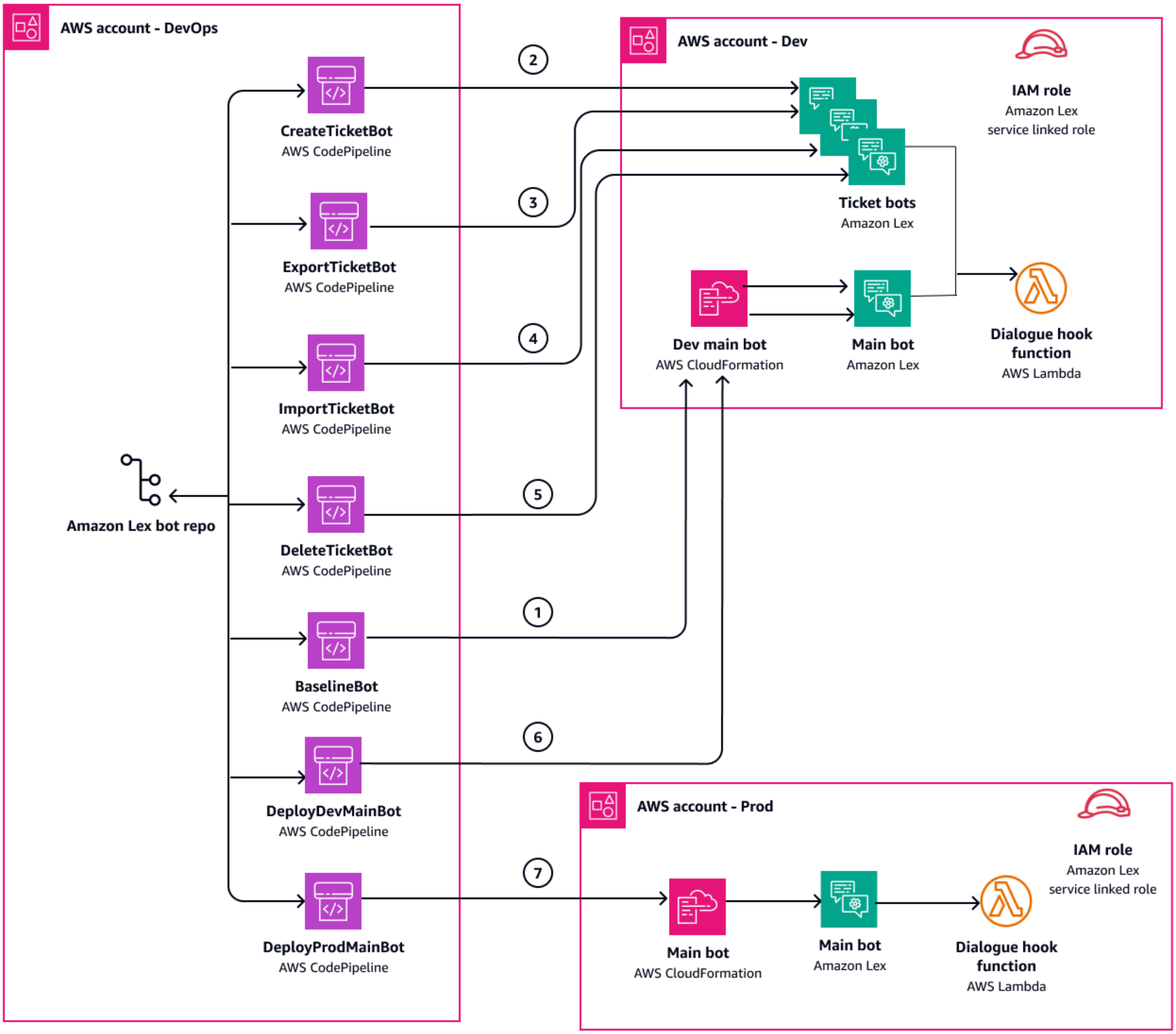
- **Repository access –** The workflow assumes that the continuous integration and continuous delivery (CI/CD) pipeline has the necessary permissions to commit changes to the source code repository.
- **Initial bot version –** The tooling requires that an initial version of the bot is deployed by using AWS CloudFormation templates. You must create the first iteration of the bot and commit it to the repo before the automated workflow can take over.
- **Merge conflicts –** Although the workflow aims to enable concurrent development, there is still a possibility of merge conflicts when integrating changes from different branches. Resolving conflicts in bot configurations might require manual intervention.

Product versions

- Python 3.9 or above
- AWS CDK v2 2.124.0 or above
- AWS SDK for Python (Boto3)1.28 or above

Architecture

The following diagram displays the high-level architecture and key components of the solution.



Key components include the following:

- **Lex bot repo –** A Git repository that stores the IaC definitions for the Amazon Lex bots.
- **DevOps –** An AWS account dedicated to housing the CI/CD pipelines and related resources for the development and deployment process.
- **Pipelines –** The AWS CodePipeline instances that automate various stages of the bot development and deployment lifecycle, such as creating a new bot, exporting a bot's definition, importing a bot definition, and deleting a bot.

- **Ticket bots and main bot –** The Amazon Lex bot resources, where the ticket bots are feature-specific bots developed by individual teams or developers and the main bot is the baseline bot that integrates all the features.

The architecture diagram illustrates the following workflow:

1. **Baseline main bot –** The starting point of the workflow is to baseline the main bot in the development (Dev) environment. The main bot serves as the foundation for future development and feature additions.
2. **Create ticket bot –** When a new feature or change is required, a ticket bot is created. The ticket bot is essentially a copy or branch of the main bot that developers can work on without affecting the main version.
3. **Export ticket bot** - After work on the ticket bot is complete, it's exported from the Amazon Lex service. Then, the branch that contains the ticket bot is rebased from the main branch. This step ensures that any changes made to the main bot while the ticket bot was in development are incorporated, reducing potential conflicts.
4. **Import rebased ticket bot and validate –** The rebased ticket bot is imported back into the development environment and validated to ensure it functions correctly with the latest changes from the main branch. If validation is successful, a pull request (PR) is created to merge the ticket bot changes into the main branch.
5. **Delete ticket bot –** After the changes have been successfully merged into the main branch, the ticket bot is no longer needed. The ticket bot can be deleted to keep the environment clean and manageable.
6. **Deploy main bot into development environment and test –** The updated main bot, now including the new features or changes, is deployed to the development environment. Here, it undergoes thorough testing to ensure all functionalities work as expected.
7. **Deploy main bot into production environment –** After testing in the development environment is complete and successful, the main bot is deployed to the production environment. This step is the final stage of the workflow, where the new features become available to end users.

Automation and scale

The automated workflow allows developers to work on different features in parallel, each in separate branches. This facilitates concurrent development, enabling teams to collaborate effectively and deliver features faster. With branches isolated from each other, changes can be merged and deployed without blocking or interfering with other ongoing work.

The workflow automates the deployment and promotion of bot versions across different environments, such as development, testing, and production.

Storing bot definitions in a version control system such as Git provides a comprehensive audit trail and enables efficient collaboration. Changes are tracked to individual developers, ensuring transparency and accountability throughout the development lifecycle. This approach also facilitates code reviews, enabling teams to identify and address issues before deploying to production.

By using AWS CodePipeline and other AWS services, the automated workflow can scale to accommodate increasing workloads and team sizes.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is an open-source software development framework for defining AWS Cloud infrastructure in code by using familiar programming languages and provisioning it through AWS CloudFormation. The sample implementation in this pattern uses Python.
- AWS CDK Command Line Interface (AWS CDK CLI) - The AWS CDK Toolkit is the primary tool for interacting with your AWS CDK app. It executes your app, interrogates the application model you defined, and produces and deploys the AWS CloudFormation templates generated by the CDK.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions. This pattern uses CloudFormation for deploying the Amazon Lex bot configurations and related resources using infrastructure as code.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy. This pattern uses CodeBuild for building and packaging the deployment artifacts.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously. This pattern uses CodePipeline to orchestrate the continuous delivery pipeline.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact withAWS services through commands in your command line shell.

- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [Amazon Lex V2](#) is an AWS service for building conversational interfaces (bots) for applications using voice and text.
- [AWS SDK for Python (Boto3)](#) is a software development kit that helps you integrate your Python application, library, or script with AWS services.

Other tools

- [Git](#) is an open source distributed version control system.

Code repository

The code for this pattern is available in the GitHub [management-framework-sample-for-amazon-lex](#) repository. The code repo contains the following folders and files:

- `prerequisite` folder – Contains CloudFormation stack definitions (using the AWS CDK) for setting up the required resources and environments.
- `prerequisite/lexmgmtworkflow` folder – Main directory for the Lex Management Workflow project, including stack definitions and Python code.
- `prerequisite/tests` – Contains unit tests.
- `src` folder – Source code directory, including Amazon Lex bot management wrapper and utilities.
- `src/dialogue_lambda` – Source code directory of the dialogue hook Lambda function that intercepts and processes user inputs during a conversation with an Amazon Lex bot.

Best practices

- **Separation of concerns**
  - Maintain a clear separation of responsibilities between the DevOps, development, and production environments.
  - Use separate AWS accounts for each environment to enforce proper isolation and security boundaries.

  - Use cross-account roles and least-privilege access principles to ensure controlled access between environments.
- **Infrastructure as code**
  - Regularly review and update the infrastructure code to align with best practices and evolving requirements.
  - Establish a clear branching and merging strategy for the source code repository
- **Testing and validation**
  - Implement automated testing at various stages of the pipeline to catch issues early in the development cycle.
  - Use the Amazon Lex console or automated testing frameworks to validate bot configurations and functionality before promoting to higher environments.
  - Consider implementing manual approval gates for deployments to production or critical environments.
- **Monitoring and logging**
  - Set up monitoring and logging mechanisms for the pipelines, deployments, and bot interactions.
  - Monitor pipeline events, deployment statuses, and bot performance metrics to identify and address issues promptly.
  - Use AWS services such as Amazon CloudWatch, AWS CloudTrail, and AWS X-Ray for centralized logging and monitoring.
  - Regularly review and analyze the performance, efficiency, and effectiveness of the automated workflow.
- **Security and compliance**
  - Implement secure coding practices and follow AWS security best practices for Amazon Lex bot development and deployment.
  - Regularly review and update IAM roles, policies, and permissions to align with the principle of least privilege.
  - Consider integrating security scanning and compliance checks into the pipelines.

Epics

Set up IaC for Amazon Lex bot management

Task	Description	Skills required

Set up the local CDK environment.	1. To clone this pattern's repo and navigate to the `prerequisite` directory, run the following command:
	`git clone https://github.com/aws-samples/management-framework-sample-for-amazon-lex.git`
	
	`cd management-framework-sample-for-amazon-lex`
	2. To install and activate the Python virtual environment, run the following command that installs the CDK's dependencies locally in the project folder, instead of globally:
	`pip install virtualenv`
	`python<version> -m venv .venv`
	`source .venv/bin/activate`
	`python -m pip install -r requirements.txt`
		AWS DevOps

Task	Description	Skills required

Create a cross-account role in the `devops` environment.	The `devops` account is responsible for hosting and managing the CI/CD pipelines. To enable the CI/CD pipelines to interact with the `dev` and `prod` environments, run the following commands to create a cross-account role in the `devops` account.
	
	`cdk bootstrap --profile =devops`
	
	`cdk deploy LexMgmtDevopsRoleStack -c dev-account-id=2222222222222 -c prod-account-id=333333333333 --profile=devops`
		AWS DevOps

Task	Description	Skills required

Create a cross-account role in the `dev` environment.	Create an IAM role in the `dev` account with the necessary permissions to allow the `devops` account to assume this role. The CI/CD pipeline uses this role to perform actions in the `dev` account, such as deploying and managing Amazon Lex bot resources.
	
	To create the IAM role, run the following commands:
	
	`cdk bootstrap --profile=dev`
	
	`cdk deploy LexMgmtCrossaccountRoleStack -c devops-account-id=1111111111111 --profile=dev`
		AWS DevOps

Task	Description	Skills required

Create a cross-account role in the `prod` environment.	Create an IAM role in the `prod` account with the necessary permissions to allow the `devops` account to assume this role. The CI/CD pipeline uses this role to perform actions in the `prod` account, such as deploying and managing Amazon Lex bot resources.
	
	`cdk bootstrap --profile=prod`
	
	`cdk deploy LexMgmtCrossaccountRoleStack -c devops-account-id=1111111111111 --profile=prod`
		AWS DevOps

Create pipelines in the `devops` environment.	To manage the development workflow for Amazon Lex bots, run the following command to set up pipelines in the `devops` environment . `cdk deploy LexMgmtWorkflowStack -c devops-account-id=1111111111111 -c dev-account-id=2222222222222 -c prod-account-id=333333333333 --profile=devops`	AWS DevOps

Establish the baseline for the main bot

Task	Description	Skills required
Define the initial version of the main bot.	To define the initial version of the main bot, [trigger](#) the `BaselineBotPipeline` pipeline. The pipeline deploys the basic bot definition that's defined in the CloudFormation template, exports the main bot definition as .json files. and stores the main bot code in a version control system.	AWS DevOps

Implement the feature development workflow

Task	Description	Skills required
Create the ticket bot to develop and test a feature.	`TicketBot` is a new bot instance that's imported from the existing main bot definition in the feature branch. This approach ensures that the new bot has all the current functionality and configurations from the main bot. To define the initial version of the ticket bot, trigger the `CreateTicketBotPipeline` pipeline.	Lex Bot Developer

Task	Description	Skills required
	The pipeline creates a new feature branch in the version control system and creates a new ticket bot instance based on the main bot.	
Develop and test the ticket bot feature.	To develop and test the feature, sign in to the AWS Management Console and open the Amazon Lex console at https://console.aws.amazon.com/lex/. For more information, see [Testing a bot using the console](#) in the Amazon Lex documentation. With the `TicketBot` instance, you can now add, modify, or extend the bot's functionality to implement the new feature. For example, you can create or modify intents, utterances, slots, and dialog flows. For more information, see [Adding intents](#) in the Amazon Lex documentation.	Lex Bot Developer

Task	Description	Skills required
Export the ticket bot definitio n.	The exported bot definition is essentially a representation of the bot's configuration and functionality in a JSON format. To export the ticket bot definition, trigger the `ExportTicketBotPipeline` pipeline. The pipeline exports the ticket bot definition as .json files and stores the ticket bot code in a feature branch in the version control system.	Lex Bot Developer
Rebase the feature branch from the latest main branch.	During the development of a new feature, the main branch might have received other changes from different developers or teams. To incorporate these changes into the feature branch, perform a Git `rebase` operation. This operation essentially replays the commits from the feature branch on top of the latest commits from the main branch, ensuring that the feature branch includes all the latest changes	Lex Bot Developer

Task	Description	Skills required
Import and validate the rebased ticket bot.	After you rebase the feature branch, you must import it into the ticket bot instance. This import updates the existing ticket bot with the latest changes from the rebased branch. To import the rebased ticket bot, trigger the `ImportTicketBotPipeline` pipeline. The pipeline imports the ticket bot definition .json files in the feature branch in the version control system into the `TicketBot` instance.	Lex Bot Developer

Task	Description	Skills required
Validate the rebased bot definition.	After you import the rebased bot definition, it's crucial to validate its functionality. You want to make sure that the new feature works as expected and doesn't conflict with existing functionality. This validation typically involves testing the bot with various input scenarios, checking the responses, and verifying that the bot behaves as intended. You can perform validation in either of the following ways: • Test the bot manually by using the Amazon Lex console. • Use an automated approach by using testing frameworks and tools that can simulate user interacti ons and assert expected responses.	Lex Bot Developer

Task	Description	Skills required
Merge the feature branch into the main branch.	After you develop and test the new feature in the isolated `TicketBot` instance, do the following: 1. Commit the changes to the corresponding feature branch in the version control system. 2. To merge the feature branch into the main branch, create a pull request (PR). This PR serves as a request to review and incorporate the changes into the main codebase.	Lex Bot Developer, Repository Adminstrator

Task	Description	Skills required
Delete the feature branch and the ticket bot.	After a feature branch is merged successfully into the main branch, delete the feature branch and the ticket bot from the source code repo. To delete the feature branch and the ticket bot, trigger the `DeleteTicketBotPipeline` pipeline. The pipeline removes temporary bot resources that were created during the development process (for example, the ticket bot). This action helps to maintain a clean repo and prevent confusion or conflicts with future feature branches.	Lex Bot Developer

Maintain the main bot

Task	Description	Skills required
Import the latest main bot definition into the `dev` environment.	To import the latest main bot definition in the main branch into the `dev` environment, trigger the `DeployBotDevPipeline` pipeline. The pipeline also creates a git tag on approval.	AWS DevOps

Task	Description	Skills required
Import the latest main bot definition into the `prod` environment.	To import the latest bot definition in the main branch into the `prod` environment, provide the tag reference from the previous task as a parameter and trigger the `DeployBotProdPipeline` pipeline. The pipeline imports the latest bot definition from a specific tag into the `prod` environment.	AWS DevOps

Troubleshooting

Issue	Solution
When you deploy Amazon Lex bots to different AWS accounts, the tooling services must have the necessary permissions to access resources in those accounts.	To grant cross-account access, use IAM roles and policies. Create IAM roles in the target accounts and attach policies to the roles that grant the required permissions. Then, assume these roles from the account where the Amazon Lex bot is deployed. For more information, see IAM permissions required to import and IAM permissions required to export bots in Lex V2 in the Amazon Lex documentation.

Related resources

- Importing bots in Amazon Lex V2
- Start a pipeline in CodePipeline

- Working with Amazon Lex V2 bots

Automate stack set deployment by using AWS CodePipeline and AWS CodeBuild

Thiyagarajan Mani, Mihir Borkar, and Raghu Gowda, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

In your continuous integration and continuous delivery (CI/CD) processes, you might want to deploy applications automatically into all your existing AWS accounts and into new accounts that you add to your organization in AWS Organizations. When you architect a CI/CD solution for this requirement, the delegated stack set administrator capability of AWS CloudFormation is useful because it enables a layer of security by restricting access to the management account. However, AWS CodePipeline uses the service-managed permissions model to deploy applications into multiple acounts and Regions. You must use the AWS Organizations management account to deploy with stack sets, because AWS CodePipeline doesn't support the delegated stack set administrator feature.

This pattern describes how you can work around this limitation. The pattern uses AWS CodeBuild and a custom script to automate stack set deployment with AWS CodePipeline. It automates these application deployment activities:

- Deploy an application as stack sets into existing organizational units (OUs)
- Extend the deployment of an application into additional OUs and Regions
- Remove a deployed application from all or specific OUs or Regions

Prerequisites and limitations

Prerequisites

Before you follow the steps in this pattern:

- Create organizations in your AWS Organizations management account. For instructions, see the AWS Organizations documentation.

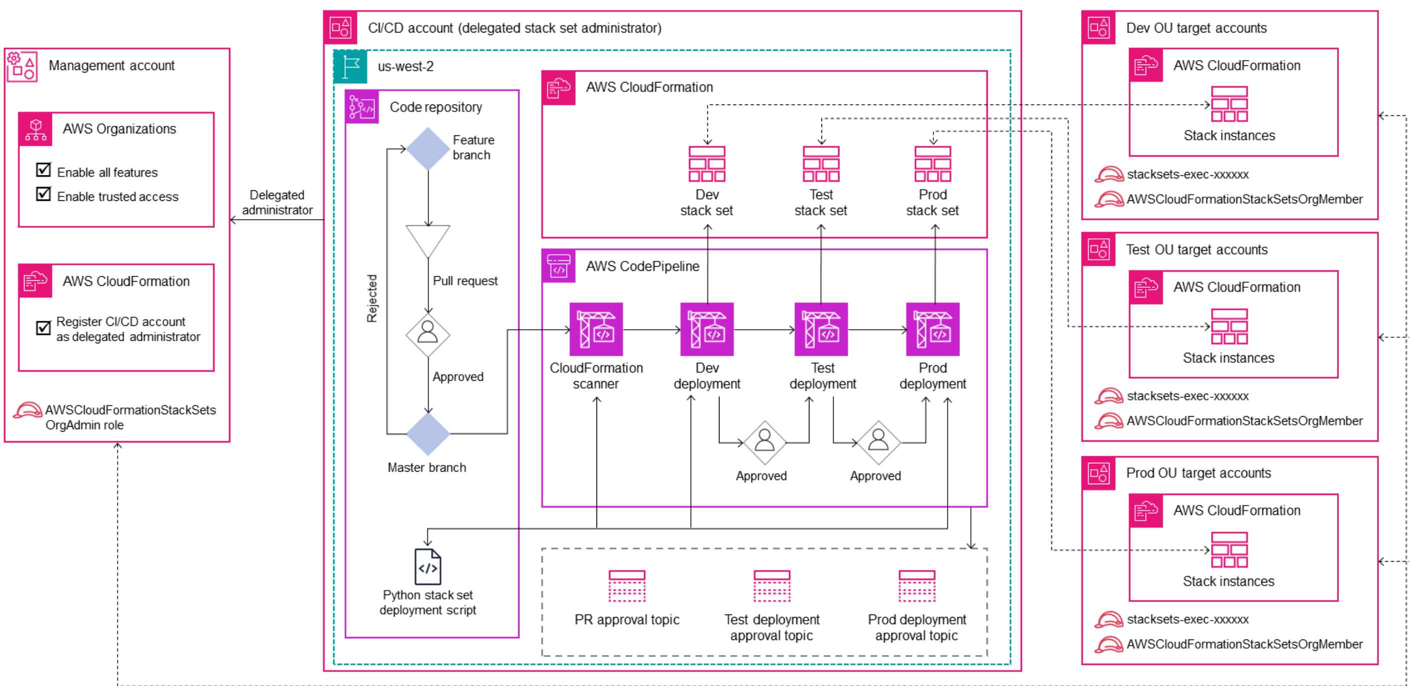
- Enable trusted access between AWS Organizations and CloudFormation to use service-managed permissions. For instructions, see [Enable trusted access with AWS Organizations](#) in the CloudFormation documentation.

Limitations

The code that's supplied with this pattern has the following limitations:

- You can deploy only a single CloudFormation template for an application; multiple template deployment isn't currently supported.
- Customizing the current implementation requires DevOps expertise.
- This pattern doesn't use AWS Key Management System (AWS KMS) keys. However, you can enable this functionality by reconfiguring the CloudFormation template included with this pattern.

Architecture



This architecture for the CI/CD deployment pipeline handles the following:

- Restricts direct access to the management account by delegating stack set deployment responsibility to a dedicated CI/CD account as the stack set administrator for application deployments.
- Uses the service-managed permission model to deploy the application automatically whenever a new account is created and mapped under an OU.
- Ensures application version consistency across all accounts at the environment level.
- Uses multiple approval stages at the repository and pipeline levels to provide additional layers of security and governance for the deployed application.
- Overcomes the current limitation of CodePipeline by using a custom-built deployment script in CodeBuild to automatically deploy or remove stack sets and stack instances. For an illustration of the flow control and hierarchy of API calls implemented by the custom script, see the Additional information section.
- Creates individual stack sets for the development, testing, and production environments. In addition, you can create stack sets that combine multiple OUs and Regions at every stage. For example, you can combine sandbox and development OUs within a development deployment stage.
- Supports application deployment into, or exclusion from, a subset of accounts or list of OUs.

Automation and scale

You can use the code provided with this pattern to create a AWS CodeCommit repository and a code pipeline for your application. You can then deploy these as stack sets into multiple accounts at the OU level. The code also automates components such as Amazon Simple Notification Service (Amazon SNS) topics to notify approvers, the required AWS Identity and Access Management (IAM) roles, and the service control policy (SCP) to apply in the management account.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.

- AWS CodeDeploy automates deployments to Amazon Elastic Compute Cloud (Amazon EC2) or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

Code repository

The code for this pattern is available in the GitHub automated-code-pipeline-stackset-deployment repository. For the folder structure and other details, see the readme file for the repository.

Best practices

This pattern restricts direct access to the management account while deploying the application at the OU level. Adding multiple approval stages to the pipeline and repository process helps provide additional security and governance for the applications and components that you deploy by using this approach.

Epics

Configure accounts in AWS Organizations

Task	Description	Skills required
Enable all features in the management account.	Enable all features in the management account for your organization by following the instructions in the AWS Organizations documentation.	AWS administrator, Platform administrator
Create a CI/CD account.	In AWS Organizations, in your organization, create a	AWS administrator

Task	Description	Skills required
	dedicated CI/CD account, and assign a team to own and control access to the account.	
Add a delegated administrator.	In the management account, register the CI/CD account that you created in the previous step as a delegated stack set administrator. For instructions, see the [AWS CloudFormation documentation]().	AWS administrator, Platform administrator

Create an application repository and CI/CD pipeline

Task	Description	Skills required
Clone the code repository.	1. Clone the code repository that's provided with this pattern to your computer: `git clone https://github.com/aws-samples/automated-code-pipeline-stackset-deployment.git` 2. Review the [readme file]() to understand the directory structure and other details.	AWS DevOps
Create SNS topics.	You can use the `sns-template.yaml` template that's provided in the GitHub repository to create SNS	AWS DevOps

Task	Description	Skills required
	topics and configure subscriptions for approval requests. 1. On the AWS console, sign in to the CI/CD account. 2. Open the CloudFormation console at at https://console.aws.amazon.com/cloudformation. 3. Create a new stack with new resources (standard option). 4. For **Specify template**, choose **Upload a template file**, **Choose file**, and then select the `sns-template.yaml` file from the `templates` folder of the cloned GitHub repository. Choose **Next**. 5. Provide a meaningful application stack name. 6. Specify a prefix for resources. 7. Choose **Next**, **Next**, and **Submit**. 8. When the stack has been created successfully, choose the **Outputs** tab, and note the Amazon Resource Names (ARNs) of the SNS topics for pull requests, the test environment, and the	

Task	Description	Skills required
	production environment. You'll use this information in subsequent steps.	

Task	Description	Skills required
Create IAM roles for CI/CD components.	You can use the `cicd-role-template.yaml` template that's provided in the GitHub repository to create IAM roles and policies required by CI/CD components. 1. On the AWS console, sign in to the CI/CD account. 2. Open the CloudFormation console at at https://console.aws.amazon.com/cloudformation. 3. Create a new stack with new resources (standard option). 4. For **Specify template**, choose **Upload a template file**, **Choose file**, and then select the `cicd-role-template.yaml` file from the `templates` folder of the cloned GitHub repository. Choose **Next**. 5. Provide a meaningful application stack name. 6. Enter values for the following parameters: • The ARN for the permission boundary policy. You can obtain this ARN from the	AWS DevOps

Task	Description	Skills required
	Policy details section of your permissions boundary policy on the IAM console. • The ARN for the SNS production approval topic that you noted previously. • The ARN for the SNS test approval topic that you noted previously. • A prefix for resources created by the template. 7. Choose **Next**, **Next**, and **Submit**. 8. When the stack has been created successfully, choose the **Outputs** tab, and note the ARNs of the IAM roles that were created. You'll use this information in subsequent steps.	

Task	Description	Skills required
Create a CodeCommit repository and a code pipeline for your application.	You can use the `cicd-pipeline-template.yaml` template that's provided in the GitHub repository to create a CodeCommit repository and a code pipeline for your application. 1. On the AWS console, sign in to the CI/CD account. 2. Open the CloudFormation console at at https://console.aws.amazon.com/cloudformation. 3. Create a new stack with new resources (standard option). 4. For **Specify template**, choose **Upload a template file**, **Choose file**, and then select the `cicd-pipeline-template.yaml` file from the `templates` folder of the cloned GitHub repository. Choose **Next**. 5. Provide a meaningful application stack name. 6. Enter values for the following parameters: • **AppRepositoryName** – The name of the CodeCommit repository	AWS DevOps

Task	Description	Skills required
	that will be created for the application. • **AppRepositoryDescription –** A brief description of the CodeCommit repository that will be created for the application. • **ApplicationName –** The name of your application. This string is used as the name of the CodeCommit repository and as the prefix of the CI/CD pipeline. • **CloudWatchEventRoleARN –** The ARN of the CloudWatch event role from the previous task. • **CodeBuildProjectRoleARN –** The ARN of the CodeBuild project role from the previous task. • **CodePipelineRoleARN –** The ARN of the CodePipeline role from the previous task. • **DeploymentConfigBucket –** The Amazon Simple Storage Service (Amazon S3) bucket name where the deployment configura	

Task	Description	Skills required
	tion files and script .zip file will be stored. • **DeploymentConfigKey** – The path and .zip filename (Amazon S3 key). • **PRApprovalSNSARN** – The ARN for the SNS topic for pull request notifications. • **ProdApprovalSNSARN** – The ARN for the SNS topic for production approvals. • **TESTApprovalSNSARN** – The ARN for the SNS topic for test approvals. • **TemplateBucket** – The name of the S3 bucket in the CI/CD account where the CI/CD pipeline creation template will be stored. 7. Choose **Next**, **Next**, and **Submit**. 8. When the stack completes successfully, it creates a CodeCommit repository that has the specified name and a default directory structure, deployment configuration	

Task	Description	Skills required
	files, scripts, and a code pipeline for the repository.	

Deploy a stack set

Task	Description	Skills required

Clone the application repository.	The CI/CD pipeline template you used previously creates a sample application repositor y and code pipeline. To clone and verify the repository:
	
	1. Sign in to the CI/CD account.
	2. Find the application repository and CI/CD pipeline that you created in the previous epic.
	3. Copy the URL for the repository and use the git clone command to clone the repository on your local machine.
	4. Verify that the directory structure and files match the following:
	
	`root`
	` |- deploy_configs`
	` |        |- deploymen t_config.json`
	` |- parameters`
	` |      |- template- parameter-dev.json`
		App developer, Data engineer

Task	Description	Skills required
	``` | |- template- parameter-test.json | |- template- parameter-prod.json |- templates | |- template. yml |- buildspec.yml ``` where the `deploy_configs` folder contains the deployment configuration file, and the `templates` and `parameters` folders include default files that you'll replace with your own CloudFormation template and parameter files. **Important** Do not customize the folder structure. 5. Create a feature branch.	

Task	Description	Skills required
Add application artifacts.	Update the application repository by using a CloudFormation template. **Note** This solution supports the deploymen t of only a single CloudFormation template. 1. Build your CloudFormation template for deploying your application code changes, and name it `<application-name>.yaml`. 2. Replace the `template.yml` file in the `templates` folder of the application repository with the CloudFormation template you created in step 1. 3. Prepare parameter files for each environment (development, testing, and production). 4. Name the parameter files by using the format `<cloudformation-template-name>-param`	App developer, Data engineer

Task	Description	Skills required
	`eter-<environment-name>.json` . 5. Replace the default parameter files in the `parameters` folder with your files from step 4.	

Task	Description	Skills required
Update the deployment configuration file.	Update the `deployment_config.json` file: 1. In the application repository, navigate to the `deploy_configs` folder. 2. Open the file `deployment_config.json` :	App developer, Data engineer

```
{
    "deployment_action": "<deploy/delete>",
    "stack_set_name": "<stack set name>",
    "stack_set_desciption": "<stack set description>",
    "deployment_targets": {

      "dev": {


 "org_units": ["list of OUs"],


 "regions": ["list of regions"],


 "filter_accounts": ["list of accounts"],


    "filter_type":
```

Task	Description	Skills required

	``
	 "<DIFFERENCE/INTER
	SECTION/UNION>"
	
	              },
	
	      "test": {
	
	 "org_units": ["list
	 of OUs"],
	
	 "regions": ["list of
	 regions"],
	
	 "filter_accounts":
	 ["list of accounts"
	],
	
	    "filter_type":
	 "<DIFFERENCE/INTER
	SECTION/UNION>"
	
	              },
	
	      "prod": {
	
	 "org_units": ["list
	 of OUs"],
	
	 "regions": ["list of
	 regions"],
	
	 "filter_accounts":
	 ["list of accounts"
	],
	``	

Task	Description	Skills required

	``
	    "filter_type": "<DIFFERENCE/INTERSECTION/UNION>"
	            }
	    },
	    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
	    "auto_deployement": "<True/False>",
	    "retain_stacks_on_account_removal": "<True/False>",
	    "region_deployment_concurrency": "<SEQUENTIAL/PARALLEL>"
	}
	``
	3. Update the values for deployment action, stack set name, stack set description, and deployment targets.
	
	For example, you can set `deployment_action` to `delete` to delete the entire stack set and its associated stack instances . Use `deploy` to create a new stack set, to update	

Task	Description	Skills required
	an existing stack set, or to add or remove stack instances for additional OUs or Regions. For more examples, see the [Additional information](#) section. This pattern creates individual stack sets for each environment by adding the environment name to the stack set name you provide in the deployment configuration file.	

Task	Description	Skills required
Commit changes and deploy the stack set.	Commit the changes you specified in your applicati on template, and merge and deploy the stack set into multiple environments stage by stage: 1. Save all your files and commit changes to the feature branch of your local application repository. 2. Push the feature branch to the remote repository. 3. Create a pull request to merge the changes to the main branch. When the pull request has been approved and changes have been merged to the main branch, the CI/ CD pipeline will be initiated . 4. When the Dev-deployment stage has completed successfully, check the CloudFormation console, **StackSets**, **Service-M anaged** tab. You'll see a new stack set with the suffix `dev`. 5. Check the CodeBuild logs for the Dev-deployment stage for any issues.	App developer, Data engineer

Task	Description	Skills required
	6. Deploy the stack set into the testing and productio n environments by asking your approvers to approve the deployments for those stages and repeating steps 5 and 6. The stack sets for the testing and productio n environments have the suffixes `test` and `prod`.	

Troubleshooting

Issue	Solution
Deployment fails with the exception: *Change the Name of Template Parameter file as <application name>-parameter-<evn>.json with, default names are not allowed*	The CloudFormation template parameter files must follow the naming convention specified . Update the parameter file names and try again.
Deployment fails with the exception: *Change the Name of CloudFormation Template as <application name>.yml, default template. yml or template.yaml are not allowed*	The CloudFormation template name must follow the naming convention specified. Update the file name and try again.
Deployment fails with the exception: *No valid CloudFormation Template and its Parameter File found for {environment name} environment*	Check the file naming conventions for the CloudFormation template and its parameter file for the specified environment.
Deployment fails with the exception:	You specified an invalid value for the `deployment_action` parameter in the deployment configuration file. The parameter

Issue	Solution
Invalid deployment action provided in deployment config file. Valid options are 'deploy' and 'delete'.	has two valid values: `deploy` and `delete`. Use `deploy` to create and update the stack sets and their associated stack instances. Use `delete` only when you want to remove the entire stack set and associated stack instances.

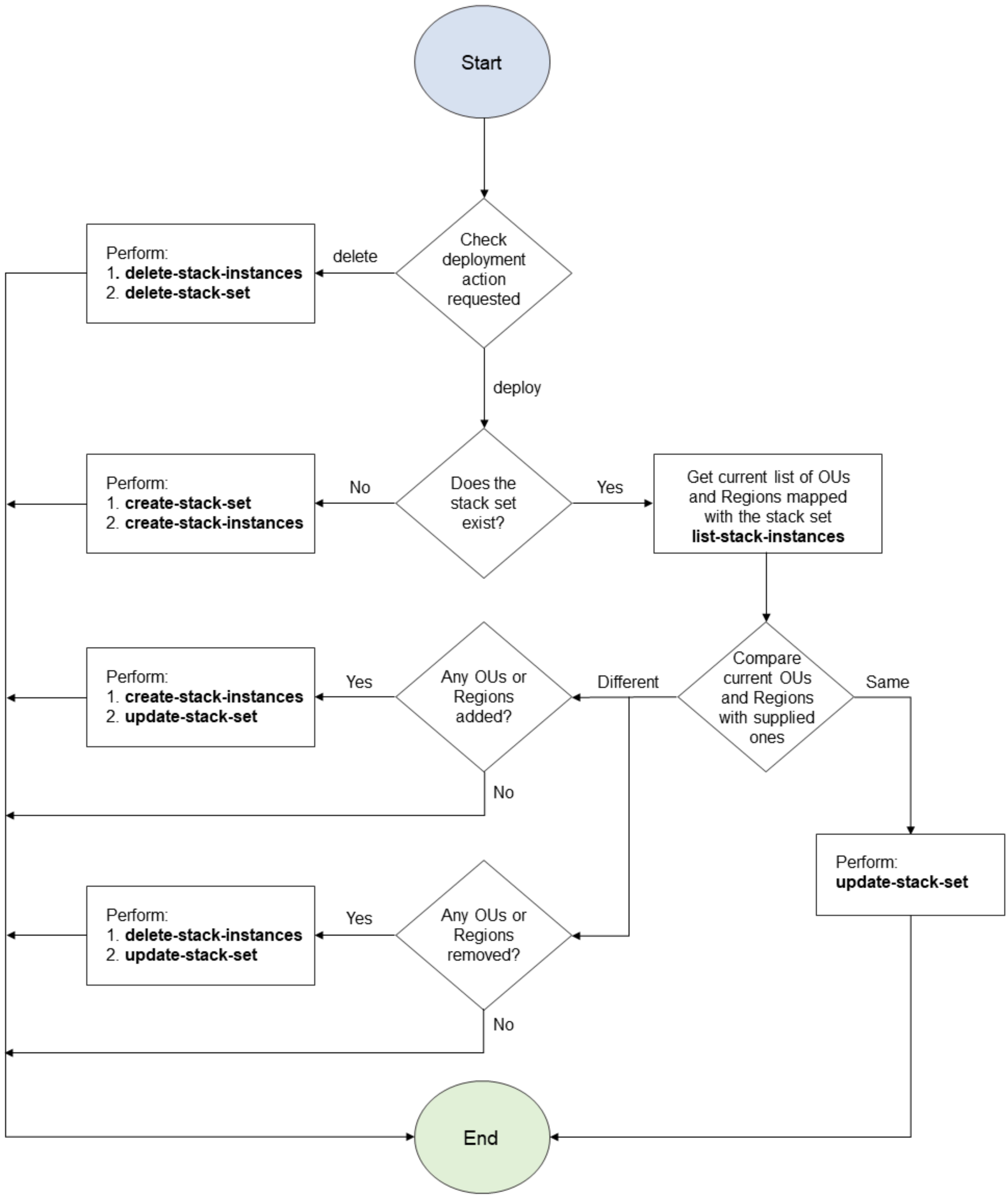
Related resources

- GitHub automated-code-pipeline-stackset-deployment repository
- Enabling all features in your organization (AWS Organizations documentation)
- Register a delegated administrator (AWS CloudFormation documentation)
- Account level targets for service-managed Stack Sets (AWS CloudFormation documentation)

Additional information

Flow chart

The following flow chart depicts the flow control and hierarchy of API calls implemented by the custom script to automate stack set deployment.



Sample deployment configuration files

Creating a new stack set

The following deployment configuration file creates a new stack set called `sample-stack-set` in the AWS Region `us-east-1` in three OUs.

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                           "dev": {
                                       "org_units": ["dev-org-unit-1"],
                                       "regions": ["us-east-1"],
                                       "filter_accounts": [],
                                       "filter_type": ""
                                   },
                           "test": {
                                       "org_units": ["test-org-unit-1"],
                                       "regions": ["us-east-1"],
                                       "filter_accounts": [],
                                       "filter_type": ""
                                   },
                           "prod": {
                                       "org_units": ["prod-org-unit-1"],
                                       "regions": ["us-east-1"],
                                       "filter_accounts": [],
                                       "filter_type": ""
                                   }
                         },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

Deploying an existing stack set to another OU

If you deploy the configuration shown in the previous example and you want to deploy the stack set to an additional OU called `dev-org-unit-2` in the development environment, the deployment configuration file might look like the following.

```
{
     "deployment_action": "deploy",
```

```
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                            "dev": {
                                        "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "test": {
                                        "org_units": ["test-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "prod": {
                                        "org_units": ["prod-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    }
                          },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

Deploying an existing stack set to another AWS Region

If you deploy the configuration shown in the previous example and you want to deploy the stack set to an additional AWS Region (`us-east-2`) in the development environment for two OUs (`dev-org-unit-1` and `dev-org-unit-2`), the deployment configuration file might look like the following.

> **Note**
>
> The resources in the CloudFormation template must be valid and Region-specific.

```
{
```

```
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                        "dev": {
                                    "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                    "regions": ["us-east-1", "us-east-2"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                        "test": {
                                    "org_units": ["test-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                        "prod": {
                                    "org_units": ["prod-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                }
                      },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

Removing a stack instance from an OU or AWS Region

Let's say that the deployment configuration shown in the previous example has been deployed. The following configuration file removes the stack instances from both Regions of the OU `dev-org-unit-2`.

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                        "dev": {
                                    "org_units": ["dev-org-unit-1"],
```

```
                                        "regions": ["us-east-1", "us-east-2"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "test": {
                                        "org_units": ["test-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "prod": {
                                        "org_units": ["prod-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    }
                          },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

The following configuration file removes the stack instance from the AWS Region `us-east-1` for both OUs in the development environment.

```
{
     "deployment_action": "deploy",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                            "dev": {
                                        "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                        "regions": ["us-east-2"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "test": {
                                        "org_units": ["test-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
```

```
                                        "filter_type": ""
                                    },
                            "prod": {
                                        "org_units": ["prod-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    }
                          },
     "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
     "auto_deployement": "True",
     "retain_stacks_on_account_removal": "True",
     "region_deployment_concurrency": "PARALLEL"
 }
```

Deleting the entire stack set

The following deployment configuration file deletes the entire stack set and all its associated stack instances.

```
{
     "deployment_action": "delete",
     "stack_set_name": "sample-stack-set",
     "stack_set_desciption": "this is a sample stack set",
    "deployment_targets": {
                            "dev": {
                                        "org_units": ["dev-org-unit-1", "dev-org-
unit-2"],
                                        "regions": ["us-east-2"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "test": {
                                        "org_units": ["test-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
                                    },
                            "prod": {
                                        "org_units": ["prod-org-unit-1"],
                                        "regions": ["us-east-1"],
                                        "filter_accounts": [],
                                        "filter_type": ""
```

```
                                    }
                            },
    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
    "auto_deployement": "True",
    "retain_stacks_on_account_removal": "True",
    "region_deployment_concurrency": "PARALLEL"
}
```

Excluding an account from deployment

The following deployment configuration file excludes the account 111122223333, which is part of the OU `dev-org-unit-1`, from deployment.

```
{
    "deployment_action": "deploy",
    "stack_set_name": "sample-stack-set",
    "stack_set_desciption": "this is a sample stack set",
   "deployment_targets": {
                              "dev": {
                                          "org_units": ["dev-org-unit-1"],
                                          "regions": ["us-east-1"],
                                          "filter_accounts": ["111122223333"],
                                          "filter_type": "DIFFERENCE"
                                      },
                              "test": {
                                          "org_units": ["test-org-unit-1"],
                                          "regions": ["us-east-1"],
                                          "filter_accounts": [],
                                          "filter_type": ""
                                      },
                              "prod": {
                                          "org_units": ["prod-org-unit-1"],
                                          "regions": ["us-east-1"],
                                          "filter_accounts": [],
                                          "filter_type": ""
                                      }
                            },
    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
    "auto_deployement": "True",
    "retain_stacks_on_account_removal": "True",
    "region_deployment_concurrency": "PARALLEL"
}
```

Deploying the application to a subset of accounts in an OU

The following deployment configuration file deploys the application to only three accounts (111122223333, 444455556666, and 777788889999) in the OU `dev-org-unit-1`.

```
{
    "deployment_action": "deploy",
    "stack_set_name": "sample-stack-set",
    "stack_set_desciption": "this is a sample stack set",
   "deployment_targets": {
                        "dev": {
                                    "org_units": ["dev-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": ["111122223333",
"444455556666", "777788889999"],
                                    "filter_type": "INTERSECTION"
                                },
                        "test": {
                                    "org_units": ["test-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                },
                        "prod": {
                                    "org_units": ["prod-org-unit-1"],
                                    "regions": ["us-east-1"],
                                    "filter_accounts": [],
                                    "filter_type": ""
                                }
                      },
    "cft_capabilities": ["CAPABILITY_IAM", "CAPABILITY_NAMED_IAM"],
    "auto_deployement": "True",
    "retain_stacks_on_account_removal": "True",
    "region_deployment_concurrency": "PARALLEL"
}
```

Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK

Ali Asfour and Aaron Lennon, Amazon Web Services

Summary

You can integrate Amazon Elastic Compute Cloud (Amazon EC2) instances with AWS Systems Manager to automate operational tasks and provide more visibility and control. To integrate with Systems Manager, EC2 instances must have an installed [AWS Systems Manager Agent (SSM Agent)](#) and an `AmazonSSMManagedInstanceCore` AWS Identity and Access Management (IAM) policy attached to their instance profiles.

However, if you want to ensure that all EC2 instance profiles have the `AmazonSSMManagedInstanceCore` policy attached, you can face challenges updating new EC2 instances that don't have instance profiles or EC2 instances that have an instance profile but don't have the `AmazonSSMManagedInstanceCore` policy. It can also be difficult to add this policy across multiple Amazon Web Services (AWS) accounts and AWS Regions.

This pattern helps solve these challenges by deploying three [Cloud Custodian](#) policies in your AWS accounts:

- The first Cloud Custodian policy checks for existing EC2 instances that have an instance profile but don't have the `AmazonSSMManagedInstanceCore` policy. The `AmazonSSMManagedInstanceCore` policy is then attached.
- The second Cloud Custodian policy checks for existing EC2 instances without an instance profile and adds a default instance profile that has the `AmazonSSMManagedInstanceCore` policy attached.
- The third Cloud Custodian policy creates [AWS Lambda functions](#) in your accounts to monitor the creation of EC2 instances and instance profiles. This ensures that the `AmazonSSMManagedInstanceCore` policy is automatically attached when an EC2 instance is created.

This pattern uses [AWS DevOps](#) tools to achieve a continuous, at-scale deployment of the Cloud Custodian policies to a multi-account environment, without provisioning a separate compute environment.

Prerequisites and limitations

Prerequisites

- Two or more active AWS accounts. One account is the *security account* and the others are *member accounts*.
- Permissions to provision AWS resources in the security account. This pattern uses administrator permissions, but you should grant permissions according to your organization's requirements and policies.
- Ability to assume an IAM role from the security account to member accounts and create the required IAM roles. For more information about this, see Delegate access across AWS accounts using IAM roles in the IAM documentation.
- > **Important**
  >
  > AWS Command Line Interface (AWS CLI), installed and configured. For testing purposes, you can configure AWS CLI by using the `aws configure` command or setting environment variables. : This isn't recommended for production environments and we recommend that this account is only granted least privilege access. For more information about this, see Grant least privilege in the IAM documentation.
- The `devops-cdk-cloudcustodian.zip` file (attached), downloaded to your local computer.
- Familiarity with Python.
- The required tools (Node.js, AWS Cloud Development Kit (AWS CDK), and Git), installed and configured. You can use the `install-prerequisites.sh` file in the `devops-cdk-cloudcustodian.zip` file to install these tools. Make sure you run this file with root privileges.

Limitations

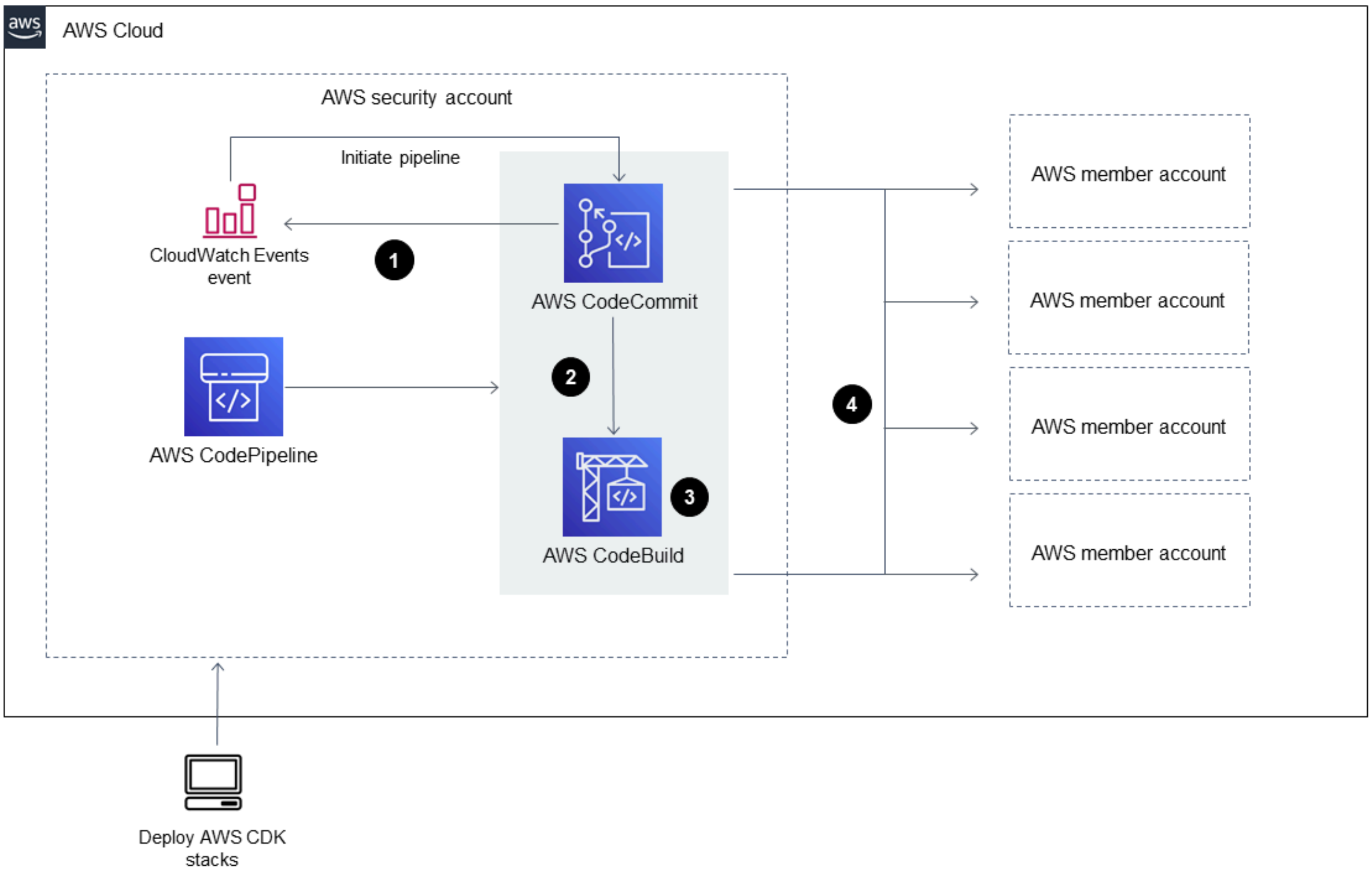
- Although this pattern can be used in a production environment, make sure that all IAM roles and policies meet your organization's requirements and policies.

Package versions

- Cloud Custodian version 0.9 or later
- TypeScript version 3.9.7 or later
- Node.js version 14.15.4 or later

- npm version 7.6.1 or later
- AWS CDK version 1.96.0 or later

Architecture



The diagram shows the following workflow:

1. Cloud Custodian policies are pushed to an AWS CodeCommit repository in the security account. An Amazon CloudWatch Events rule automatically initiates the AWS CodePipeline pipeline.
2. The pipeline fetches the most recent code from CodeCommit and sends it to the continuous integration part of the continuous integration and continuous delivery (CI/CD) pipeline handled by AWS CodeBuild.
3. CodeBuild performs the complete DevSecOps actions, including policy syntax validation on the Cloud Custodian policies, and runs these policies in `--dryrun` mode to check which resources are identified.

4. If there are no errors, the next task alerts an administrator to review the changes and approve the deployment into the member accounts.

Technology stack

- AWS CDK
- CodeBuild
- CodeCommit
- CodePipeline
- IAM
- Cloud Custodian

Automation and scale

The AWS CDK pipelines module provisions a CI/CD pipeline that uses CodePipeline to orchestrate the building and testing of source code with CodeBuild, in addition to the deployment of AWS resources with AWS CloudFormation stacks. You can use this pattern for all member accounts and Regions in your organization. You can also extend the `Roles creation` stack to deploy other IAM roles in your member accounts.

Tools

- AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.
- AWS CodeBuild is a fully managed build service in the cloud.
- AWS CodeCommit is a version control service that you can use to privately store and manage assets.
- AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software.
- AWS Identity and Access Management is a web service that helps you securely control access to AWS resources.
- Cloud Custodian is a tool that unifies the dozens of tools and scripts most organizations use for managing their public cloud accounts into one open-source tool.

- Node.js is a JavaScript runtime built on Google Chrome's V8 JavaScript engine.

Code

For a detailed list of modules, account functions, files, and deployment commands used in this pattern, see the `README` file in the `devops-cdk-cloudcustodian.zip` file (attached).

Epics

Set up the pipeline with AWS CDK

Task	Description	Skills required
Set up the CodeCommit repository.	1. Unzip the `devops-cdk-cloudcustodian.zip` file (attached) in the working directory on your local computer. 2. Sign in to the AWS Management Console for your security account, open the CodeCommit console, and then create a new `devops-cdk-cloudcustodian` repository. 3. Change into the project directory and set up the CodeCommit repository as the origin, commit the changes, and then push them to the origin branch by running the following commands: • `cd devops-cdk-cloudcustodian`	Developer

Task	Description	Skills required
	• `git init --initial-branch=main` • `git add . git commit -m 'initial commit'` • `git remote add origin https://git-codecommit.us-east-1.amazonaws.com/v1/devops-cdk-cloudcustodian` • `git push origin main` For more information about this, see Creating a CodeCommit repository in the AWS CodeCommit documentation.	
Install the required tools.	Use the `install-prerequisites.sh` file to install all the required tools on Amazon Linux. This doesn't include AWS CLI because it comes pre-installed. For more information about this, see the Prerequisites section of Getting started with the AWS CDK in the AWS CDK documentation.	Developer

Task	Description	Skills required
Install the required AWS CDK packages.	1. Set up your virtual environment by running the following command in AWS CLI: `$ python3 -m venv .env` 2. Activate your virtual environment by running the following command: `$ source .env/bin/activate` 3. After the virtual environment is activated, install the required dependencies by running the following command: `$ pip install -r requirements.txt` 4. To add additional dependencies (for example, other AWS CDK libraries), add them to the `requirements.txt` file, and then run the following command: `pip install -r requirements.txt` The following packages are required by AWS CDK and are included in the `requirements.txt` file: • `aws-cdk.aws-cloudwatch`	Developer

Task	Description	Skills required
	• `aws-cdk.aws-codebuild` • `aws-cdk.aws-codecommit` • `aws-cdk.aws-codedeploy` • `aws-cdk.aws-codepipeline` • `aws-cdk.aws-codepipeline-actions` • `aws-cdk.aws-events` • `aws-cdk.aws-events-targets` • `aws-cdk.aws-iam` • `aws-cdk.aws-logs` • `aws-cdk.aws-s3` • `aws-cdk.aws-sns` • `aws-cdk.aws-sns-subscriptions` • `aws-cdk.aws-sqs` • `aws-cdk.core`	

Configure your environment

Task	Description	Skills required
Update the required variables.	Open the `vars.py` file in the root folder of your CodeCommit repository and update the following variables:	Developer

Task	Description	Skills required
	• Update `var_deploy_region = 'us-east-1'` with the AWS Region where you want the pipeline to be deployed. • Update `var_codecommit_repo_name = "cdk-cloudcustodian"` with the name of your CodeCommit repository. • Update `var_codecommit_branch_name = "main"` with name of the CodeCommit branch. • Update `var_adminEmail=notifyadmin@email.com'` with the email address for the administrator that approves changes. • Update `var_slackWebHookUrl = https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXX '` with the Slack webhook used to send Cloud Custodian notifications when changes are made.	

Task	Description	Skills required
	• Update `var_orgId = 'o-yyyyyyyyyy'` with your organization ID. • Update `security_account = '123456789011'` with the AWS account ID for the account where the pipeline is deployed. • Update `member_accounts = ['111111111111','111111111112','111111111113']` with the member accounts where you want to bootstrap the AWS CDK stack and deploy the required IAM roles. • Set `cdk_bootstrap_member_accounts = True` to `True` if you want the pipeline to automatically bootstrap the AWS CDK to your member accounts. If set to `True` this also requires the name of an existing IAM role in the member accounts that can be assumed from the security account. This IAM role must also have the required permissions to bootstrap the AWS CDK.	

Task	Description	Skills required
	• **Note** Update `cdk_bootstrap_role = 'AWSControlTowerExecution'` with the existing IAM role in the member accounts that can be assumed from the security account. This role must also permission to bootstrap the AWS CDK. : This only applies if `cdk_bootstrap_member_accounts` is set to `True`.	

Task	Description	Skills required

Update the account.yml file with the member account information.	To run the [c7n-org Cloud Custodian](https://cloudcustodian.io/docs/tools/c7n-org.html) tool against multiple accounts, you must place the `accounts.yml` config file in the root of the repository. The following is a sample Cloud Custodian config file for AWS:
	`accounts:`
	`- account_id: '123123123123'`
	`  name: account-1`
	`  regions:`
	`  - us-east-1`
	`  - us-west-2`
	`  role: arn:aws:iam::123123123123:role/CloudCustodian`
	`  vars:`
	`    charge_code: xyz`
	`  tags:`
	`  - type:prod`
	`  - division:some division`
	`  - partition:us`
	`  - scope:pci`
		Developer

Bootstrap the AWS accounts

Task	Description	Skills required
Boostrap the security account.	Bootstrap the `deploy_account` with the `cloudcustodian_stack` applicati	Developer

Task	Description	Skills required
	on by running the following command: `cdk bootstrap -a 'python3 cloudcustodian/cloudcustodian_stack.py`	
Option 1 - Automatically bootstrap the member accounts.	If the `cdk_bootstrap_member_accounts` variable is set to `True` in the `vars.py` file, the accounts specified in the `member_accounts` variable are automatically bootstrapped by the pipeline. If required, you can update `*cdk_bootstrap_role*` with an IAM role that you can assume from the security account and that has the required permissions to bootstrap the AWS CDK. New accounts added to the `member_accounts` variable are automatically bootstrapped by the pipeline so that the required roles can be deployed.	Developer

Task	Description	Skills required

Option 2 - Manually bootstrap the member accounts.	Although we don't recommend using this approach, you can set the value of `cdk_bootstrap_member_accounts` to `False` and perform this step manually by running the following command:
	
	`$ cdk bootstrap -a 'python3 cloudcustodian/member_account_roles_stack.py' \`
	
	`--trust {security_account_id} \`
	
	`--context assume-role-credentials:writeIamRoleName={role_name} \`
	
	`--context assume-role-credentials:readIamRoleName={role_name} \`
	
	`--mode=ForWriting \`
	
	`--context bootstrap=true \`
	
	`--cloudformation-execution-policies arn:aws:iam::aws:policy/AdministratorAccess`
		Developer

Task	Description	Skills required
	Important Make sure that you update the `{security_account_id}` and `{role_name}` values with the name of an IAM role that you can assume from the security account and that has the required permissions to bootstrap the AWS CDK. You can also use other approaches to bootstrap the member accounts, for example, with AWS CloudFormation. For more information about this, see [Bootstrapping]() in the AWS CDK documentation.	

Deploy the AWS CDK stacks

Task	Description	Skills required
Create the IAM roles in the member accounts.	Run the following command to deploy the `member_account_roles_stack`	Developer

Task	Description	Skills required
	stack and create the IAM roles in the member accounts: `cdk deploy --all -a 'python3 cloudcustodian/member_account_roles_stack.py' --require-approval never`	
Deploy the Cloud Custodian pipeline stack.	Run the following command to create the Cloud Custodian `cloudcustodian_stack.py` pipeline that is deployed into the security account: `cdk deploy -a 'python3 cloudcustodian/cloudcustodian_stack.py'`	Developer

Related resources

- Getting started with the AWS CDK

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK

Varsha Raju, Amazon Web Services

Summary

This pattern describes how to automatically create the continuous integration and continuous delivery (CI/CD) pipelines and underlying infrastructure for building and deploying microservices on Amazon Elastic Container Service (Amazon ECS). You can use this approach if you want to set up proof-of-concept CI/CD pipelines to show your organization the benefits of CI/CD, microservices, and DevOps. You can also use this approach to create initial CI/CD pipelines that you can then customize or change according to your organization's requirements.

The pattern's approach creates a production environment and non-production environment that each have a virtual private cloud (VPC) and an Amazon ECS cluster configured to run in two Availability Zones. These environments are shared by all your microservices and you then create a CI/CD pipeline for each microservice. These CI/CD pipelines pull changes from a source repository in AWS CodeCommit, automatically build the changes, and then deploy them into your production and non-production environments. When a pipeline successfully completes all of its stages, you can use URLs to access the microservice in the production and non-production environments.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account.
- An existing Amazon Simple Storage Service (Amazon S3) bucket that contains the `starter-code.zip` file (attached).
- AWS Cloud Development Kit (AWS CDK), installed and configured in your account. For more information about this, see [Getting started with the AWS CDK] in the AWS CDK documentation.
- Python 3 and `pip`, installed and configured. For more information about this, see the [Python documentation].
- Familiarity with AWS CDK, AWS CodePipeline, AWS CodeBuild, CodeCommit, Amazon Elastic Container Registry (Amazon ECR), Amazon ECS, and AWS Fargate.
- Familiarity with Docker.
- An understanding of CI/CD and DevOps.

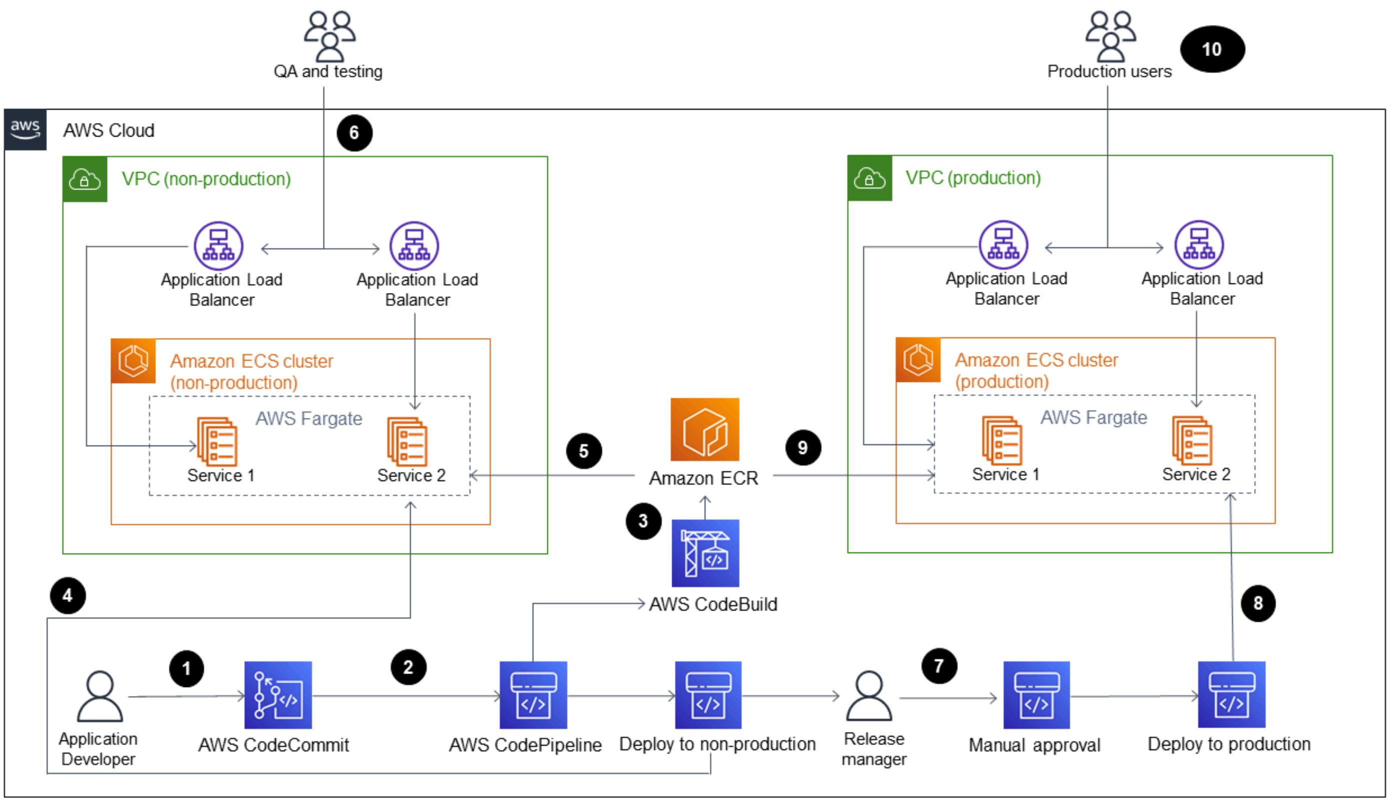
Limitations

- General AWS account limits apply. For more information about this, see [AWS service quotas](#) in the AWS General Reference documentation.

Product versions

- The code was tested using Node.js version 16.13.0 and AWS CDK version 1.132.0.

Architecture



The diagram shows the following workflow:

1. An application developer commits code to a CodeCommit repository.
2. A pipeline is initiated.
3. CodeBuild builds and pushes the Docker image to an Amazon ECR repository
4. CodePipeline deploys a new image to an existing Fargate service in a non-production Amazon ECS cluster.

5. Amazon ECS pulls the image from the Amazon ECR repository into a non-production Fargate service.
6. Testing is performed using a non-production URL.
7. The release manager approves the production deployment.
8. CodePipeline deploys the new image to an existing Fargate service in a production Amazon ECS cluster
9. Amazon ECS pulls the image from the Amazon ECR repository into the production Fargate service.

10Production users access your feature by using a production URL.

Technology stack

- AWS CDK
- CodeBuild
- CodeCommit
- CodePipeline
- Amazon ECR
- Amazon ECS
- Amazon VPC

Automation and scale

You can use this pattern's approach to create pipelines for microservices deployed in a shared AWS CloudFormation stack. The automation can create more than one Amazon ECS cluster in each VPC and also create pipelines for microservices deployed in a shared Amazon ECS cluster. However, this requires that you provide new resource information as inputs to the pipeline stack.

Tools

- AWS CDK – AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- AWS CodeBuild – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.

- AWS CodeCommit – AWS CodeCommit is a version control service that enables you to privately store and manage Git repositories in the AWS Cloud. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure.
- AWS CodePipeline – AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. CodePipeline automates the steps required to release your software changes continuously.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that is used for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage.
- Docker – Docker helps developers to pack, ship, and run any application as a lightweight, portable, and self-sufficient container.

Code

The code for this pattern is available in the `cicdstarter.zip` and `starter-code.zip` files (attached).

Epics

Set up your environment

Task	Description	Skills required
Set up the working directory for AWS CDK.	1. Create a directory named `cicdproject` on your local machine. 2. Download the `cicdstarter.zip` file (attached) into the `cicdproject` directory and unzip it. This creates a folder named `cicdstarter`.	AWS DevOps, Cloud infrastructure

Task	Description	Skills required
	3. Run the `cd <user-home>/cicdproject/cicdstarter` command. 4. Set up the Python virtual environment by running the `python3 -m venv .venv` command. 5. Run the `source ./.venv/bin/activate` command. 6. Configure your AWS environment by running the `aws configure` command or by using the following environment variables: • `AWS_ACCESS_KEY_ID` • `AWS_SECRET_ACCESS_KEY` • `AWS_DEFAULT_REGION`	

Create the shared infrastructure

Task	Description	Skills required
Create the shared infrastructure.	1. In your working directory, run the `cd cicdvpcecs` command. 2. Run the `pip3 install -r requirements.txt` command to install	AWS DevOps, Cloud infrastructure

Task	Description	Skills required
	all required Python dependencies 3. Run the `cdk bootstrap` command to set the AWS environment for the AWS CDK. 4. Run the `cdk synth --context aws_account=<aws_account_ID> --context aws_region=<aws-region>` command. 5. Run the `cdk deploy --context aws_account=<aws_account_ID> --context aws_region=<aws-region>` command. 6. The AWS CloudFormation stack creates the following infrastructure: • A non-production VPC named `cicd-vpc-ecs/cicd-vpc-nonprod` • A production VPC named `cicd-vpc-ecs/cicd-vpc-prod` • A non-production Amazon ECS cluster named `cicd-ecs-nonprod`	

Task	Description	Skills required
	• A production Amazon ECS cluster named `cicd-ecs-prod`	
Monitor the AWS CloudFormation stack.	1. Sign in to the AWS Management Console, open the AWS CloudFormation console, and then choose the `cicd-vpc-ecs` stack from the list. 2. In the stack details pane, choose the **Events** tab and monitor the progress of your stack creation.	AWS DevOps, Cloud infrastructure

Task	Description	Skills required
Test the AWS CloudFormation stack.	1. After the `cicd-vpc-ecs` AWS CloudFormation stack is created, ensure that the `cicd-vpc-ecs/cicd-vpc-nonprod` and `cicd-vpc-ecs/cicd-vpc-prod` VPCs are created. 2. Ensure that the `cicd-ecs-nonprod` and `cicd-ecs-prod` Amazon ECS clusters are created. **Important** Make sure that you record the IDs for the two VPCs and the security group IDs for the default security groups in both VPCs.	AWS DevOps, Cloud infrastructure

Create a CI/CD pipeline for a microservice

Task	Description	Skills required
Create the infrastructure for the microservice.	1. Name your microservice. For example, this pattern uses `myservice1` as the microservice's name. 2. In your working directory run the `cd <working-`	AWS DevOps, Cloud infrastructure

Task	Description	Skills required
	directory>/cdkpipeline command. 3. Run the `pip3 install -r requirements.txt` command. 4. Run the full `cdk synth` command that is available in the *Additional information* section of this pattern. 5. Run the full `cdk deploy` command that is available in the *Additional information* section of this pattern. **Note** You can also provide the values for both commands by using the `cdk.json` file in the directory.	
Monitor the AWS CloudFormation stack.	Open the AWS CloudFormation console and monitor the progress of the `myservice1-cicd-stack` stack. Eventually, the status changes to `CREATE_COMPLETE`.	AWS DevOps, Cloud infrastructure

Task	Description	Skills required
Test the AWS CloudFormation stack.	1. On the AWS CodeCommit console, verify that a repository named `myservice1` exists and contains the starter code. 2. On the AWS CodeBuild console, verify that a build project named `myservice1` exists. 3. On the Amazon ECR console, verify that an Amazon ECR repository named `myservice1` exists. 4. On the Amazon ECS console, verify that a Fargate service named `myservice1` exists in both a non-production and production Amazon ECS cluster. 5. On the Amazon Elastic Compute Cloud (Amazon EC2) console, verify that the non-production and production Application Load Balancers are created. Record the DNS names of the ALBs. 6. On the AWS CodePipeline console, verify that a pipeline named `myservice1` exists. It	

Task	Description	Skills required
	must have `Source`, `Build`, `Deploy-NonProd`, and `Deploy-Prod` stages. The pipeline should also have an `in progress` status. 7. Monitor the pipeline until all stages are complete. 8. Manually approve it for production. 9. In a browser window, enter the DNS names of the ALBs. 10. The application should display `Hello World` in the non-production and production URLs.	

Task	Description	Skills required
Use the pipeline.	1. Open the CodeCommit repository that you created earlier and open the `index.js` file. 2. Replace `Hello World` with `Hello CI/CD`. 3. Save and commit the changes to the main branch. 4. Verify that the pipeline initiates and that the change goes through the `Build`, `Deploy-NonProd` , and `Deploy-Prod` stages. 5. Manually approve the production. 6. Both production and non-production URLs should now display `Hello CICD`.	AWS DevOps, Cloud infrastructure
Repeat this epic for each microservice.	Repeat the tasks in this epic to create a CI/CD pipeline for each of your microservices.	AWS DevOps, Cloud infrastructure

Related resources

- [Using Python with AWS CDK]
- [AWS CDK Python reference]
- [Creating an AWS Fargate service using the AWS CDK]

Additional information

cdk synth command

```
cdk synth --context aws_account=<aws_account_number> --context
 aws_region=<aws_region> --context vpc_nonprod_id=<id_of_non_production
 VPC> --context vpc_prod_id=<id_of_production_VPC> --context
 ecssg_nonprod_id=< default_security_group_id_of_non-production_VPC>
 --context ecssg_prod_id=<default_security_group_id_of_production_VPC>
 --context code_commit_s3_bucket_for_code=<S3 bucket name> --context
 code_commit_s3_object_key_for_code=<Object_key_of_starter_code> --context
 microservice_name=<name_of_microservice>
```

cdk deploy command

```
cdk deploy --context aws_account=<aws_account_number> --context
 aws_region=<aws_region> --context vpc_nonprod_id=<id_of_non_production_VPC>
 --context vpc_prod_id=<id_of_production_VPC> --context ecssg_nonprod_id=<
 default_security_group_id_of_non-production_VPC> --context
 ecssg_prod_id=<default_security_group_id_of_production_VPC> --
context code_commit_s3_bucket_for_code=<S3 bucket name> --context
 code_commit_s3_object_key_for_code=<Object_key_of_starter_code> --context
 microservice_name=<name_of_microservice>
```

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9

Alexandre Nardi, Amazon Web Services

Summary

Notice: AWS Cloud9 is no longer available to new customers. Existing customers of AWS Cloud9 can continue to use the service as normal. Learn more

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern demonstrates how to develop a typical web application in a serverless architecture, for developers and development leads who are beginning to test DevOps practices on Amazon Web Services (AWS). It builds a sample application that creates a storefront and backend for browsing and purchasing books, and provides a microservice that can be developed independently. The pattern uses AWS Cloud9 as a development environment, an Amazon DynamoDB database as a data store, and AWS services such as AWS CodePipeline and AWS CodeBuild for continuous integration and continuous deployment (CI/CD) functionality.

The pattern guides you through the following development activities:

- Creating a standard AWS Cloud9 development environment
- Using AWS CloudFormation templates to create a web application and a microservice for books
- Using AWS Cloud9 to modify the front-end, commit changes, and test changes
- Creating and testing a CI/CD pipeline to the microservice
- Automating unit tests

The code for this pattern is provided in GitHub, in the AWS DevOps End-to-End Workshop repository.

Prerequisites and limitations

Prerequisites

- An active AWS account

- Files from the AWS DevOps End-to-End Workshop downloaded to your computer

> **Important**
>
> Building this demo application in your AWS account creates and consumes AWS resources. You are responsible for the cost of the AWS services and resources used to create and run the application. After you finish your work, be sure to remove all resources to avoid ongoing charges. For cleanup instructions, see the *Epics* section.

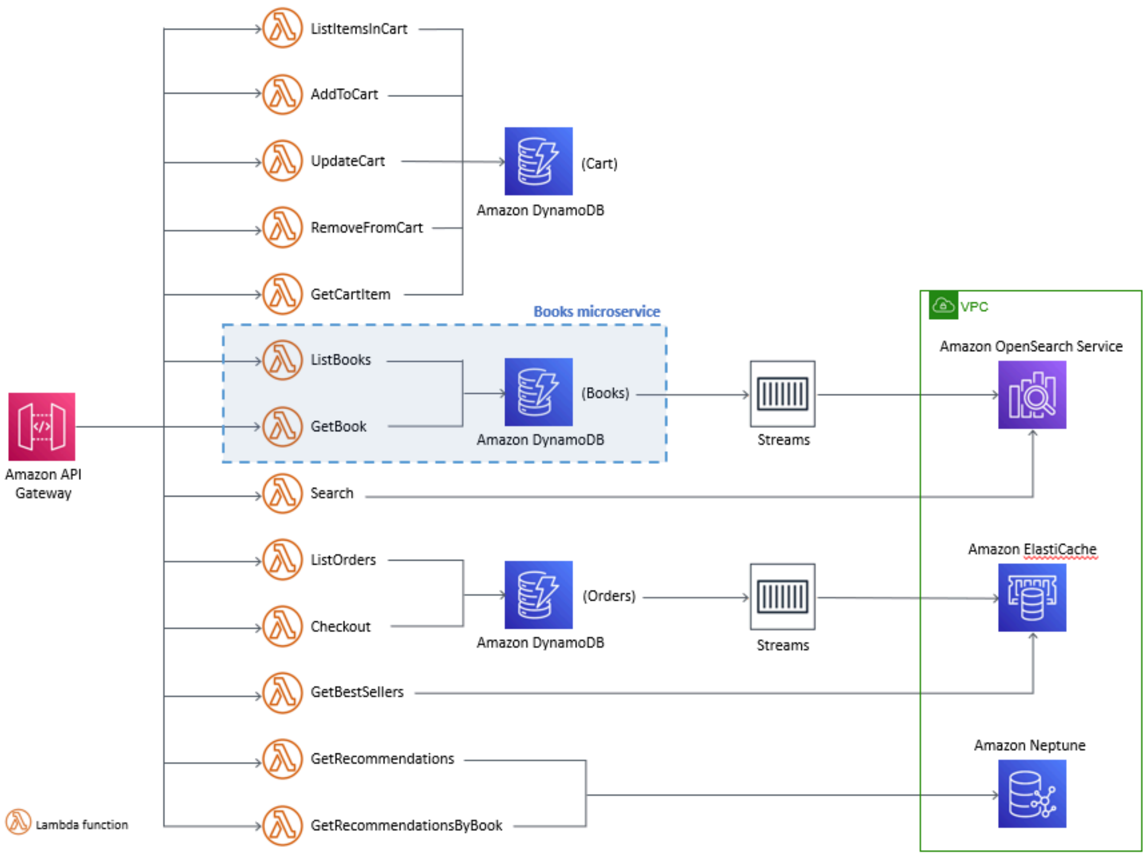
Limitations

This walkthrough is intended for demonstration and development purposes only. To use it in a production environment, see Security best practices in the AWS Identity and Access Management (IAM) documentation, and make the necessary changes to IAM roles, Amazon DynamoDB, and other services used. The web application is derived from the AWS Bookstore Demo App; for additional considerations, see the Known limitations section of the README file.

Architecture

The architecture of the bookstore application is illustrated in the Architecture section of the README file for the AWS Bookstore Demo App.

From a deployment perspective, the Bookstore Demo App uses a single CloudFormation template to deploy all services and objects in one stack. This pattern makes a few changes to demonstrate how a particular developer or team could work in a specific product (Books), and update it independently from the rest of the application. For this reason, the code for this pattern separates the AWS Lambda functions and related objects for the Books microservice into a second CloudFormation template, which creates a Books stack. That makes it possible to see the microservice being updated by using CI/CD practices. In the following diagram, the dashed border identifies the Books microservice.



Tools

Tools

- Jest framework for JavaScript testing
- Python 3.9

Code

The source code and templates for this pattern are available on GitHub, in the AWS DevOps End-to-End Workshop repository. Before you follow the steps in the *Epics* section, download all the files from the repository to your computer.

> **Note**
>
> The *Epics* section provides the high-level steps for this walkthrough, to give you general information about the process. To complete each step, see the README file in the AWS DevOps End-to-End Workshop repository for detailed instructions.

The AWS DevOps End-to-End Workshop repository extends the AWS Bookstore Demo App repository and uses a modified version of the AWS Cloud9 Bootstrapping code to create the AWS Cloud9 IDE.

Best practices

Using the Bookstore application is straightforward. Here are some recommended best practices:

- When you install the application, you can use a project name of your choice or use the default name (`demobookstore`) for convenience.
- After you have the application up and running, it's a good practice to shut down the Amazon Neptune database if you want to keep testing for another day, because the database instance could result in additional charges. However, be aware that the database will automatically be started after seven days.
- For code details, see the documentation for the AWS Bookstore Demo App repository. It describes each microservice and table.
- For additional best practices, see the *Some challenges if you have time...* section of the README file in the AWS DevOps End-to-End Workshop repository. We recommend that you review the information to dive deep into additional features for security and to practice decoupling services.

Epics

Download the source code

Task	Description	Skills required
Download the source code from GitHub.	The source code and templates for this pattern are available in GitHub, in the AWS DevOps End-to-End Workshop repository. Before you follow the next steps in the *Epics* section, download all the files from the repository to your computer.	App developer

Task	Description	Skills required
	Note The *Epics* section provides the high-level steps for this walkthrough, to give you general information about the process. To complete each step, see the README file in the AWS DevOps End-to-End Workshop repository for detailed instructions. The AWS DevOps End-to-End Workshop repository extends the AWS Bookstore Demo App repository and uses a modified version of the AWS Cloud9 Bootstrapping code to create the AWS Cloud9 IDE.	

Build the Bookstore web application and the Books microservice

Task	Description	Skills required
Create the front-end and Lambda functions for the Bookstore app.	1. Log in to the CloudFormation console, and deploy the `DemoBookstoreMainTemplate.yml` template to create	Developer

Task	Description	Skills required
	the DemoBookStoreStack stack. This creates the front-end and Lambda functions that are outside the Books microservice. 2. In the **Outputs** tab of the stack, note the website URL under the **WebApplication** label.	
Create the Books microservice.	On the [CloudFormation console](https://console.aws.amazon.com/cloudformation/), deploy the `DemoBookstoreBooksServiceTemplate.yml` template to create the DemoBooksServiceStack stack.	Developer
Test your application.	Use the website URL from the DemoBookStoreStack stack to access the Bookstore application.	Developer

Use the Cloud9 environment to maintain your application

Task	Description	Skills required
Create an AWS Cloud9 IDE.	On the [CloudFormation console](https://console.aws.amazon.com/cloudformation/), deploy the `C9EnvironmentTemplate.yml` template to create an AWS Cloud9 environment.	Developer, Developer lead
Create CodeCommit repositories.	1. Log in to the [AWS CodeCommit console](https://console.aws.amazon.com/codecommit/),	Developer

Task	Description	Skills required
	and verify that you have a `demobookstore-WebAssets` repository, which contains the code for the front-end application. 2. Create a repository for the Books microservice called `demobookstore-BooksService` . 3. Clone the two repositories in AWS Cloud9 (`demobookstore-WebAssets` and `demobookstore-BooksService`) by using the `git clone` command.	
Change the code in the frontend and check the pipeline.	1. Use AWS Cloud9 to make some code changes on a webpage. This will update the `demobookstore-WebAssets` repository. 2. On the [AWS CodePipeline console](https://console.aws.amazon.com/codepipeline/), verify that **demobookstore-Assets-Pipeline** is running. 3. Test your web application by refreshing it from the browser (**Ctrl+F5** on Firefox).	Developer

Implement a CI/CD pipeline for the Books microservice

Task	Description	Skills required
Add the YAML files for the build and service update.	1. In AWS Cloud9, upload the `buildspec.yml` and `DemoBookstoreBooksServiceUpdateTemplate.yml` files. • `buildspec.yml` has building instructions, and also includes testing instructions for automated tests. They are commented at this point, and will be used later. • `DemoBookstoreBooksServiceUpdateTemplate.yml` is an updated version of `DemoBookstoreBooksServiceTemplate.yml` , to be used in the deployment stage of the pipeline. 2. Commit and push the files.	Developer
Create an S3 bucket for the build pipeline.	To create an S3 bucket, follow the instructions in the [Amazon S3 documentation](). • The bucket name must be globally unique; for example, `demobookstore-books-service`	Developer

Task	Description	Skills required
	`-pipeline-bucket-<YYYYMMDDHHMM>` . • Clear the **Block all public access** check box, and select the **I acknowledge...** check box.	
Use IAM to create a role for CloudFormation deployment.	Create a `demobookstore-CloudFormation-role` role and attach the `AdministratorAccess` policy. In the next epic, you can reconfigure this role for minimum permissions.	Developer
Create a new pipeline to automate building and deploying the Books microservice.	Create a pipeline (for example, **demobookstore-BooksService-Pipeline**) with Commit, Build, and Deploy stages, as described in the README file.	Developer
Test your microservice in AWS Cloud9.	Make a change in the **ListBooks** function and see the pipeline working.	Developer
Automate the unit test for the ListBooks Lambda function.	In the AWS Cloud9 IDE, enable the build to run unit tests, and check the test results. For instructions, see the README file.	Developer

(Optional) Implement additional functionality

Task	Description	Skills required
Make your solution secure.	Configure `demobookstore-CloudFormation-role` to have minimum permissions, and check other used roles as well.	Developer
Eliminate dependencies in the CloudFormation templates.	The method for exchanging information between the `DemoBookstoreMainTemplate.yml` template and the `DemoBookstoreBooksServiceTemplate.yml` template is based on outputs and imports. Passing values between these two templates adds dependencies. To eliminate the dependencies, consider using [AWS Systems Manager Parameter Store]().	Developer
Create a Cart microservice.	Use the Books microservice as an example for taking shopping cart-related functions out of the `DemoBookstoreMainTemplate.yml` template and creating a Cart microservice.	Developer

Clean up

Task	Description	Skills required
Delete the S3 buckets.	On the Amazon S3 console, delete the following buckets that are associated with the sample web application: • Two buckets created for the AWS Bookstore Demo App. The buckets names start with the stack name you provided for AWS CloudFormation when you created the frontend; for example, **DemoBookStoreStack**. • One bucket for the build pipeline; for example, **demobookstore-books-service-pipeline-bucket-<YYYYMMDDHHMM>**.	Developer
Delete the stacks.	On the CloudFormation console, delete the stacks associated with the sample web applicaton: • DemoBooksServiceStack • DemoBookStoreStack The removal could take more than 90 minutes. If the removal fails, delete them again, and also delete any manual resources (for	Developer

Task	Description	Skills required
	example, the VPC or network interfaces) based on notificat ions.	
Delete the IAM roles.	On the IAM console, delete the following roles: • `demobookstore-Clou dformation-role` • `demobookstore-Book sService-BuildProj ect-service-role` For step-by-step instructions, see the IAM documentation.	Developer

Related resources

- AWS Bookstore Demo App
- AWS Cloud9 Bootstrapping example
- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Creating a bucket (Amazon S3 documentation)

Additional information

For detailed, step-by-step instructions, see the README file in the AWS DevOps End-to-End Workshop GitHub repository.

About the May 2023 update: This pattern was updated to use newer versions of Node and Python. We updated many of the packages in the source code and removed Glyphicon because it's no longer free. We also removed all dependencies on the AWS Bookstore Demo App repository, so the two repositories can now evolve independently.

Build and push Docker images to Amazon ECR using GitHub Actions and Terraform

Ruchika Modi, Amazon Web Services

Summary

This pattern explains how you can create reusable GitHub workflows to build your Dockerfile and push the resulting image to Amazon Elastic Container Registry (Amazon ECR). The pattern automates the build process of your Dockerfiles by using Terraform and GitHub Actions. This minimizes the possibility of human error and substantially reduces deployment time.

A GitHub push action to the main branch of your GitHub repository initiates the deployment of resources. The workflow creates a unique Amazon ECR repository based on the combination of the GitHub organization and repository name. It then pushes the Dockerfile image to the Amazon ECR repository.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An active GitHub account.
- A [GitHub repository](https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository).
- Terraform version 1 or later [installed and configured](https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli).
- An Amazon Simple Storage Service (Amazon S3) bucket for the [Terraform backend](https://developer.hashicorp.com/terraform/language/settings/backends/configuration).
- An [Amazon DynamoDB](https://aws.amazon.com/dynamodb/) table for Terraform state locking and consistency. The table must have a partition key named `LockID` with a type of `String`. If this isn't configured, state locking will be disabled.
- An AWS Identity and Access Management (IAM) role that has permissions to set up the Amazon S3 backend for Terraform. For configuration instructions, see the [Terraform documentation](https://developer.hashicorp.com/terraform/language/settings/backends/s3#assume-role-configuration).

Limitations

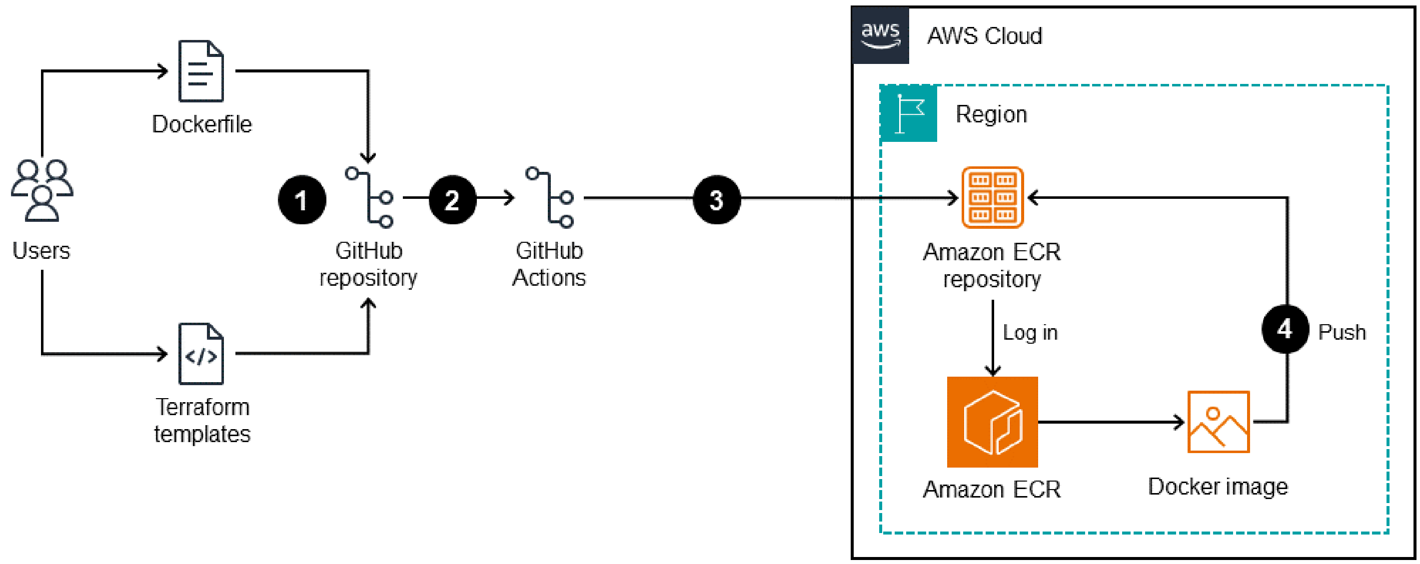
This reusable code has been tested only with GitHub Actions.

Architecture

Target technology stack

- Amazon ECR repository
- GitHub Actions
- Terraform

Target architecture



The diagram illustrates the following:

1. A user adds a Dockerfile and Terraform templates to the GitHub repository.

2. These additions initiate a GitHub Actions workflow.

3. The workflow checks whether an Amazon ECR repository exists. If not, it creates the repository based on the GitHub organization and repository name.

4. The workflow builds the Dockerfile and pushes the image to the Amazon ECR repository.

Tools

Amazon services

- Amazon Elastic Container Registry (Amazon ECR) is a managed container registry service that's secure, scalable, and reliable.

Other tools

- GitHub Actions is integrated into the GitHub platform to help you create, share, and run workflows within your GitHub repositories. You can use GitHub Actions to automate tasks such as building, testing, and deploying your code.
- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises infrastructure.

Code repository

The code for this pattern is available in the GitHub Docker ECR Actions Workflow repository.

- When you create GitHub Actions, Docker workflow files are saved in the `/.github/workflows/` folder of this repository. The workflow for this solution is in the workflow.yaml file.
- The `e2e-test` folder provides a sample Dockerfile for reference and testing.

Best practices

- For best practices for writing Dockerfiles, see the Docker documentation.
- Use a VPC endpoint for Amazon ECR. VPC endpoints are powered by AWS PrivateLink, a technology that enables you to privately access Amazon ECR APIs through private IP addresses. For Amazon ECS tasks that use the Fargate launch type, the VPC endpoint enables the task to pull private images from Amazon ECR without assigning a public IP address to the task.

Epics

Set up the OIDC provider and GitHub repository

Task	Description	Skills required
Configure OpenID Connect.	Create an OpenID Connect (OIDC) provider. You will use the provider in the trust policy for the IAM role used in this action. For instructi ons, see Configuring OpenID	AWS administrator, AWS DevOps, General AWS

Task	Description	Skills required
	Connect in Amazon Web Services in the GitHub documentation.	
Clone the GitHub repository.	Clone the GitHub Docker ECR Actions Workflow repository into your local folder: `$git clone https://github.com/aws-samples/docker-ecr-actions-workflow`	DevOps engineer

Customize the GitHub reusable workflow and deploy the Docker image

Task	Description	Skills required
Customize the event that initiates the Docker workflow.	The workflow for this solution is in workflow.yaml. This script is currently configured to deploy resources when it receives the `workflow_dispatch` event. You can customize this configuration by changing the event to `workflow_call` and calling the workflow from another parent workflow.	DevOps engineer
Customize the workflow.	The workflow.yaml file is configured to create a dynamic, reusable GitHub workflow. You can edit this file to customize the default configuration, or you can	DevOps engineer

Task	Description	Skills required

	pass the input values from the GitHub Actions console if you're using the `workflow_dispatch` event to initiate deployment manually.
	
	• Make sure to specify the correct AWS account ID and target Region.
	• Create an Amazon ECR lifecycle policy (see [sample policy](#)) and update the default path (`e2e-test/policy.json`) accordingly.
	• The workflow file requires two IAM roles as input:
	  • An IAM role that has permissions to set up the Amazon S3 backend for Terraform (see the [Prerequisites](#) section). You can update the default role name `workload-assumable-role` in the `.yaml` file accordingly.
	  • An IAM role that has permissions to access GitHub. This role is also used in the Amazon ECR policy to restrict Amazon ECR operations. For more	

Task	Description	Skills required
	information, see the data.tf file.	
Deploy the Terraform templates.	The workflow automatically deploys the Terraform templates that create the Amazon ECR repository, based on the GitHub event you configured. These templates are available as `.tf` files at the root of the Github repository.	AWS DevOps, DevOps engineer

Troubleshooting

Issue	Solution
Issues or errors when you configure Amazon S3 and DynamoDB as the Terraform remote backend.	Follow the instructions in the Terraform documentation to set up the required permissions on the Amazon S3 and DynamoDB resources for the remote backend configuration.
Unable to run or start the workflow with the `workflow_dispatch` event.	The workflow that's configured to deploy from the `workflow_dispatch` event will work only if the workflow is configured on the main branch as well.

Related resources

- Reusing workflows (GitHub documentation)
- Triggering a workflow (GitHub documentation)

Build and test iOS apps with AWS CodeCommit, AWS CodePipeline, and AWS Device Farm

Abdullahi Olaoye, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern outlines the steps for creating a continuous integration and continuous delivery (CI/CD) pipeline that uses AWS CodePipeline to build and test iOS applications on real devices on AWS. The pattern uses AWS CodeCommit to store the application code, the Jenkins open-source tool to build the iOS application, and AWS Device Farm to test the built application on real devices. These three phases are orchestrated together in a pipeline by using AWS CodePipeline.

This pattern is based on the post Building and testing iOS and iPadOS apps with AWS DevOps and mobile services on the AWS DevOps blog. For detailed instructions, see the blog post.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Apple developer account
- Build server (macOS)
- Xcode version 11.3 (installed and set up on the build server)
- AWS Command Line Interface (AWS CLI) installed and configured on the workstation
- Basic knowledge of Git

Limitations

- The application build server must be running macOS.
- The build server must have a public IP address, so CodePipeline can connect to it remotely to initiate builds.

Architecture

Source technology stack

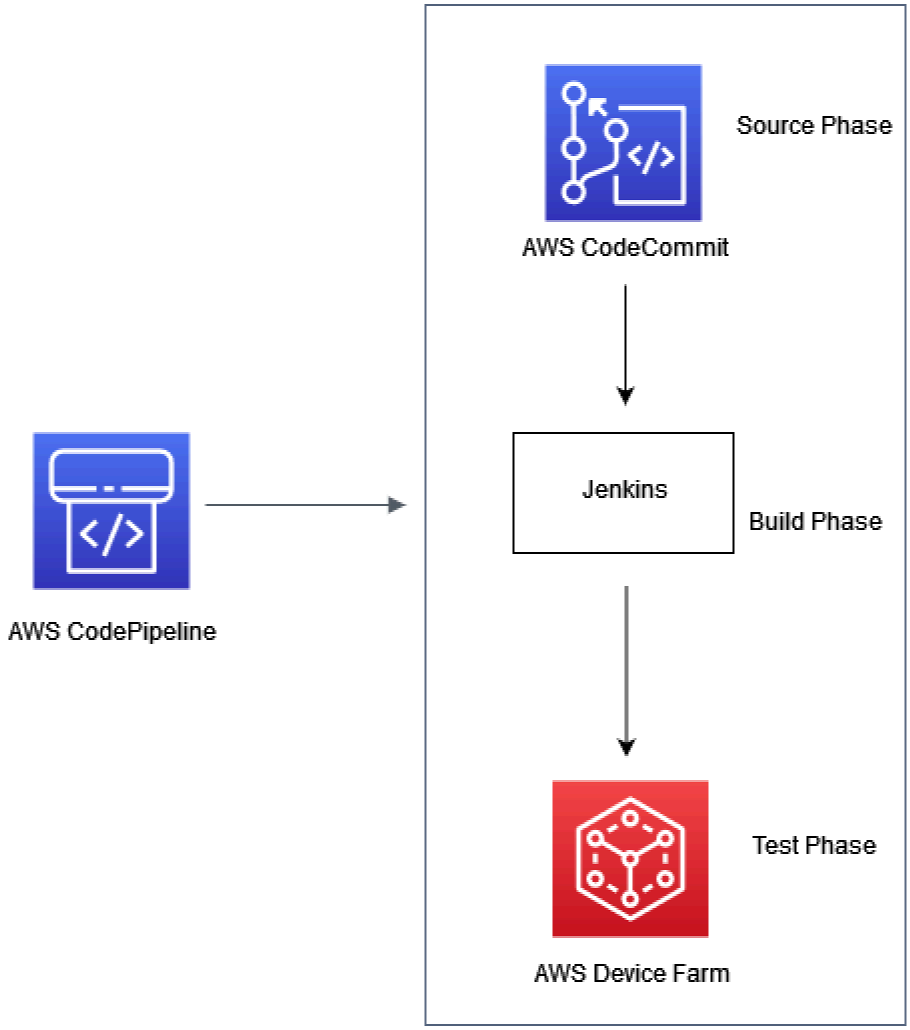
- An on-premises iOS application build process that involves using a simulator or manual test on physical devices

Target technology stack

- An AWS CodeCommit repository for storing application source code
- A Jenkins server for application builds using Xcode
- An AWS Device Farm device pool for testing applications on real devices

Target architecture

When a user commits changes to the source repository, the pipeline (AWS CodePipeline) fetches the code from the source repository, initiates a Jenkins build, and passes the application code to Jenkins. After the build, the pipeline retrieves the build artifact and starts an AWS Device Farm job to test the application against a device pool.



Tools

- [AWS CodePipeline] is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define.
- [AWS CodeCommit] is a fully managed source control service that hosts secure Git-based repositories. It makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem. CodeCommit eliminates the need to operate your own source control system or worry about scaling its infrastructure.
- [AWS Device Farm] is an application testing service that lets you improve the quality of your web and mobile apps by testing them across an extensive range of desktop browsers and real mobile devices, without having to provision and manage any testing infrastructure.

- Jenkins is an open-source automation server that enables developers to build, test, and deploy their software.

Epics

Set up the build environment

Task	Description	Skills required
Install Jenkins on the build server that's running macOS.	Jenkins will be used for building the application, so you must first install it on the build server. To get detailed instructions for this and subsequent tasks, see the AWS blog post Building and testing iOS and iPadOS apps with AWS DevOps and mobile services and other resources in the Related resources section at the end of this pattern.	DevOps
Configure Jenkins.	Follow the on-screen instructions to configure Jenkins.	DevOps
Install the AWS CodePipeline plugin for Jenkins.	This plugin must be installed on the Jenkins server in order for Jenkins to interact with the AWS CodePipeline service.	DevOps
Create a Jenkins freestyle project.	In Jenkins, create a freestyle project. Configure the project to specify triggers and other build configuration options.	DevOps

Configure AWS Device Farm

Task	Description	Skills required
Create a Device Farm project.	Open the AWS Device Farm console. Create a project and a device pool for testing. For instructions, see the blog post.	Developer

Configure the source repository

Task	Description	Skills required
Create a CodeCommit repository.	Create a repository where the source code will be stored.	DevOps
Commit your application code to the repository.	Connect to the CodeCommit repository you created. Push the code from your local machine to the repository.	DevOps

Configure the pipeline

Task	Description	Skills required
Create a pipeline in AWS CodePipeline.	Open the AWS CodePipeline console, and create a pipeline. The pipeline orchestrates all the phases of the CI/CD process. For instructions, see the AWS blog post [Building and testing iOS and iPadOS apps with AWS DevOps and mobile services]().	DevOps

Task	Description	Skills required
Add a test stage to the pipeline.	To add a test stage and integrate it with AWS Device Farm, edit the pipeline.	DevOps
Initiate the pipeline.	To start the pipeline and the CI/CD process, choose **Release change**.	DevOps

View application test results

Task	Description	Skills required
Review test results.	In the AWS Device Farm console, select the project you created, and review the results of the tests. The console will show the details of each test.	Developer

Related resources

Step-by-step instructions for this pattern

- [Building and testing iOS and iPadOS apps with AWS DevOps and mobile services] (AWS DevOps blog post)

Configure AWS Device Farm

- [AWS Device Farm console]

Configure the source repository

- [Create an AWS CodeCommit repository]
- [Connect to an AWS CodeCommit repository]

Configure the pipeline

- AWS CodePipeline console

Additional resources

- AWS CodePipeline documentation
- AWS CodeCommit documentation
- AWS Device Farm documentation
- Jenkins documentation
- Jenkins installation on macOS
- AWS CodePipeline plugin for Jenkins
- Xcode installation
- AWS CLI Installation and configuration
- Git documentation

Check AWS CDK applications or CloudFormation templates for best practices by using cdk-nag rule packs

Arun Donti, Amazon Web Services

Summary

This pattern explains how you can use the cdk-nag utility to check AWS Cloud Development Kit (AWS CDK) applications for best practices by using a combination of rule packs. **cdk-nag** is an open-source project that was inspired by cfn_nag. It implements rules in evaluation packs such as AWS Solutions Library, Health Insurance Portability and Accountability Act (HIPAA), and National Institute of Standards and Technology (NIST) 800-53 by using AWS CDK Aspects. You can check your AWS CDK applications for best practices by using the rules in these packs, detect and remediate code based on best practices, and suppress the rules that you don't want to use in your evaluations.

You can also use **cdk-nag** to check your AWS CloudFormation templates by using the cloudformation-include module.

For information about all available packs, see the Rules section of the cdk-nag repository. Evaluation packs are available for:

- AWS Solutions Library
- HIPAA security
- NIST 800-53 rev 4
- NIST 800-53 rev 5
- Payment Card Industry Data Security Standard (PCI DSS) 3.2.1

Prerequisites and limitations

Prerequisites

- An application that uses the AWS CDK

Tools

- AWS CDK – Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and you can launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

Epics

Integrate cdk-nag with your AWS CDK application

Task	Description	Skills required
Learn about cdk-nag.	Navigate to the cdk-nag GitHub repository and read through the documentation.	App developer
Install the cdk-nag package in your AWS CDK application.	To use **cdk-nag** in your AWS CDK application, you must install it first. **cdk-nag** is available to download from PyPI, npm, NuGet, and Apache Maven. For the latest information about available versions and download locations, see the Readme file in the repository.	App developer
Choose your NagPacks.	**cdk-nag** has different packs of rules called *NagPacks*. Each NagPack contains rules that conform to a specific standard. For example, the AWS Solutions NagPack	App developer

Task	Description	Skills required
	contains general best practices, and the NIST 800-53 rev 5 NagPack can help with compliance. You can apply multiple NagPacks to your application, and you can add and remove packs as necessary. For a list of available packs, see the Readme file in the GitHub repository. For informati on about the individual rules in each pack, see the Rules section of the GitHub repository.	

Task	Description	Skills required

Integrate cdk-nag into your AWS CDK application.	You can integrate **cdk-nag** into your application on an applicationwide level, or integrate it into individual stages or stacks in your application. For example, to integrate the AWS Solutions and HIPAA security NagPacks into an AWS CDK v2 TypeScript application on an applicationwide level, you can use the following code:
	`import { App, Aspects } from 'aws-cdk-lib';`
	`import { CdkTestStack } from '../lib/cdk-test-stack';`
	`import { AwsSolutionsChecks, HIPAASecurityChecks } from 'cdk-nag';`
	
	`const app = new App();`
	`new CdkTestStack(app, 'CdkNagDemo');`
	`// Simple rule informational messages`
	`Aspects.of(app).add(new AwsSolutionsChecks());`
	`// Additional explanations on the purpose of triggered rules`
	`Aspects.of(app).add(new HIPAASecurityChecks({ verbose: true }));`
		App developer

Related resources

- cdk-nag code repository
- cdk-nag in Construct Hub

Configure mutual TLS authentication for applications running on Amazon EKS

Mahendra Siddappa, Amazon Web Services

Summary

Certificate-based mutual Transport Layer Security (TLS) is an optional TLS component that provides two-way peer authentication between servers and clients. With mutual TLS, clients must provide an X.509 certificate during the session negotiation process. The server uses this certificate to identify and authenticate the client.

Mutual TLS is a common requirement for Internet of Things (IoT) applications and can be used for business-to-business applications or standards such as Open Banking.

This pattern describes how to configure mutual TLS for applications running on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster by using an NGINX ingress controller. You can enable built-in mutual TLS features for the NGINX ingress controller by annotating the ingress resource. For more information about mutual TLS annotations on NGINX controllers, see Client certificate authentication in the Kubernetes documentation.

Important

This pattern uses self-signed certificates. We recommend that you use this pattern only with test clusters, and not in production environments. If you want to use this pattern in a production environment, you can use AWS Private Certificate Authority (AWS Private CA) or your existing public key infrastructure (PKI) standard to issue private certificates.

Prerequisites and limitations

Prerequisites

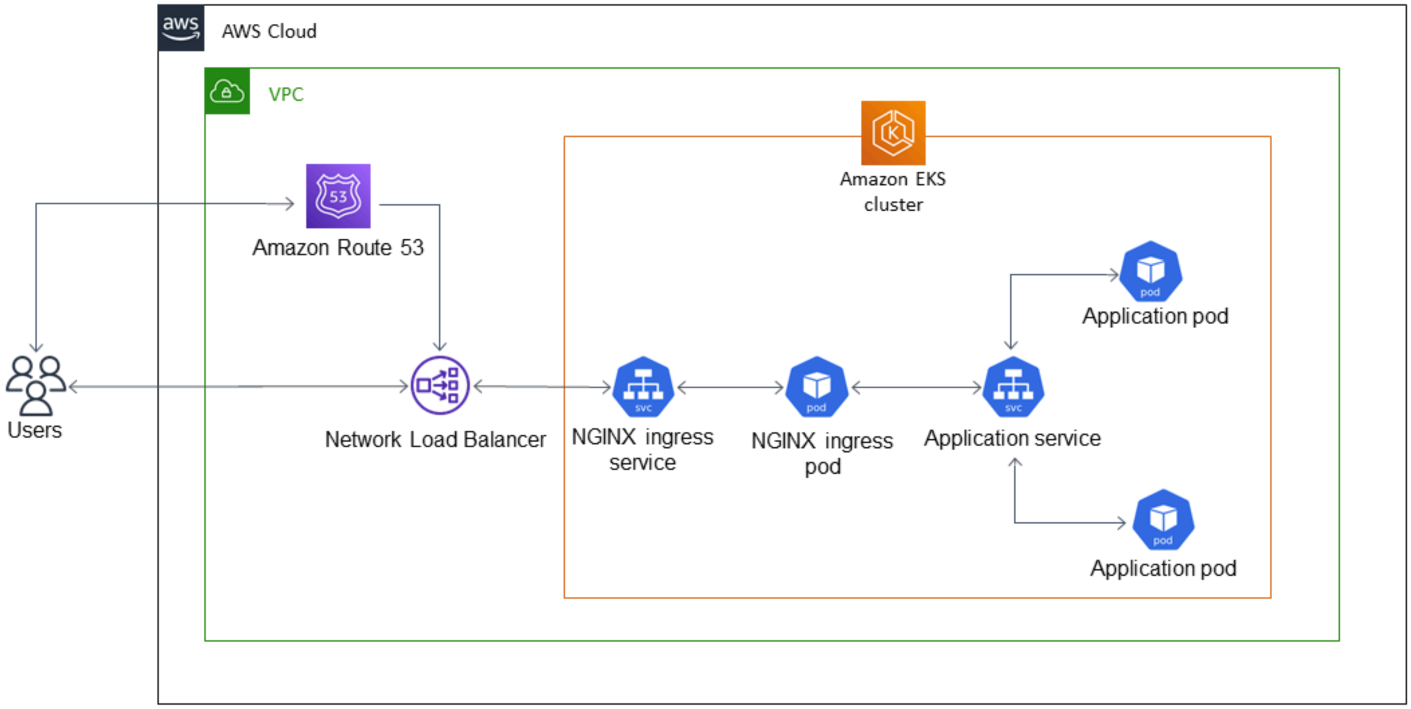
- An active Amazon Web Services (AWS) account.
- An existing Amazon EKS cluster.
- AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows.
- The kubectl command line utility, installed and configured to access the Amazon EKS cluster. For more information about this, see Installing kubectl in the Amazon EKS documentation.

- An existing Domain Name System (DNS) name to test the application.

Limitations

- This pattern uses self-signed certificates. We recommend that you use this pattern only with test clusters, and not in production environments.

Architecture



Technology stack

- Amazon EKS
- Amazon Route 53
- Kubectl

Tools

- [Amazon Elastic Kubernetes Service (Amazon EKS)](#) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- [Amazon Route 53](#) is a highly available and scalable DNS web service.

- Kubectl is a command line utility that you use to interact with an Amazon EKS cluster.

Epics

Generate the self-signed certificates

Task	Description	Skills required
Generate the CA key and certificate.	Generate the certificate authority (CA) key and certificate by running the following command. `openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356 -nodes -subj '/CN=Test Cert Authority'`	DevOps engineer
Generate the server key and certificate, and sign with the CA certificate.	Generate the server key and certificate, and sign with the CA certificate by running the following command. `openssl req -new -newkey rsa:4096 -keyout server.key -out server.csr -nodes -subj '/CN= <your_domain_name> ' && openssl x509 -req -sha256 -days 365 -in server.csr -CA ca.crt -CAkey ca.key -set_serial 01 -out server.crt`	DevOps engineer

Task	Description	Skills required
	Important Make sure sure that you replace `<your_domain_name>` with your existing domain name.	
Generate the client key and certificate, and sign with the CA certificate.	Generate the client key and certificate, and sign with the CA certificate by running the following command. `openssl req -new -newkey rsa:4096 -keyout client.key -out client.csr -nodes -subj '/CN=Test' && openssl x509 -req -sha256 -days 365 -in client.csr -CA ca.crt -CAkey ca.key -set_serial 02 -out client.crt`	DevOps engineer

Deploy the NGINX ingress controller

Task	Description	Skills required
Deploy the NGINX ingress controller in your Amazon EKS cluster.	Deploy the NGINX ingress controller by using the following command. `kubectl apply -f https://raw.githubusercontent.com/ku`	DevOps engineer

Task	Description	Skills required
	`bernetes/ingress-nginx/controller-v1.7.0/deploy/static/provider/aws/deploy.yaml`	
Verify that the NGINX ingress controller service is running.	Verify that the NGINX ingress controller service is running by using the following command. `kubectl get svc -n ingress-nginx` **Important** Make sure that the field of service address contains the Network Load Balancer's domain name.	DevOps engineer

Create a namespace in the Amazon EKS cluster to test mutual TLS

Task	Description	Skills required
Create a namespace in the Amazon EKS cluster.	Create a namespace called `mtls` in your Amazon EKS cluster by running the following command. `kubectl create ns mtls`	DevOps engineer

Task	Description	Skills required
	This deploys the sample application to test mutual TLS.	

Create the deployment and service for the sample application

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create the Kubernetes deployment and service in the mtls namespace.</td><td>Create a file named <code>mtls.yaml</code> . Paste the following code into the file.
<pre><code>kind: Deployment
apiVersion: apps/v1
metadata:
  name: mtls-app
  labels:
    app: mtls
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mtls
  template:
    metadata:
      labels:
        app: mtls
    spec:
      containers:
      - name: mtls-app
        image: hashicorp
/http-echo
        args:
          - "-text=mTLS
 is working"

</code></pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required

	<pre>kind: Service
	apiVersion: v1
	metadata:
	  name: mtls-service
	spec:
	  selector:
	    app: mtls
	  ports:
	    - port: 5678 #
	 Default port for image</pre> Create the Kubernetes deployment and service in the `mtls` namespace by running the following command. <pre>kubectl create -f
	 mtls.yaml -n mtls</pre>	

Verify that the Kubernetes deployment is created.	Run the following command to verify that the deployment is created and has one pod in available status. <pre>kubectl get deploy -n
	 mtls</pre>
		DevOps engineer

Verify that the Kubernetes service is created.	Verify that the Kubernetes service is created by running the following command. <pre>kubectl get service -n
	 mtls</pre>
		DevOps engineer

Create a secret in the mtls namespace

Task	Description	Skills required
Create a secret for the ingress resource.	Run the folllowing command to create a secret for the NGINX ingress controller by using the certificates that you created earlier. `kubectl create secret generic mtls-certs --from-file=tls.crt=server.crt --from-file=tls.key=server.key --from-file=ca.crt=ca.crt -n mtls` Your secret has a server certificate for the client to identify the server and a CA certificate for the server to verify the client certificates.	DevOps engineer

Create the ingress resource in the mtls namespace

Task	Description	Skills required

Create the ingress resource in the mtls namespace.	Create a file named `ingress.yaml` . Paste the following code into the file (replace `<your_domain_name>` with your existing domain name).
	`apiVersion: networking.k8s.io/v1`
	`kind: Ingress`
		DevOps engineer

Task	Description	Skills required

	``
	metadata:
	  annotations:
	    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
	    nginx.ingress.kubernetes.io/auth-tls-secret: mtls/mtls-certs
	  name: mtls-ingress
	spec:
	  ingressClassName: nginx
	  rules:
	  - host: "*.<your_domain_name>"
	    http:
	      paths:
	      - path: /
	        pathType: Prefix
	        backend:
	          service:
	            name: mtls-service
	            port:
	              number: 5678
	  tls:
	  - hosts:
	    - "*.<your_domain_name>"
	    secretName: mtls-certs
	``
	Create the ingress resource in the `mtls` namespace by running the following command.
	``
	kubectl create -f ingress.yaml -n mtls
	``	

Task	Description	Skills required
	This means that the NGINX ingress controller can route traffic to your sample application.	
Verify that the ingress resource is created.	Verify that the ingress resource is created by running the following command. `kubectl get ing -n mtls` **Important** Make sure that the address of the ingress resource shows the load balancer created for the NGINX ingress controller.	DevOps engineer

Configure DNS to point the hostname to the load balancer

Task	Description	Skills required
Create CNAME record that points to the load balancer for the NGINX ingress controller.	Sign in to the AWS Management Console, open the Amazon Route 53 console, and create a Canonical Name (CNAME) record that points `mtls.<your_domain_name>` to the load balancer for the NGINX ingress controller.	DevOps engineer

Task	Description	Skills required
	For more information, see [Creating records by using the Route 53 console] in the Route 53 documentation.	

Test the application

Task	Description	Skills required
Test mutual TLS setup without certificates.	Run the following command. `curl -k https://mtls.<your_domain_name>` You should receive the "400 No required SSL certificate was sent" error response.	DevOps engineer
Test mutual TLS setup with certificates.	Run the following command. `curl -k https://mtls.<your_domain_name> --cert client.crt --key client.key` You should receive the "mTLS is working" response.	DevOps engineer

Related resources

- Creating records by using the Amazon Route 53 console
- Using a Network Load Balancer with the NGINX ingress controller on Amazon EKS
- Client Certificate Authentication

Automate the creation of AppStream 2.0 resources using AWS CloudFormation

Ram Kandaswamy, Amazon Web Services

Summary

This pattern provides code samples and steps to automate the creation of Amazon AppStream 2.0 resources in the Amazon Web Services (AWS) Cloud by using an AWS CloudFormation template. The pattern shows you how to use an AWS CloudFormation stack to automate the creation of your AppStream 2.0 application resources, including an image builder, image, fleet instance, and stack. You can stream your AppStream 2.0 application to end users on an HTML5-compliant browser by using either the desktop or application delivery mode.

Prerequisites and limitations

Prerequisites

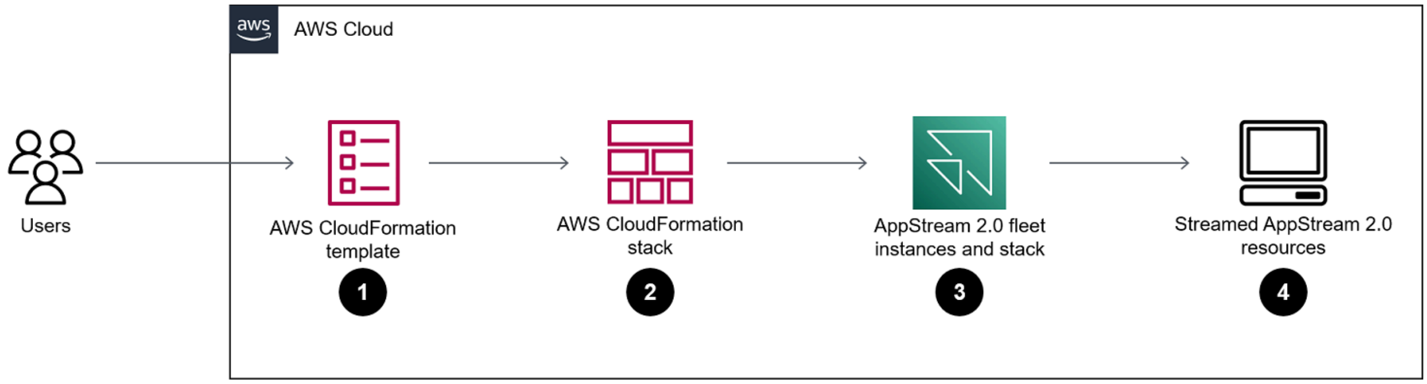
- An active AWS account
- An acceptance of AppStream 2.0 terms and conditions
- Basic knowledge of AppStream resources, such as stacks, fleets, and image builders

Limitations

- You can't modify the AWS Identity and Access Management (IAM) role associated with an AppStream 2.0 instance after that instance is created.
- You can't modify properties (such as the subnet or security group) on the AppStream 2.0 image builder instance after that image builder is created.

Architecture

The following diagram shows you how to automate the creation of AppStream 2.0 resources by using an AWS CloudFormation template.



The diagram shows the following workflow:

1. You create an AWS CloudFormation template based on the YAML code in the *Additional information* section of this pattern.
2. The AWS CloudFormation template creates an AWS CloudFormation test stack.
   a. (Optional) You create an image builder instance by using AppStream 2.0.
   b. (Optional) You create a Windows image by using your custom software.
3. The AWS CloudFormation stack creates an AppStream 2.0 fleet instance and stack.
4. You deploy your AppStream 2.0 resources to end users on an HTML5-compliant browser.

Technology stack

- Amazon AppStream 2.0
- AWS CloudFormation

Tools

- Amazon AppStream 2.0 is a fully managed application streaming service that provides you with instant access to your desktop applications from anywhere. AppStream 2.0 manages the AWS resources required to host and run your applications, scales automatically, and provides access to your users on demand.
- AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

Epics

(Optional) Create an AppStream 2.0 image

Task	Description	Skills required
Install custom software and create an image.	1. Install the AppStream 2.0 application that you plan to deploy to your users. 2. Use the Photon create image agent or a PowerShell script to create a new Windows image for your custom software. **Note** Consider using the Windows AppLocker feature to further lock down the image.	AWS DevOps, Cloud architect

Deploy the AWS CloudFormation template

Task	Description	Skills required
Update the AWS CloudFormation template.	1. Save the code in the *Additional information* section of this pattern as a YAML file. 2. Update the YAML file with the required values for the parameters in your environment.	AWS systems administrator, Cloud administrator, Cloud architect, General AWS, AWS administrator

Task	Description	Skills required
Create an AWS CloudFormation stack using the template.	1. Sign in to the AWS Management Console and open the [AWS CloudFormation console](). 2. On the navigation pane, choose **Stacks**. 3. Choose **Create stack** and then choose **With new resources (standard)**. 4. In the **Prerequisite – Prepare template** section, choose **Template is ready**. 5. In the **Specify template** section, choose **Upload a template file**. 6. Choose **Choose file** and then choose your updated AWS CloudFormation template. 7. Complete the rest of the steps in the wizard to create your stack.	App owner, AWS systems administrator, Windows Engineer

Related resources

References

- Get Started with Amazon AppStream 2.0: Set Up With Sample Applications
- Create an AppStream 2.0 Fleet and Stack

Tutorials and videos

- Amazon AppStream 2.0 User Workflow

- [How to Migrate a Legacy Windows Forms App to Amazon AppStream 2.0]
- [AWS re:Invent 2018: Securely Deliver Desktop Applications with Amazon AppStream 2.0 (BAP201)]

Additional information

The following code is an example of an AWS CloudFormation template that allows you to automatically create AppStream 2.0 resources.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  SubnetIds:
    Type: 'List<AWS::EC2::Subnet::Id>'
  testSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup::Id'
  ImageName:
    Type: String
Resources:

  AppStreamFleet:
    Type: 'AWS::AppStream::Fleet'
    Properties:
      ComputeCapacity:
        DesiredInstances: 5
      InstanceType: stream.standard.medium
      Name: appstream-test-fleet
      DisconnectTimeoutInSeconds: 1200
      FleetType: ON_DEMAND
      IdleDisconnectTimeoutInSeconds: 1200
      ImageName: !Ref ImageName
      MaxUserDurationInSeconds: 345600
      VpcConfig:
        SecurityGroupIds:
          - !Ref testSecurityGroup
        SubnetIds: !Ref SubnetIds
  AppStreamStack:
    Type: 'AWS::AppStream::Stack'
    Properties:
      Description: AppStream stack for test
      DisplayName: AppStream test Stack
      Name: appstream-test-stack
      StorageConnectors:
```

```
        - ConnectorType: HOMEFOLDERS
      UserSettings:
        - Action: CLIPBOARD_COPY_FROM_LOCAL_DEVICE
          Permission: ENABLED
        - Action: CLIPBOARD_COPY_TO_LOCAL_DEVICE
          Permission: ENABLED
        - Action: FILE_DOWNLOAD
          Permission: ENABLED
        - Action: PRINTING_TO_LOCAL_DEVICE
          Permission: ENABLED
  AppStreamFleetAssociation:
    Type: 'AWS::AppStream::StackFleetAssociation'
    Properties:
      FleetName: appstream-test-fleet
      StackName: appstream-test-stack
    DependsOn:
      - AppStreamFleet
      - AppStreamStack
```

Create a custom log parser for Amazon ECS using a Firelens log router

Varun Sharma, Amazon Web Services

Summary

Firelens is a log router for Amazon Elastic Container Service (Amazon ECS) and AWS Fargate. You can use Firelens to route container logs from Amazon ECS to Amazon CloudWatch and other destinations (for example, Splunk or Sumo Logic). Firelens works with Fluentd or Fluent Bit as the logging agent, which means that you can use Amazon ECS task definition parameters to route logs.

By choosing to parse logs at the source level, you can analyze your logging data and perform queries to more efficiently and effectively respond to operational issues. Because different applications have different logging patterns, you need to use a custom parser that structures the logs and makes searching easier at your end destination.

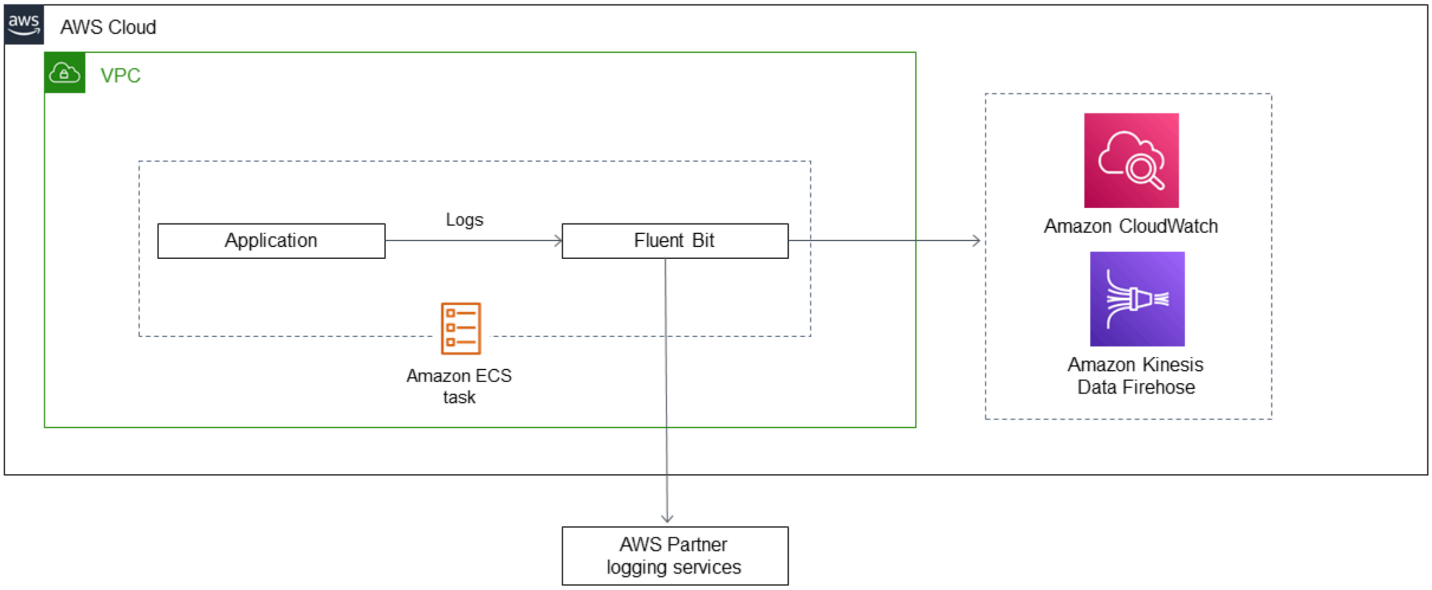
This pattern uses a Firelens log router with a custom parser to push logs to CloudWatch from a sample Spring Boot application running on Amazon ECS. You can then use Amazon CloudWatch Logs Insights to filter the logs based on custom fields that are generated by the custom parser.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account.
- AWS Command Line Interface (AWS CLI), installed and configured on your local machine.
- Docker, installed and configured on your local machine.
- An existing Spring Boot-based containerized application on Amazon Elastic Container Registry (Amazon ECR).

Architecture



Technology stack

- CloudWatch
- Amazon ECR
- Amazon ECS
- Fargate
- Docker
- Fluent Bit

Tools

- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster.
- AWS Identity and Access Management (IAM) – IAM is a web service for securely controlling access to AWS services.
- AWS CLI – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.
- Docker – Docker is an open platform for developing, shipping, and running applications.

Code

The following files are attached to this pattern:

- `customFluentBit.zip` – Contains the files to add the custom parsing and configurations.
- `firelens_policy.json` – Contains the policy document to create an IAM policy.
- `Task.json` – Contains a sample task definition for Amazon ECS.

Epics

Create a custom Fluent Bit image

Task	Description	Skills required
Create an Amazon ECR repository.	Sign in to the AWS Management Console, open the Amazon ECR console, and create a repository called `fluentbit_custom` . For more information about this, see [Creating a repository](#) in the Amazon ECR documentation.	Systems administrator, Developer
Unzip the customFluentBit.zip package.	1. Download the `customFluentBit.zip` package (attached) to your local machine. 2. Unzip to the `customFluentBit` directory by running the following command: `unzip -d customFluentBit.zip`	

Task	Description	Skills required
	3. The directory contains the following files that are required for adding the custom parsing and configurations: • `parsers/springboot_parser.conf` – Contains the parser directive and defines the regular expression (regex) pattern for the custom parser. You can add the regex pattern for your specific parser. • `conf/parse_springboot.conf` – Contains the filter and the service directive. • The Dockerfile	

Task	Description	Skills required
Create the custom Docker image.	1. Change the directory to `customFluentBit .` 2. Open the Amazon ECR console, choose the `fluentbit_custom` repository, and then choose **View push commands**. 3. Upload your project. 4. After the upload is complete, copy the build's URL. This URL is required when you create a container in Amazon ECS For more information about this, see Pushing a Docker image in the Amazon ECR documentation.	Systems administrator, Developer

Set up the Amazon ECS cluster

Task	Description	Skills required
Create an Amazon ECS cluster.	Create an Amazon ECS cluster by following the instructions from the *Networking only template* section of Creating a cluster in the Amazon ECS documentation.	Systems administrator, Developer

Task	Description	Skills required
	Note Make sure that you choose **Create VPC** to create a new virtual private cloud (VPC) for your Amazon ECS cluster.	

Set up the Amazon ECS task

Task	Description	Skills required
Set up the Amazon ECS task execution IAM role.	Create an Amazon ECS task execution IAM role by using the `AmazonECSTaskExecutionRolePolicy` managed policy. For more information about this, see [Amazon ECS task execution IAM role] in the Amazon ECS documentation. **Note** Make sure that you record the IAM role's Amazon Resource Name (ARN).	Systems administrator, Developer
Attach the IAM policy to the Amazon ECS task execution IAM role.	1. Create an IAM policy by using the `firelens_policy.json` (attached	Systems administrator, Developer

Task	Description	Skills required
	) policy document. For more information about this, see Creating policies on the JSON tab in the IAM documentation. 2. Attach this policy to the Amazon ECS task execution IAM role that you created earlier. For more informati on about this, see Adding IAM policies (AWS CLI) in the IAM documentation.	

Task	Description	Skills required
Set up the Amazon ECS task definition.	1. Update the following sections in the `Task.json` sample task definition (attached): • Update the `executionRoleArn` and `taskRoleArn` with the ARN of the task execution IAM role • Update the image in `containerDefinitions` with the custom Fluent Bit Docker image that you created earlier • Update the image in `containerDefinitions` with your application image's name 2. Open the Amazon ECS console, choose **Task Definitions**, choose **Create new task definition**, and then choose **Fargate** on the **Select compatibilities** page. 3. Choose **Configure via Json**, paste the updated `Task.json` file into the text area, and then choose **Save**. 4. Create the task definition.	Systems administrator, Developer

Task	Description	Skills required
	For more information about this, see Creating a task definition in the Amazon ECS documentation.	

Run the Amazon ECS task

Task	Description	Skills required
Run the Amazon ECS task.	On the Amazon ECS console, choose **Clusters**, choose the cluster that you created earlier, and then run the standalone task. For more information about this, see Run a standalone task in the Amazon ECS documentation.	Systems administrator, Developer

Verify the CloudWatch logs

Task	Description	Skills required
Verify the logs.	1. Open the CloudWatch console, choose **Log groups**, and then choose `/aws/ecs/containerinsights/{{cluster_ARN}}/firelens/application` . 2. Verify the logs, particularly the custom fields added by the custom parser.	Systems administrator, Developer

Task	Description	Skills required
	3. Use CloudWatch to filter logs based on the custom fields.	

Related resources

- Docker basics for Amazon ECS
- Amazon ECS on AWS Fargate
- Configuring basic service parameters

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Create a pipeline and AMI using CodePipeline and HashiCorp Packer

Akash Kumar, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern provides code samples and steps to create both a pipeline in the Amazon Web Services (AWS) Cloud by using AWS CodePipeline and an Amazon Machine Image (AMI) by using HashiCorp Packer. The pattern is based on the continuous integration practice, which automates the building and testing of code with a Git-based version control system. In this pattern, you create and clone a code repository by using AWS CodeCommit. Then, create a project and configure your source code by using AWS CodeBuild. Finally, create an AMI that gets committed to your repository.

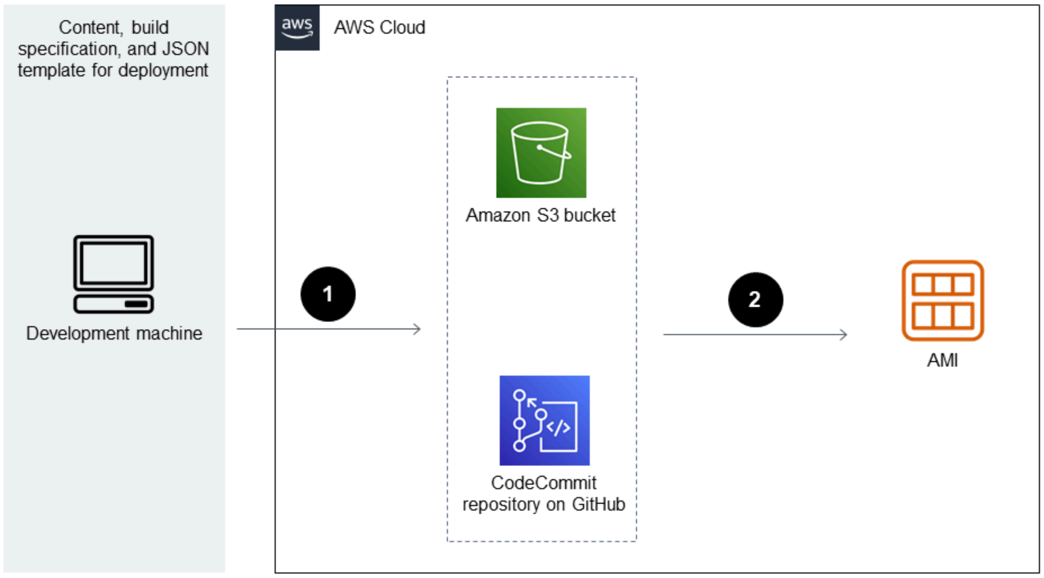
Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Linux AMI for launching Amazon Elastic Compute Cloud (Amazon EC2) instances
- HashiCorp Packer 0.12.3 or later
- Amazon CloudWatch Events (optional)
- Amazon CloudWatch Logs (optional)

Architecture

The following diagram shows an example of application code that automates the creation of an AMI by using this pattern's architecture.



The diagram shows the following workflow:

1. The developer commits code changes to a private CodeCommit Git repository. Then, CodePipeline uses CodeBuild to initiate the build and add new [artifacts](https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome-introducing-artifacts.html) that are ready for deployment to the Amazon Simple Storage Service (Amazon S3) bucket.
2. CodeBuild uses Packer to bundle and package the AMI based on a JSON template. If enabled, CloudWatch Events can automatically start the pipeline when a change occurs in the source code.

Technology stack

- CodeBuild
- CodeCommit
- CodePipeline
- CloudWatch Events (optional)

Tools

- AWS CodeBuild – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- AWS CodeCommit – AWS CodeCommit is a version control service that enables you to privately store and manage Git repositories in the AWS Cloud. CodeCommit eliminates the need for you to manage your own source control system or worry about scaling its infrastructure.
- AWS CodePipeline – AWS CodePipeline is a continuous delivery service that you can use to model, visualize, and automate the steps required to release your software.
- HashiCorp Packer – HashiCorp Packer is an open-source tool for automating the creation of identical machine images from a single source configuration. Packer is lightweight, runs on every major operating system, and creates machine images for multiple platforms in parallel.

Code

This pattern includes the following attachments:

- `buildspec.yml` – This file uses CodeBuild to build and create an artifact for deployment.
- `amazon-linux_packer-template.json` – This file uses Packer to create an Amazon Linux AMI.

Epics

Set up the code repository

Task	Description	Skills required
Create the repository.	Create a CodeCommit repository.	AWS systems administrator
Clone the repository.	Connect to the CodeCommit repository by cloning the repository.	App developer
Push the source code to the remote repository.	1. Create a commit to add the `buildspec.yml` and `amazon-linux_packe`	App developer

Task	Description	Skills required
	`r-template.json` files to your local repository. 2. [Push the commit](#) from your local repository to the remote CodeCommit repository.	

Create a CodeBuild project for the application

Task	Description	Skills required
Create a build project.	1. Sign in to the AWS Management console, open the [AWS CodeBuild console](#), and then choose **Create build project**. 2. For **Project name**, enter the name of your project. 3. For **Source provider**, choose **AWS CodeCommit**. 4. For **Repository**, choose the repository where you want to build the code pipeline. 5. For **Environment image**, choose **Managed image** or **Custom image**. 6. For **Operating system**, choose **Ubuntu**. 7. For **RunTime(s)**, choose **Standard**. 8. For **Image**, choose **aws/codebuild/standard:4.0**.	App developer, AWS systems administrator

Task	Description	Skills required
	9. For **Image version**, choose **Always use the latest image for this runtime version**. 10. For **Environment**, choose **Linux**. 11. Choose the **Privileged** check box. 12. For **Service role**, choose **New service role** or **Existing service role**. 13. For **Build specifications**, choose **Use a buildspec file** or **Insert build commands**. 14. (Optional) For **Type** in the **Artifacts** section, choose **No artifacts**. 15. (Recommended) To upload build output logs to CloudWatch Logs, choose **CloudWatch logs**. 16. (Optional) To upload build output logs to Amazon S3, choose the **S3 logs** check box. 17. Choose **Create build project**.	

Set up the pipeline

Task	Description	Skills required
Pipeline name	1. Sign in to the AWS Management console, open the [AWS CodePipeline console](), and then choose **Create pipeline**. 2. For **Pipeline name**, enter a name for the pipeline. 3. For **Service role**, choose **New service role** or **Existing service role**. 4. For **Role name**, enter a name for your role. 5. In the **Advanced settings** section, for **Artifact store**, choose **Default location** if you want Amazon S3 to create a bucket and store the artifacts in the bucket. To use an existing S3 bucket, choose **Custom location**. Choose **Next**. 6. For **Source provider**, choose **AWS CodeCommit**. 7. For **Repository name**, choose the repository that you cloned earlier. For **Branch name**, choose your source code branch. 8. For **Change detection options**, choose **Amazon CloudWatch Events**	App developer, AWS systems administrator

Task	Description	Skills required
	(recommended) to start the pipeline or **AWS CodePipeline** to periodically check for changes. Choose **Next**. 9. For **Build provider**, choose **AWS CodeBuild**. 10. For **Project Name**, choose the build project that you created in the *Create a CodeBuild project for the application* epic. 11. Choose your build options and then choose **Next**. 12. Choose **Skip deploy stage**. 13. Choose **Create pipeline**.	

Related resources

- Working with repositories in AWS CodeCommit
- Working with build projects
- Working with pipelines in CodePipeline

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Create dynamic CI pipelines for Java and Python projects automatically

Aromal Raj Jayarajan, Vijesh Vijayakumaran Nair, MAHESH RAGHUNANDANAN, and Amarnath Reddy, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. [Learn more]

This pattern shows how to create dynamic continuous integration (CI) pipelines for Java and Python projects automatically by using AWS developer tools.

As technology stacks diversify and development activities increase, it can become difficult to create and maintain CI pipelines that are consistent across an organization. By automating the process in AWS Step Functions, you can make sure that your CI pipelines are consistent in their usage and approach.

To automate the creation of dynamic CI pipelines, this pattern uses the following variable inputs:

- Programming language (Java or Python only)
- Pipeline name
- Required pipeline stages

> **Note**
> Step Functions orchestrates pipeline creation by using multiple AWS services. For more information about the AWS services used in this solution, see the **Tools** section of this pattern.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon S3 bucket in the same AWS Region that this solution is being deployed

- An AWS Identity and Access Management (IAM) principal that has the AWS CloudFormation permissions required to create the resources needed for this solution

Limitations

- This pattern supports Java and Python projects only.
- The IAM roles provisioned in this pattern follow the principle of least privilege. The IAM roles' permissions must be updated based on the specific resources that your CI pipeline needs to create.

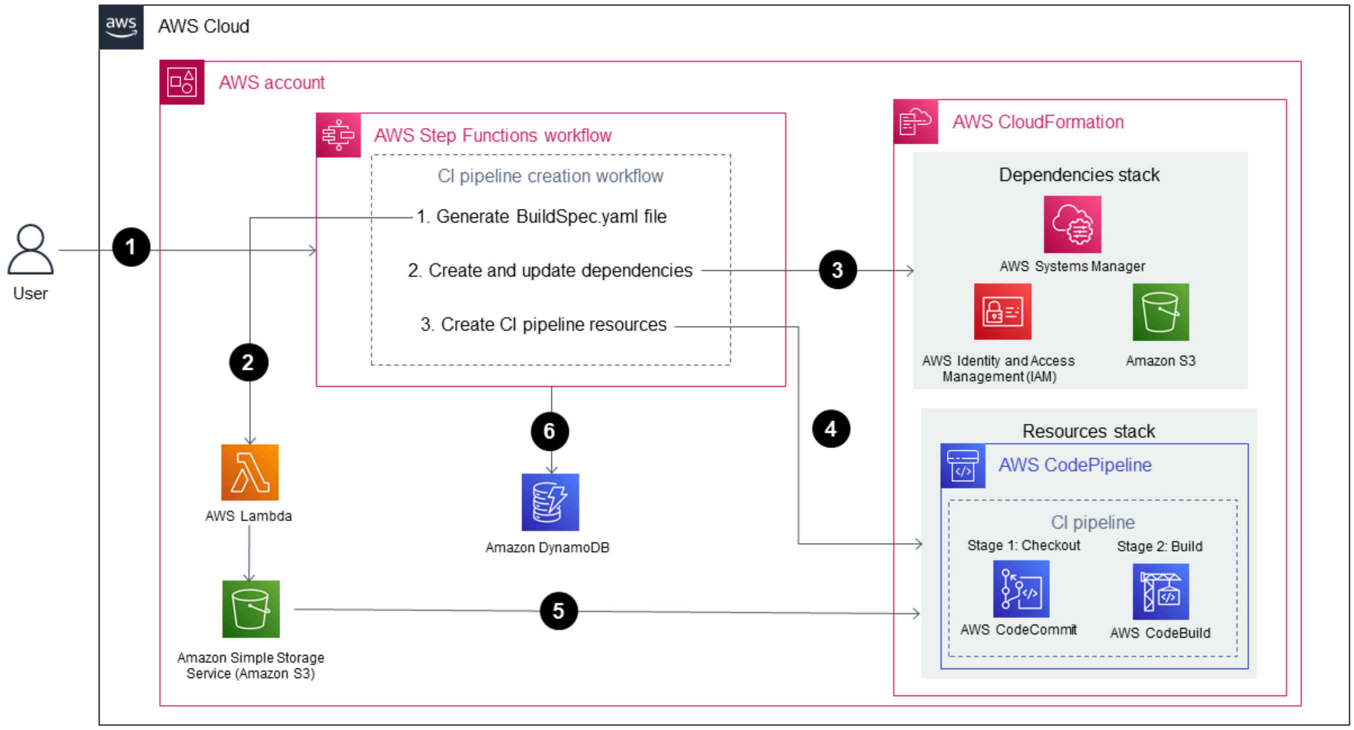
Architecture

Target technology stack

- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- IAM
- Amazon Simple Storage Service (Amazon S3)
- AWS Systems Manager
- AWS Step Functions
- AWS Lambda
- Amazon DynamoDB

Target architecture

The following diagram shows an example workflow for creating dynamic CI pipelines for Java and Python projects automatically by using AWS developer tools.



The diagram shows the following workflow:

1. An AWS user provides the input parameters for CI pipeline creation in JSON format. This input starts a Step Functions workflow (*state machine*) that creates a CI pipeline by using AWS developer tools.
2. A Lambda function reads a folder named **input-reference**, which is stored in an Amazon S3 bucket, and then generates a **buildspec.yml** file. This generated file defines the CI pipeline stages and is stored back in the same Amazon S3 bucket that stores the parameter references.
3. Step Functions checks the CI pipeline creation workflow's dependencies for any changes, and updates the dependencies stack as needed.
4. Step Functions creates the CI pipeline resources in a CloudFormation stack, including a CodeCommit repository, CodeBuild project, and a CodePipeline pipeline.
5. The CloudFormation stack copies the sample source code for the selected technology stack (Java or Python) and the **buildspec.yml** file to the CodeCommit repository.
6. CI pipeline runtime details are stored in a DynamoDB table.

Automation and scale

- This pattern is for use in a single development environment only. Configuration changes are required for use across multiple development environments.
- To add support for more than one CloudFormation stack, you can create additional CloudFormation templates. For more information, see Getting started with AWS CloudFormation in the CloudFormation documentation.

Tools

Tools

- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management.

Code

The code for this pattern is available in the GitHub automated-ci-pipeline-creation repository. The repository contains the CloudFormation templates required to create the target architecture outlined in this pattern.

Best practices

- Don't enter credentials (*secrets*) such as tokens or passwords directly into CloudFormation templates or Step Functions action configurations. If you do, the information will be displayed in the DynamoDB logs. Instead, use AWS Secrets Manager to set up and store secrets. Then, reference the secrets stored in Secrets Manager within the CloudFormation templates and Step Functions action configurations as needed. For more information, see What is AWS Secrets Manager in the Secrets Manager documentation.
- Configure server-side encryption for CodePipeline artifacts stored in Amazon S3. For more information, see Configure server-side encryption for artifacts stored in Amazon S3 for CodePipeline in the CodePipeline documentation.
- Apply least-privilege permissions when configuring IAM roles. For more information, see Apply least-privilege permissions in the IAM documentation.
- Make sure that your Amazon S3 bucket is not publicly accessible. For more information, see Configuring block public access setting for your S3 buckets in the Amazon S3 documentation.
- Make sure that you activate versioning for your Amazon S3 bucket. For more information, see Using versioning in S3 buckets in the Amazon S3 documentation.
- Use IAM Access Analyzer when configuring IAM policies. The tool provides actionable recommendations to help you author secure and functional IAM policies. For more information, see Using AWS Identity and Access Management Access Analyzer in the IAM documentation.
- When possible, define specific access conditions when configuring IAM policies.
- Activate Amazon CloudWatch logging for monitoring and auditing purposes. For more information, see What is Amazon CloudWatch Logs? in the CloudWatch documentation.

Epics

Configure the prerequisites

Task	Description	Skills required
Create an Amazon S3 bucket.	Create an Amazon S3 bucket (or use an existing bucket) to store the required CloudFormation templates, source code, and input files for the solution. For more information, see [Step 1: Create your first S3 bucket](#) in the Amazon S3 documentation. **Note** The Amazon S3 bucket must be in the same AWS Region that you're deploying the solution to.	AWS DevOps
Clone the GitHub repository.	Clone the GitHub [automated-ci-pipeline-creation](#) repository by running the following command in a terminal window: `git clone https://github.com/aws-samples/automated-ci-pipeline-creation.git`	AWS DevOps

Task	Description	Skills required
	For more information, see [Cloning a repository] in the GitHub documentation.	
Upload the Solution Templates folder from the cloned GitHub repository to your Amazon S3 bucket.	Copy the contents from the cloned **Solution-Templates** folder and upload them into the Amazon S3 bucket that you created. For more information, see Uploading objects in the Amazon S3 documentation. **Note** Make sure that you upload the contents of the **Solution-Templates** folder only. You can upload the files at the Amazon S3 bucket's root level only.	AWS DevOps

Deploy the solution

Task	Description	Skills required
Create a CloudFormation stack to deploy the solution by using the template.yml file in the cloned GitHub repository.	1. Sign into the AWS Management Console and then open the AWS CloudFormation console.	AWS administrator, AWS DevOps

Task	Description	Skills required
	2. Choose **Create stack**. A dropdown list appears. 3. In the dropdown list, select **With new resources (standard)**. The **Create stack** page opens. 4. In the **Specify template** section, select the checkbox next to **Upload a template file**. 5. Select **Choose file**. Then, navigate to the cloned GitHub repository's root folder and select the **template.yml** file. Then, choose **Open**. 6. Choose **Next**. The **Specify stack details** page opens. 7. In the **Parameters** section, specify the following parameters: • For **S3TemplateBucketName**, enter the name of the Amazon S3 bucket that you created earlier, which contains the source code and references for this solution. Make sure that the bucket name parameter is in lowercase.	

Task	Description	Skills required
	• For **DynamoDBTable**, enter a name for the DynamoDB table that the CloudFormation stack creates. • For **StateMachineName**, enter a name for the Step Functions state machine that the CloudFormation stack creates. 8. Choose **Next**. The **Configure stack options** page opens. 9. On the **Configure stack options** page, choose **Next**. Don't change any of the default values. The **Review** page opens. 10Review the stack creation settings. Then, choose **Create stack** to launch your stack. **Note** While your stack is being created, it's listed on the **Stacks** page with a status of **CREATE_IN_PROGRESS**. Make sure that you wait	

Task	Description	Skills required
	for the stack's status to change to **CREATE_COMPLETE** before completing the remaining steps in this pattern.	

Test the setup

Task	Description	Skills required

Run the step function that you created.	1. Sign in to the AWS Management Console and then open the [Step Functions console](#).
	2. Open the step function that you created.
	3. Choose **Start execution**. Then, enter your input values for the workflow in JSON format (see the following example inputs).
	4. Choose **Start execution**.
	
	JSON formatting
	
	`{`
	`  "details": {`
	`    "tech_stack": "Name of the Tech Stack (python/java)",`
	`    "project_name": "Name of the Project`
		AWS administrator, AWS DevOps

Task	Description	Skills required

	``
	that you want to create
	with",
	    "pre_build":
	"Choose the step if
	it required in the
	buildspec.yml file
	i.e., yes/no",
	    "build": "Choose the
	step if it required in
	the buildspec.yml file
	i.e., yes/no",
	    "post_build":
	"Choose the step if
	it required in the
	buildspec.yml file
	i.e., yes/no",
	    "reports": "Choose
	the step if it required
	in the buildspec.yml
	file i.e., yes/no",
	  }
	}
	``
	Java JSON input example
	``
	{
	  "details": {
	    "tech_stack":
	"java",
	    "project_name":
	"pipeline-java-pjt",
	    "pre_build": "yes",
	    "build": "yes",
	    "post_build":
	"yes",
	    "reports": "yes"
	  }
	}
	``
	Python JSON input example	

Task	Description	Skills required

	``
	{
	  "details": {
	    "tech_stack": "python",
	    "project_name": "pipeline-python-pjt",
	    "pre_build": "yes",
	    "build": "yes",
	    "post_build": "yes",
	    "reports": "yes"
	  }
	}
	``	

Confirm that the CodeCommit repository for the CI pipeline was created.	1. Sign in to the AWS Management Console and then open the [CodeCommit console](). 2. On the **Repositories** page, verify that the name of the CodeCommit repository that you created appears in the list of repositories. The name of the repository is appended with the following: **pipeline-java-pjt-Repo** 3. Open the CodeCommit repository and validate that the sample source code along with **buildspec.yml** files are pushed to the main branch.	AWS DevOps

Task	Description	Skills required
Check the CodeBuild project resources.	1. Sign in to the AWS Management Console and then open the [CodeBuild console](https://console.aws.amazon.com/codesuite/codebuild/home). 2. On the **Build projects** page, verify that the name of the CodeBuild project that you created appears in the list of projects. The name of the project is appended with the following: **pipeline-java-pjt-Build** 3. Select the name of your CodeBuild project to open the project. Then, review and validate the following configurations: • **Project Configuration** • **Source** • **Environment** • **Buildspec** • **Batch Configuration** • **Artifacts**	AWS DevOps

Task	Description	Skills required
Validate the CodePipeline stages.	1. Sign in to the AWS Management Console and then open the [CodePipeline console](). 2. On the **Pipelines** page, verify that the name of the pipeline that you created appears in the list of pipelines. The name of the pipeline is appended with the following: **pipeline-java-pjt-Pipeline** 3. Select the name of your pipeline to open the pipeline. Then, review and validate each stage of the pipeline, including **Commit** and **Deploy**.	AWS DevOps
Confirm that the CI pipeline ran successfully.	1. In the [CodePipeline console](), on the **Pipelines** page, select the name of your pipeline to view the pipeline's status. 2. Verify that each stage of the pipeline has a **Succeeded** status.	AWS DevOps

Clean up your resources

Task	Description	Skills required
Delete the resources stack in CloudFormation.	Delete the CI pipeline's resources stack in CloudFormation. For more information, see [Deleting a stack on the AWS CloudFormation console](#) in the CloudFormation documentation. **Note** Make sure that you delete the stack named **<project_name>-stack**.	AWS DevOps
Delete the CI pipeline's dependencies in Amazon S3 and CloudFormation.	1. Empty the Amazon S3 bucket named **DeploymentArtifactBucket**. For more information, see [Emptying a bucket](#) in the Amazon S3 documentation. 2. Delete the CI pipeline's dependency stack in CloudFormation. For more information, see [Deleting a stack on the AWS CloudFormation console](#) in the CloudFormation documentation.	AWS DevOps

Task	Description	Skills required
	Note Make sure that you delete the stack named **pipeline-creation-dependencies-stack**.	
Delete the Amazon S3 template bucket.	Delete the Amazon s3 bucket that you created in the **Configure the prerequisites** section of this pattern, which stores the templates for this solution. For more information, see Deleting a bucket in the Amazon S3 documentation.	AWS DevOps

Related resources

- Creating a Step Functions state machine that uses Lambda (AWS Step Functions documentation)
- AWS Step Functions WorkFlow Studio (AWS Step Functions documentation)
- DevOps and AWS
- How does AWS CloudFormation work? (AWS CloudFormation documentation)
- Complete CI/CD with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline (AWS blog post)
- IAM and AWS STS quotas, name requirements, and character limits (IAM documentation)

Deploy CloudWatch Synthetics canaries by using Terraform

Dhrubajyoti Mukherjee and Jean-Francois Landreau, Amazon Web Services

Summary

It's important to validate the health of a system from a customer perspective and confirm that customers are able to connect. This is more difficult when the customers don't constantly call the endpoint. [Amazon CloudWatch Synthetics](#) supports the creation of canaries, which can test both public and private endpoints. By using canaries, you can know the status of a system even if it isn't in use. These canaries are either Node.js Puppeteer scripts or Python Selenium scripts.

This pattern describes how to use HashiCorp Terraform to deploy canaries that test private endpoints. It embeds a Puppeteer script that tests whether a URL returns `200-OK`. The Terraform script can then be integrated with the script that deploys the private endpoint. You can also modify the solution to monitor public endpoints.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account with a virtual private cloud (VPC) and private subnets
- The URL of the endpoint that can be reached from the private subnets
- Terraform installed in the deployment environment

Limitations

The current solution works for the following CloudWatch Synthetics runtime versions:

- syn-nodejs-puppeteer-3.4
- syn-nodejs-puppeteer-3.5
- syn-nodejs-puppeteer-3.6
- syn-nodejs-puppeteer-3.7

As new runtime versions are released, you might need to update the current solution. You will also need to modify the solution to keep up with security updates.

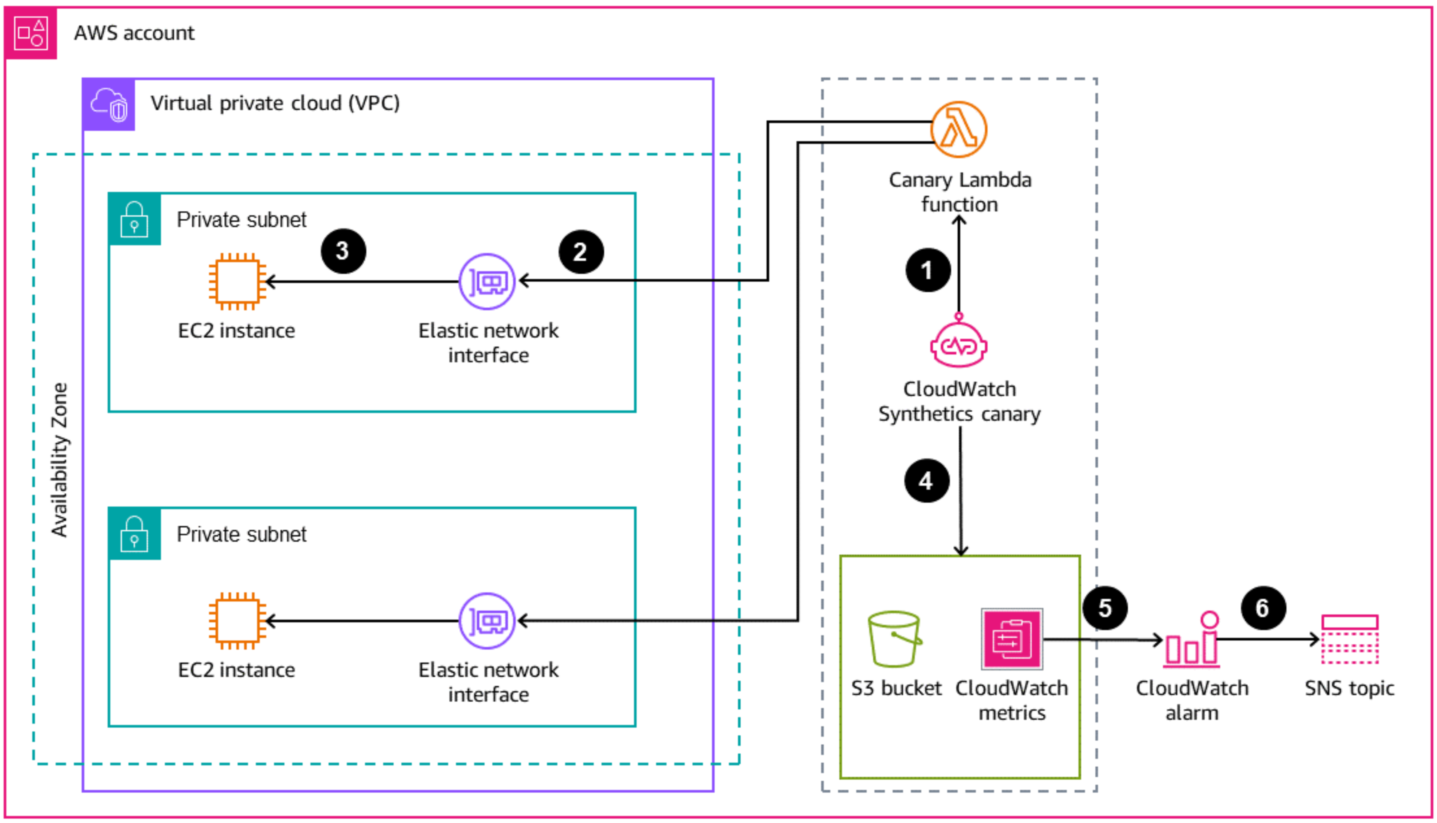
Product versions

- Terraform 1.3.0

Architecture

Amazon CloudWatch Synthetics is based on CloudWatch, Lambda, and Amazon Simple Storage Service (Amazon S3). Amazon CloudWatch offers a wizard to create the canaries and a dashboard that displays the status of the canary runs. The Lambda function runs the script. Amazon S3 stores the logs and screenshots from the canary runs.

This pattern simulates a private endpoint through an Amazon Elastic Compute Cloud (Amazon EC2) instance deployed in the targeted subnets. The Lambda function requires elastic network interfaces in the VPC where the private endpoint is deployed.



The diagram shows the following:

1. The Synthetics canary initiates the canary Lambda function.
2. The canary Lambda function connects to the elastic network interface.
3. The canary Lambda function monitors the status of the endpoint.

4. The Synthetics canary pushes run data to the S3 bucket and CloudWatch metrics.
5. A CloudWatch alarm is initiated based on the metrics.
6. The CloudWatch alarm initiates the Amazon Simple Notification Service (Amazon SNS) topic.

Tools

AWS services

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. This pattern uses VPC endpoints and elastic network interfaces.

Other services

- HashiCorp Terraform is an open-source infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources. This pattern uses Terraform to deploy the infrastructure.
- Puppeteer is a Node.js library. The CloudWatch Synthetics runtime uses the Puppeteer framework.

Code

The solution is available in the GitHub cloud watch-synthetics-canary-terraform repository. For more information, see the *Additional information* section.

Epics

Implement the solution for monitoring a private URL

Task	Description	Skills required
Gather requirements for monitoring the private URL.	Gather the full URL definition: domain, parameters, and headers. To communicate privately to Amazon S3 and Amazon CloudWatch, use VPC endpoints. Note how the VPC and subnets are accessible to the endpoint. Consider the frequency of canary runs.	Cloud architect, Network administrator
Modify the existing solution to monitor the private URL.	Modify the `terraform.tfvars` file: • `name` – The name of your canary. • `runtime_version` – The runtime version of the canary. We recommend using syn-nodejs-puppeteer-3.7. • `take_screenshot` – Whether a screenshot should be taken. • `api_hostname` – The hostname of the endpoint that is monitored. • `api_path` – The path of the endpoint that is monitored.	Cloud architect

Task	Description	Skills required
	`vpc_id` – The VPC ID that is used by the canary Lambda function. `subnet_ids` – The subnet IDs that are used by the canary Lambda function. `frequency` – The run frequency of the canary in minutes. `alert_sns_topic` – The SNS topic to which the CloudWatch alarm notification is sent.	

Deploy and operate the solution.	To deploy the solution, do the following:
	
	1. From the `cloudwatch-synthetics-canary-terraform` directory in your development environment, initialize Terraform.
	`terraform init`
	2. Plan and review the changes.
	`terraform plan`
	3. Deploy the solution.
	`terraform apply`
		Cloud architect, DevOps engineer

Troubleshooting

Issue	Solution
Deletion of the provisioned resources gets stuck.	Manually delete the canary Lambda function, corresponding elastic network interface, and security group, in that order.

Related resources

- Using synthetic monitoring
- Monitor API Gateway endpoints with Amazon CloudWatch Synthetics (blog post)

Additional information

Repository artifacts

The repository artifacts are in the following structure.

```
.
### README.md
### main.tf
### modules
#   ### canary
#   ### canary-infra
### terraform.tfvars
### tf.plan
### variable.tf
```

The `main.tf` file contains the core module, and it deploys two submodules:

- `canary-infra` deploys the infrastructure required for the canaries.
- `canary` deploys the canaries.

The input parameters for the solution are located in the `terraform.tfvars` file. You can use the following code example to create one canary.

```
module "canary" {
```

```
    source = "./modules/canary"
    name   = var.name
    runtime_version = var.runtime_version
    take_screenshot = var.take_screenshot
    api_hostname = var.api_hostname
    api_path = var.api_path
    reports-bucket = module.canary_infra.reports-bucket
    role = module.canary_infra.role
    security_group_id = module.canary_infra.security_group_id
    subnet_ids = var.subnet_ids
    frequency = var.frequency
    alert_sns_topic = var.alert_sns_topic
}
```

The corresponding .var file follows.

```
name   = "my-canary"
runtime_version = "syn-nodejs-puppeteer-3.7"
take_screenshot = false
api_hostname = "mydomain.internal"
api_path = "/path?param=value"
vpc_id = "vpc_id"
subnet_ids = ["subnet_id1"]
frequency = 5
alert_sns_topic = "arn:aws:sns:eu-central-1:111111111111:yyyyy"
```

Cleaning up the solution

If you are testing this in a development environment, you can clean up the solution to avoid accruing costs.

1. On the AWS Management Console, navigate to the Amazon S3 console. Empty the Amazon S3 bucket that the solution created. Make sure to take a backup of the data, if required.
2. In your development environment, from the `cloudwatch-synthetics-canary-terraform` directory, run the `destroy` command.

   ```
   terraform destroy
   ```

Deploy a CI/CD pipeline for Java microservices on Amazon ECS

Vijay Thompson and Sankar Sangubotla, Amazon Web Services

Summary

This pattern guides you through the steps for deploying a continuous integration and continuous delivery (CI/CD) pipeline for Java microservices on an existing Amazon Elastic Container Service (Amazon ECS) cluster by using AWS CodeBuild. When the developer commits the changes, the CI/CD pipeline is initiated and the build process starts in CodeBuild. When the build is complete, the artifact is pushed to Amazon Elastic Container Registry (Amazon ECR) and the latest build from Amazon ECR is picked up and pushed to the Amazon ECS service.

Prerequisites and limitations

Prerequisites

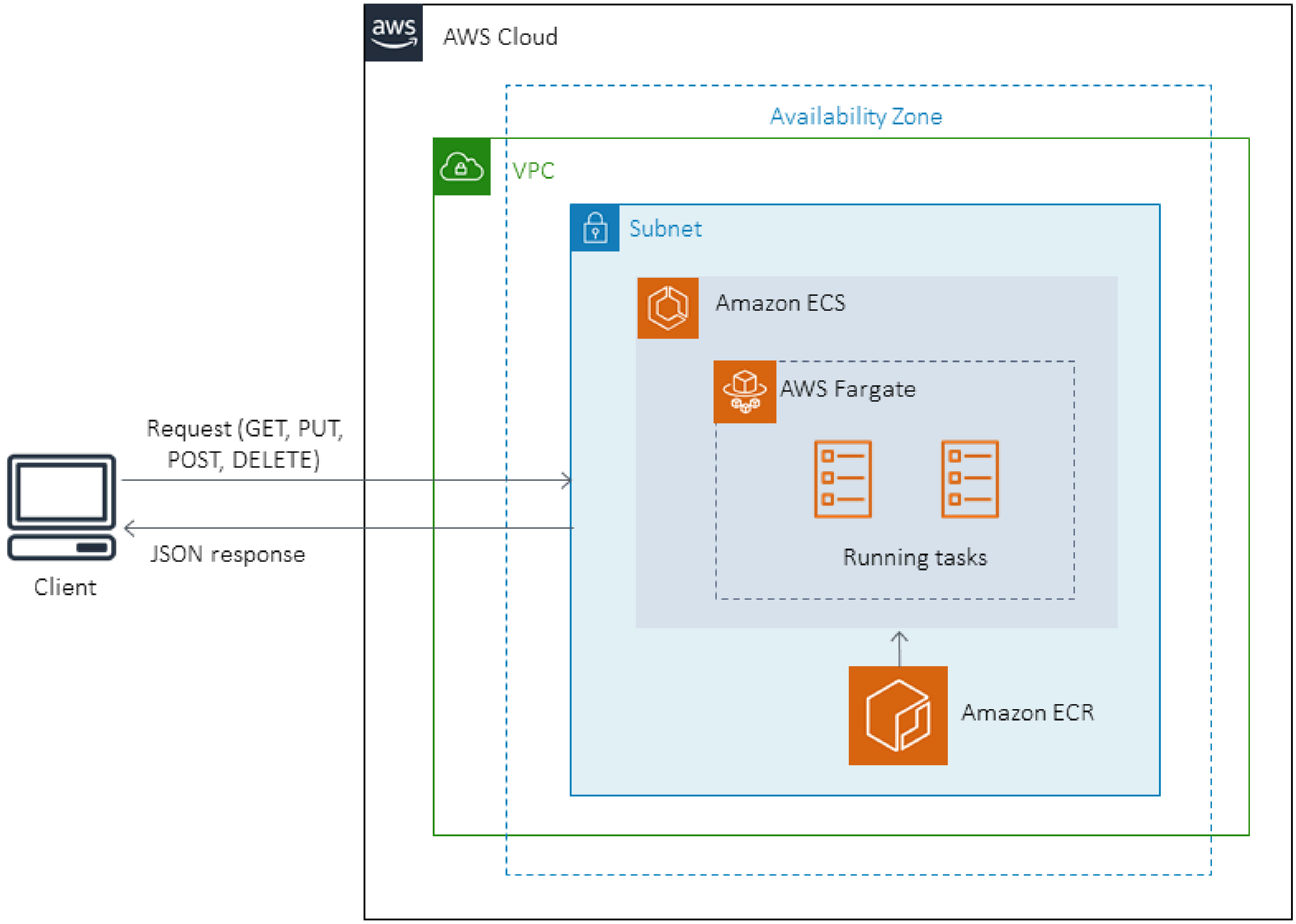
- An existing Java microservices application running on Amazon ECS
- Familiarity with AWS CodeBuild and AWS CodePipeline

Architecture

Source technology stack

- Java microservices running on Amazon ECS
- Code repository in Amazon ECR
- AWS Fargate

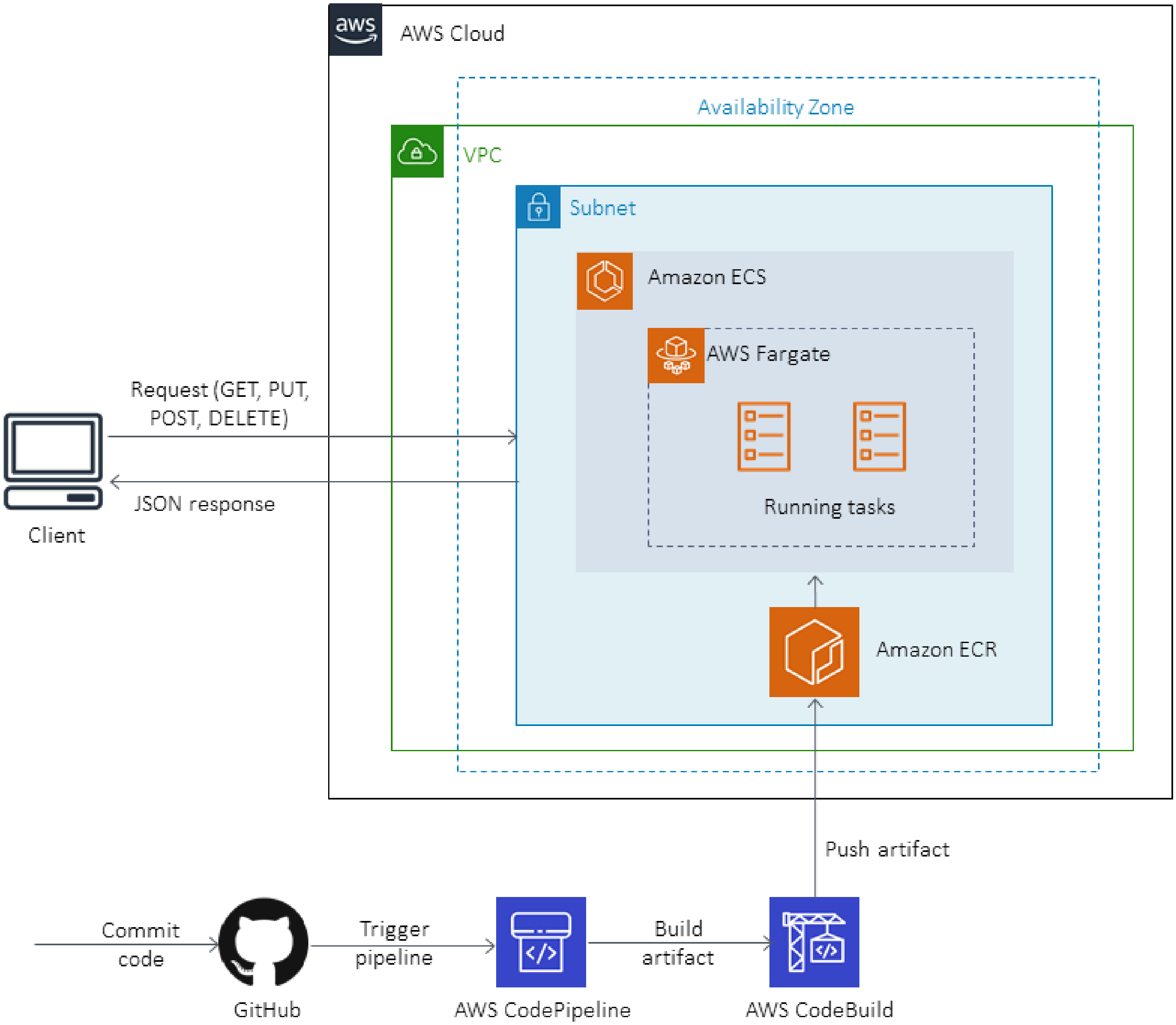
Source architecture



Target technology stack

- Amazon ECR
- Amazon ECS
- AWS Fargate
- AWS CodePipeline
- AWS CodeBuild

Target architecture



Automation and scale

CodeBuild `buildspec.yml` file:

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
```

```
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - REPOSITORY_URI=$AWS_ACCOUNT_ID.dkr.ecr.$AWS_DEFAULT_REGION.amazonaws.com/
$IMAGE_REPO
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=build-$(echo $CODEBUILD_BUILD_ID | awk -F":" '{print $2}')
  build:
    commands:
      - echo Build started on `date`
      - echo building the Jar file
      - mvn clean install
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:$BUILD_TAG .
      - docker tag $REPOSITORY_URI:$BUILD_TAG $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:$BUILD_TAG
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"%s","imageUri":"%s"}]' $DOCKER_CONTAINER_NAME
 $REPOSITORY_URI:$IMAGE_TAG > imagedefinitions.json
      - cat imagedefinitions.json
artifacts:
    files:
     - imagedefinitions.json
     - target/DockerDemo.jar
```

Tools

AWS services

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy. AWS CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left in the queue.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously. You can integrate AWS CodePipeline with third-party services like GitHub, or use an AWS service such as Amazon ECR.
- Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated

with Amazon ECS to simplify your development-to-production workflow. Amazon ECR hosts your images in a highly available and scalable architecture so you can deploy containers for your applications reliably. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each repository.

- Amazon Elastic Container Service (Amazon ECS) highly scalable, high-performance container orchestration service that supports Docker containers and allows you to easily run and scale containerized applications on AWS. Amazon ECS eliminates the need for you to install and operate your own container orchestration software, manage and scale a cluster of virtual machines, or schedule containers on those virtual machines.
- AWS Fargate is a compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing.

Other tools

- Docker is a platform that lets you build, test, and deliver applications in packages called *containers*.
- Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Epics

Set up the build project in AWS CodeBuild

Task	Description	Skills required
Create a CodeBuild build project.	In the AWS CodeBuild console, create a build project, and specify its name.	App developer, AWS systems administrator
Select the source.	This pattern uses Git for the code repository, so choose **GitHub** from the list of	App developer, AWS systems administrator

Task	Description	Skills required
	available options. Choose a public repository or from your GitHub account.	
Select a repository.	Select the repository from which you want to build the code.	App developer, AWS systems administrator
Select the environment.	You can select from a list of managed images or opt for a custom image using Docker. This pattern uses the following managed image: • **Note** Amazon Linux 2 (: Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs.) • Runtime: Standard • Image version 1.0	App developer, AWS systems administrator
Choose a service role.	You can create a service role or select from a list of existing roles.	App developer, AWS systems administrator

Task	Description	Skills required
Add environment variables.	In the **Additional configuration** section, configure the following environment variables: • **AWS_DEFAULT_REGION** for the default AWS Region • **AWS_ACCOUNT_ID** for the user account number • **IMAGE_REPO** for the Amazon ECR private repository • **BUILD_TAG** for the version of the build (**latest** build is the value for this variable) • **DOCKER_CONTAINER_NAME** for the name of the container in the task These variables are placeholders in the `buildspec.yml` file and will be replaced with their respective values.	App developer, AWS systems administrator

Task	Description	Skills required
Create a buildspec file.	You can create a `buildspec.yml` file at the same location as `pom.xml` and add the configuration that is provided in this pattern, or use the online buildspec editor and add the configuration. Configure the environmental variables with the appropriate values by following the steps provided.	App developer, AWS systems administrator
Configure the project for artifacts.	(Optional) Configure the build project for artifacts, if required.	App developer, AWS systems administrator
Configure Amazon CloudWatch Logs.	(Optional) Configure Amazon CloudWatch Logs for the build project, if required. This step is optional but recommended.	App developer, AWS systems administrator
Configure Amazon S3 logs.	(Optional) Configure Amazon Simple Storage Service (Amazon S3) logs for the build project, if you want to store the logs.	App developer, AWS systems administrator

Configure the pipeline in AWS CodePipeline

Task	Description	Skills required
Create a pipeline.	On the [AWS CodePipeline console](), create a pipeline and specify its name. For	App developer, AWS systems administrator

Task	Description	Skills required
	more information about creating a pipeline, see the [AWS CodePipeline documentation]().	
Select a service role.	Create a service role or select from the list of existing service roles. If you are creating a service role, provide a name for the role and select the option for CodePipeline to create the role.	App developer, AWS systems administrator
Choose an artifact store.	In **Advanced settings**, if you want Amazon S3 to create a bucket and store the artifacts in it, use the default location for the artifact store. Or, select a custom location and specify an existing bucket. You can also choose to encrypt the artifact by using an encryption key.	App developer, AWS systems administrator
Specify the source provider.	For **Source provider**, choose **GitHub (Version 2)**.	App developer, AWS systems administrator
Select the repository and branch of the code.	If you are not logged in, provide the connection details to connect to GitHub, and then select the repository name and branch name.	App developer, AWS systems administrator

Task	Description	Skills required
Change detection options.	Choose **Start the pipeline on source code change** and move to the next page.	App developer, AWS systems administrator
Select a build provider.	For **Build provider**, choose **AWS CodeBuild**, and then provide the AWS Region and project name details for the build project. For **Build type**, choose **Single build**.	App developer, AWS systems administrator
Choose a deploy provider.	For **Deploy provider**, choose Amazon ECS. Choose the cluster name, the service name, the image definitions file, if any, and a deploymen t timeout value, if required. Choose **Create pipeline**.	App developer, AWS systems administrator

Related resources

- AWS ECS documentation
- AWS ECR documentation
- AWS CodeBuild documentation
- AWS CodePipeline documentation
- Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source (blog post)

Deploy a ChatOps solution to manage SAST scan results by using Amazon Q Developer in chat applications custom actions and AWS CloudFormation

Anand Bukkapatnam Tirumala, Amazon Web Services

Summary

This pattern presents a comprehensive solution that uses Amazon Q Developer in chat applications to streamline the management of static application security testing (SAST) scan failures reported through SonarQube. This innovative approach integrates custom actions and notifications into a conversational interface, enabling efficient collaboration and decision-making processes within development teams.

In today's fast-paced software development environment, managing SAST scan results efficiently is crucial for maintaining code quality and security. However, many organizations face the following significant challenges:

- Delayed awareness of critical vulnerabilities because of inefficient notification systems
- Slow decision-making processes caused by disconnected approval workflows
- Lack of immediate, actionable responses to SAST scan failures
- Fragmented communication and collaboration around security findings
- Time-consuming and error-prone manual infrastructure setup for security tooling

These issues often lead to increased security risks, delayed releases, and reduced team productivity. To address these challenges effectively requires a solution that can streamline SAST result management, enhance team collaboration, and automate infrastructure provisioning.

Key features of the solution include:

- **Customized notifications –** Real-time alerts and notifications are delivered directly to team chat channels, ensuring prompt awareness and action on SAST scan vulnerabilities or failures.
- **Conversational approvals –** Stakeholders can initiate and complete approval workflows for SAST scan results seamlessly within the chat interface, accelerating decision-making processes.
- **Custom actions –** Teams can define and execute custom actions based on SAST scan outcomes, such as automatically triggering email messages for quality gate failures, enhancing responsiveness to security issues.

- **Centralized collaboration –** All SAST scan-related discussions, decisions, and actions are kept within a unified chat environment, fostering improved collaboration and knowledge-sharing among team members.
- **Infrastructure as code (IaC) –** The entire solution is wrapped with AWS CloudFormation templates, enabling faster and more reliable infrastructure provisioning while reducing manual setup errors.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An AWS Identity and Access Management (IAM) role with permissions to create and manage resources associated with the AWS services listed in Tools.
- A Slack workspace.
- Amazon Q Developer in chat applications added to the required Slack workspace as a plugin. For more information, see Add apps to your Slack workspace in the Slack documentation. Keep a note of the Slack workspace ID as shown on the AWS Management Console after successful registration.
- A configured Amazon Q Developer in chat applications client, with the workspace ID readily available for input in the AWS CloudFormation console. For instructions, see Configure a Slack client in the *Amazon Q Developer in chat applications Administrator Guide*.
- A source email account that is created and verified in Amazon Simple Email Service (Amazon SES) to send out approval email messages. For setup instructions, see Creating and verifying email identities in the *Amazon Simple Email Service Developer Guide*.
- A destination email address for receiving approval notifications. This address can be a shared inbox or a specific team distribution list.
- An operational SonarQube instance that's accessible from your AWS account. For more information, see the SonarQube installation instructions.
- A SonarQube user token with permissions to trigger and create projects through the pipeline.

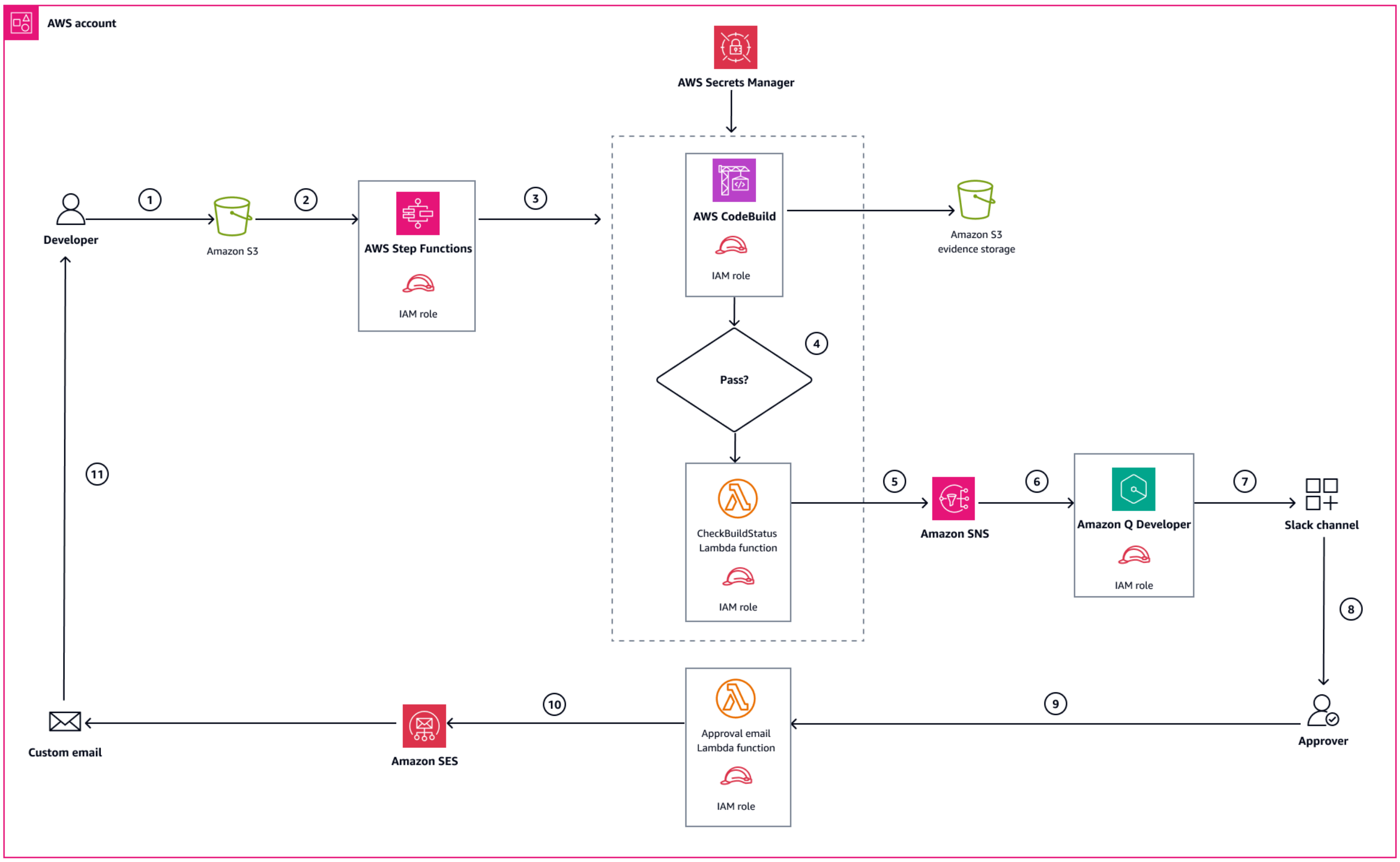
Limitations

- The creation of custom action buttons is a manual process in this solution.

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

The following diagram shows the workflow and architecture components for this pattern.



The diagram shows the automated code quality assurance workflow:

1. Code preparation and upload:
   - The developer compresses the codebase into a .zip file.
   - The developer manually uploads the .zip file to a designated Amazon Simple Storage Service (Amazon S3) bucket.
2. Amazon S3 event trigger and AWS Step Functions orchestration:
   - The Amazon S3 upload event triggers a Step Functions workflow.
   - Step Functions orchestrates a SAST scan using SonarQube.

- The workflow monitors the AWS CodeBuild job status to determine next actions. If CodeBuild succeeds (quality gate pass), the workflow terminates. If CodeBuild fails, an AWS Lambda function is invoked for diagnostics. For more details, see **AWS Step Functions logic** later in this section.

3. AWS CodeBuild execution:
   - The CodeBuild job executes a SonarQube scan on the uploaded codebase.
   - Scan artifacts are stored in a separate Amazon S3 bucket for auditing and analysis.
4. Failure analysis (Lambda function):
   - On CodeBuild failure, the `CheckBuildStatus` Lambda function is triggered.
   - On CodeBuild success, the process is terminated and no further action is needed.
5. Lambda function analyzes failure cause (quality gate failure or other issues)
   - The `CheckBuildStatus` function creates a custom payload with detailed failure information.
   - The `CheckBuildStatus` function publishes the custom payload to an Amazon Simple Notification Service (Amazon SNS) topic.
6. Notification system:
   - Amazon SNS forwards the payload to Amazon Q Developer in chat applications for Slack integration.
7. Slack integration:
   - Amazon Q Developer in chat applications posts a notification in the designated Slack channel.
8. Approval process:
   - Approvers review the failure details in the Slack notification.
   - Approvers can initiate approval using the **Approve** button in Slack.
9. Approval handler:
   - An Approval Lambda function processes the approval action from Slack.
   - The Approval function publishes the custom message to Amazon SES.
10. Message generated:
    - The Approval function generates a custom message for developer notification.
11. Developer notification:
    - Amazon SES sends an email message to the developer with next steps or required actions.

This workflow combines manual code upload with automated quality checks, provides immediate feedback through Slack, and allows for human intervention when necessary, ensuring a robust and flexible code review process.

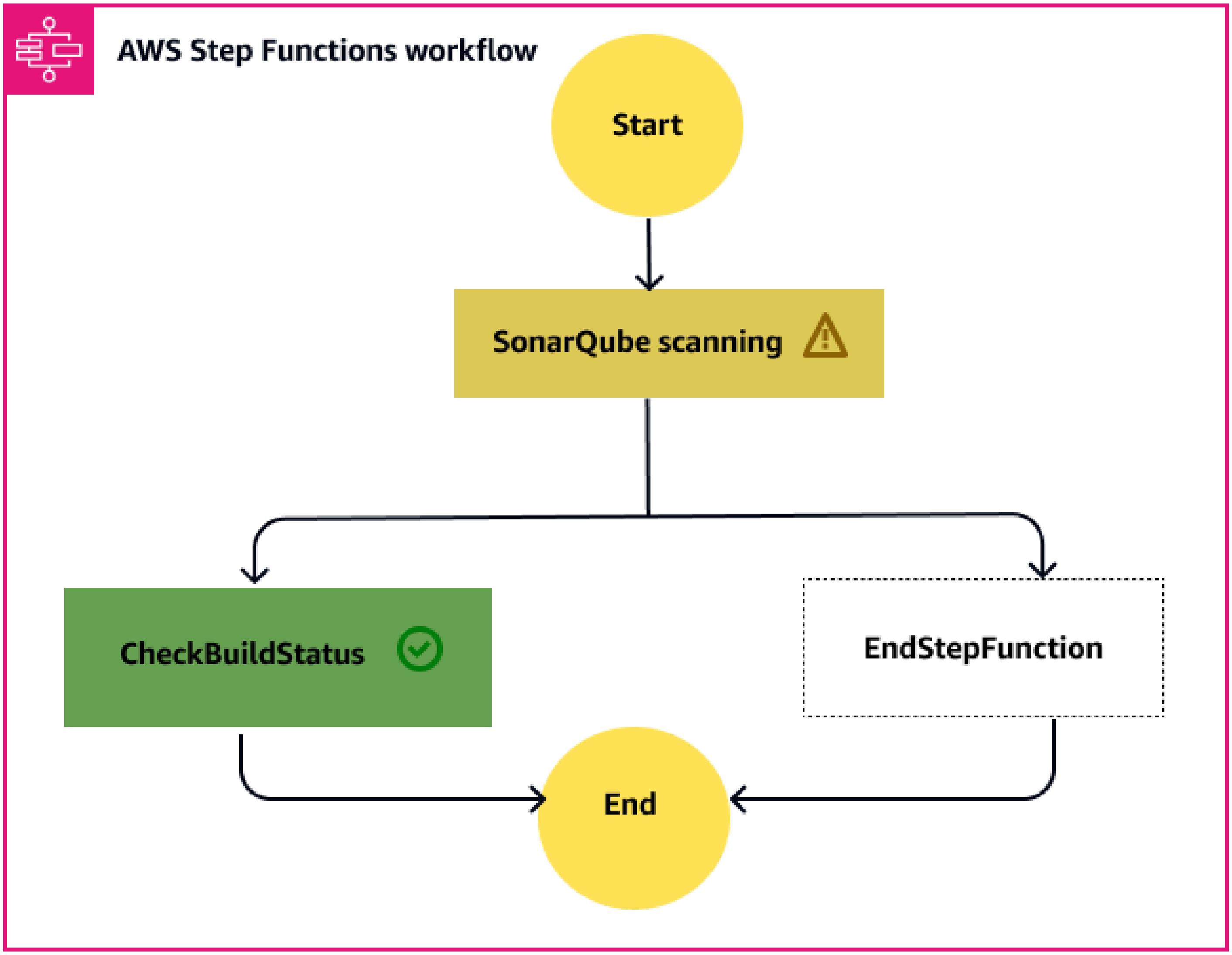
AWS Step Functions logic

As shown in the previous architecture diagram, if the quality gate pass on SonarQube fails, the workflow goes to the `CheckBuildStatus` Lambda function. The `CheckBuildStatus` function triggers a notification on the Slack channel. Each notification includes information with suggested next steps. Following are the types of notifications:

- **Application has failed in code security scan** – The user receives this notification when the uploaded code did not pass the SonarQube security scan. The user can choose **APPROVE** to accept the build. However, the notification advises the user to beware of potential poor code quality and security risks. The notification includes the following details:
  - Next steps: Error: Quality gate status: FAILED – View details at the provided URL.
  - Triage the vulnerabilities as mentioned in the document at the provided URL.
  - CodeBuild details are available at the location at the provided URL.
- **Application scan pipeline has failed because of some other reason** – The user receives this notification when the pipeline failed for some reason other than failing the code security scan. The notification includes the following details:
  - For next steps, go to the link provided for further troubleshooting.

To see screenshots of the notifications as they appear in a Slack channel, go to the assets folder in the GitHub chatops-slack repository.

The following diagram shows an example of Step Functions step status after the quality gate pass fails.



Tools

AWS services

- Amazon Q Developer in chat applications enables you to use Amazon Chime, Microsoft Teams, and Slack chat channels to monitor and respond to operational events in your AWS applications. *End of support notice:* On February 20, 2026, AWS will end support for the Amazon Chime service. After February 20, 2026, you will no longer be able to access the Amazon Chime console or Amazon Chime application resources. For more information, visit the blog post. This does not impact the availability of the Amazon Chime SDK service.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.

- [AWS CodeBuild](#) is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Key Management Service (AWS KMS)](#) helps you create and control cryptographic keys to help protect your data.
- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [AWS Secrets Manager](#) helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- [Amazon Simple Email Service (Amazon SES)](#) helps you send and receive email messages by using your own email addresses and domains.
- [Amazon Simple Notification Service (Amazon SNS)](#) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- [Amazon Simple Storage Service (Amazon S3)](#) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- [AWS Step Functions](#) is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

Other tools

- [Slack](#), a Salesforce offering, is an AI-powered conversational platform that provides chat and video collaboration, automates processes with no code, and supports information sharing.
- [SonarQube](#) is an on-premises analysis tool designed to detect coding issues in over 30 languages, frameworks, and IaC platforms.

Code repository

The code for this pattern is available in the GitHub [chatops-slack](#) repository.

Best practices

- **CloudFormation stack management** – If you encounter any failures during CloudFormation stack execution, we recommend that you delete the failed stack. Then, re-create it with the correct parameter values. This approach supports a clean deployment and helps avoid potential conflicts or partial implementations.
- **Shared inbox email configuration** – When you configure the `SharedInboxEmail` parameter, use a common distribution list that's accessible to all relevant developers. This approach promotes transparency and helps important notifications reach the relevant team members.
- **Production approval workflow** – For production environments, restrict access to the Slack channel that's used for build approvals. Only designated approvers should be members of this channel. This practice maintains a clear chain of responsibility and enhances security by limiting who can approve critical changes.
- **IAM permissions** – Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Perform initial setup

Task	Description	Skills required
Clone the repository.	To clone the chatops-slack repository for this pattern, use the following command. `git clone "git@github.com:aws-samples/chatops-slack.git"`	AWS DevOps, Build lead, DevOps engineer, Cloud administrator
Create the .zip files that contain Lambda code.	Create the .zip files for the AWS Lambda function code for the `CheckBuildStatus` and `ApprovalEmail` functionality. To create `notification.zip` and	AWS DevOps, Build lead, DevOps engineer, Cloud administrator

Task	Description	Skills required

	`approval.zip` , use the following commands.
	`cd chatops-slack/src`
	`chmod -R 775 *`
	`zip -r approval.zip approval`
	`zip -r notification.zip notification`	

Deploy the pre-requisite.yml stack file

Task	Description	Skills required
Execute the `pre-requisite.yml` stack file.	The `pre-requisite.yml` CloudFormation stack file deploys the initial resources that are required before you execute the `app-security.yml` stack file. To execute the `pre-requisite.yml` file, do the following: 1. Sign in to the AWS Management Console, and open the [AWS CloudFormation](https://console.aws.amazon.com/cloudformation/) console. Choose **Create stack**, and then choose **With new**	AWS administrator, AWS DevOps, Build lead, DevOps engineer

Task	Description	Skills required
	resources (standard) from the dropdown list. 2. On the **Create stack** page, choose **Choose an existing template** and **Upload a template file**. Then, choose **Choose file** and select **pre-requisite.yml**. Choose **Next**. 3. On the **Specify stack details** page, enter values for the parameters as described in [Additional information](). Then, choose **Next**. 4. On the **Configure stack options** page, choose the IAM role for creating the resources as described in [Prerequisites](). Then, choose **Next**. 5. On the **Review and create** page, choose **Submit.** 6. On the details page for the stack, choose the **Resources** and **Outputs** tabs. Make a note of the values for the `S3Lambda`, `CKMSKeyArn` , and `CKMSKeyId` parameters that are used in the following steps.	

Task	Description	Skills required
Upload the .zip files to the Amazon S3 bucket.	Upload the `notification.zip` and `approval.zip` files that you created earlier to the Amazon S3 bucket named `S3LambdaBucket`. The `app-security.yml` CloudFormation stack file uses `S3LambdaBucket` to provision the Lambda function.	AWS DevOps, Build lead, DevOps engineer, AWS systems administrator

Execute the app-security.yml stack file

Task	Description	Skills required
Execute the `app-security.yml` stack file.	The `app-security.yml` stack files deploys the remaining infrastructure for the notification and the approval system. To execute the `app-security.yml` file, do the following: 1. Sign in to the AWS Management Console, and open the [AWS CloudFormation](https://console.aws.amazon.com/cloudformation/) console. Choose **Create stack**, and then choose **With new resources (standard)** from the dropdown list. 2. On the **Create stack** page, choose **Choose an existing template** and **Upload**	AWS DevOps, AWS systems administrator, DevOps engineer, Build lead

Task	Description	Skills required
	a template file. Then, choose **Choose file** and select **app-security.yml**. Choose **Next**. 3. On the **Specify stack details** page, enter values for the parameters as described in [Additional information](). Then, choose **Next**. 4. On the **Configure stack options** page, choose the IAM role for creating the resources as described in [Prerequisites](). Then, choose **Next**. 5. On the **Review and create** page, choose **Submit.**	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Test the notification setup.</td><td>To test the notification setup, do the following:

1. Open the Amazon SNS console. In the left navigation pane, choose Topics.
2. Select the topic name that ends with LambdaToAWSSlackChatbot.
3. On the topic’s details page, choose Publish message.
4. On the Publish message to topic page, for Message body to send to the endpoint, enter the following:
<pre>{
    "version": "1.0",
    "source": "custom",
    "content": {
        "description": ":warning: This is a test notification"
    }
}</pre>5. Choose Publish message.

After the test message is delivered successfully, you should see a notification on the Slack channel. For more</td><td>AWS DevOps, AWS systems administrator, DevOps engineer, Build lead</td></tr>
</table>

Task	Description	Skills required
	information, see [Test notifications from AWS services to Slack](#) in the *Amazon Q Developer in chat applications Administrator Guide.*	

Set up approval flow

Task	Description	Skills required
Configure custom Lambda action.	To set up the custom AWS Lambda action, do the following: 1. Choose the vertical ellipsis button on the bottom of the delivery notification in your Slack channel. 2. In **Manage actions**, choose **Create**. 3. Enter a custom action name, for example, **Approve**. This name is a unique identifier for your custom action. 4. Enter a name for your custom action button, for example, **Approve**. This name is shown on a button on your notification. This name should be 20 characters or less and can incorporate emojis.	AWS administrator, AWS DevOps, Build lead, DevOps engineer, Slack Admin

Task	Description	Skills required
	5. For **Custom action type**, select **Lambda action**. 6. Choose **Next**. 7. Select the AWS account and AWS Region where you want to deploy this action. 8. Choose **Load Lambdas**. 9. In **Define Lambda Function**, select a Lambda function that ends with **ApprovalEmailLambda.** Then, choose **Next**. 10. To create the **Approve** button, choose **Save** on the **Display criteria** page.	
Validate approval flow.	To validate that the approval flow works as expected, choose the **Approve** button in Slack. Slackbot should send a notification on the message thread with the confirmation string **Approval Email sent successfully**.	AWS administrator, AWS DevOps, DevOps engineer, Slack Admin

Troubleshooting

Issue	Solution
Slack misconfigurations	For information about troubleshooting issues related to Slack misconfigurations, see Troubleshooting Amazon Q Developer in

Issue	Solution
	the *Amazon Q Developer in chat applications Administrator Guide*.
Scan failed because of some other reason	This error means that the code build task has failed. To troubleshoot the issue, go to the link that's in the message. The failure of the code build task might have the following possible causes: • The application is not packaged properly. The `sonar-scanner` command can't find the `sonar.project.env.properties` file. • The values for the `SonarFileName`, `SonarFileDirectory`, or `SonarToken` parameters are incorrect. Check the values, and then execute the stack files again. • The Sonar host is not reachable. • Other issues that you can use the logs to troubleshoot.

Related resources

AWS documentation

- Configure a Slack client
- Creating a custom action
- Creating an email address identity procedure
- Tutorial: Get started with Slack

Other resources

- Add apps to your Slack workspace (Slack documentation)

- [Generating and using tokens](#) (SonarQube documentation)
- [Introduction to the server installation](#) (SonarQube documentation)

Additional information

This solution emphasizes Amazon Q Developer in chat applications custom actions for release management purposes. However, you can reuse the solution by modifying the Lambda code for your specific use case and build on top of it.

Parameters of CloudFormation stack files

The following table shows the parameters and their descriptions for the CloudFormation stack file `pre-requisite.yml`.

Key	Description
`StackName`	The name of the CloudFormation stack.
`S3LambdaBucket`	The name of the Amazon S3 bucket where you upload the Lambda code. The name must be globally unique.
`SonarToken`	The SonarQube user token as described in [Prerequisites](#).

The following table shows the parameters and their descriptions for the CloudFormation stack file `app-security.yml`.

Key	Description
`CKMSKeyArn`	The AWS KMS key Amazon Resource Name (ARN) that is used in IAM roles and Lambda functions created in this stack.
`CKMSKeyId`	The AWS KMS key ID that is used in the Amazon SNS topic created in this stack.

`EnvironmentType`	The name of the client environment for deployment of the application scan pipeline. Select the environment name from the dropdown list of allowed values.
`S3LambdaBucket`	The name of the Amazon S3 bucket that contains the `approval.zip` and `notification.zip` files.
`SESEmail`	The name of the registered email identity in Amazon SES as described in Prerequisites. This identity is the source email address.
`SharedInboxMail`	The destination email address to which the scan notifications are sent.
`SlackChannelId`	The channel ID of the Slack channel where you want the notifications sent. To find the channel ID, right-click the channel name in **Channel Details** on the Slack app. The channel ID is at the bottom.
`SlackWorkspaceId`	The Slack workspace ID as described in Prerequisites. To find the Slack workspace ID, sign in to the AWS Management Console, open the Amazon Q Developer in chat applications console, and choose **Configured clients**, **Slack**, **WorkspaceID**.
`StackName`	The name of the CloudFormation stack.
`SonarFileDirectory`	The directory that contains the `sonar.project.<env>.properties` file.
`SonarFileName`	The name of the `sonar.project.<env>properties` file.

`SourceCodeZip`	The name of the .zip file that contains the `sonar.project.<env>properties` file and the source code.

Deploy agentic systems on Amazon Bedrock with the CrewAI framework by using Terraform

Vanitha Dontireddy, Amazon Web Services

Summary

This pattern demonstrates how to implement scalable multi-agent AI systems by using the CrewAI framework integrated with Amazon Bedrock and Terraform. The solution enables organizations to create, deploy, and manage sophisticated AI agent workflows through infrastructure as code (IaC). In this pattern, CrewAI multi-agent orchestration capabilities combine with Amazon Bedrock foundation models and Terraform infrastructure automation. As a result, teams can build production-ready AI systems that tackle complex tasks with minimal human oversight. The pattern implements enterprise-grade security, scalability, and operational best practices.

Prerequisites and limitations

Prerequisites

- An active AWS account with appropriate permissions to access Amazon Bedrock foundation models
- Terraform version 1.5 or later installed
- Python version 3.9 or later installed
- CrewAI framework installed

Limitations

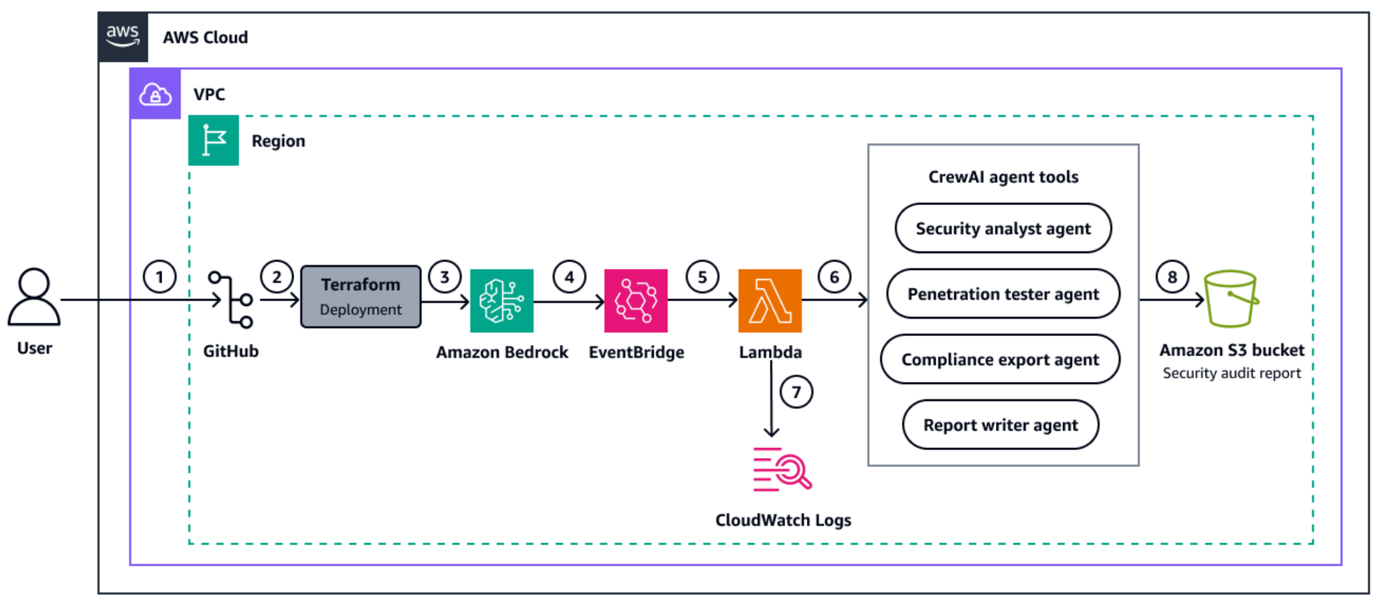
- Agent interactions are limited by model context windows.
- Terraform state management considerations for large-scale deployments apply to this pattern.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

In this pattern, the following interactions occur:

- Amazon Bedrock provides the foundation for agent intelligence through its suite of foundation models (FMs). It enables natural language processing (NLP), reasoning, and decision-making capabilities for the AI agents while maintaining high availability and scalability.
- The CrewAI framework serves as the core orchestration layer for creating and managing AI agents. It handles agent communication protocols, task delegation, and workflow management while integrating with Amazon Bedrock.
- Terraform manages the entire infrastructure stack through code, including compute resources, networking, security groups, and AWS Identity and Access Management (IAM) roles. It ensures consistent, version-controlled deployments across environments. The Terraform deployment creates the following:
  - AWS Lambda function to run the CrewAI application
  - Amazon Simple Storage Service (Amazon S3) buckets for code and reports
  - IAM roles with appropriate permissions
  - Amazon CloudWatch logging
  - Scheduled execution by Amazon EventBridge

The following diagram illustrates the architecture for deploying CrewAI multi-agent systems by using Amazon Bedrock and Terraform.



The diagram shows the following workflow:

1. The user clones the repository.

2. The user runs the command `terraform apply` to deploy the AWS resources.
3. Amazon Bedrock model configuration includes specifying the foundation model (FM) to use for configuring the CrewAI agents.
4. An EventBridge rule is established to trigger the Lambda function according to the defined schedule.
5. When triggered (either by schedule or manually), the Lambda function initializes and assumes the IAM role with permissions to access AWS services and Amazon Bedrock.
6. The CrewAI framework loads agent configurations from YAML files and creates specialized AI agents (the *AWS infrastructure security audit* crew). The Lambda function sequentially executes these agents to scan AWS resources, analyze security vulnerabilities, and generate comprehensive audit reports.
7. CloudWatch Logs captures detailed execution information from the Lambda function with a 365-day retention period and AWS Key Management Service (AWS KMS) encryption for compliance requirements. The logs provide visibility into agent activities, error tracking, and performance metrics, enabling effective monitoring and troubleshooting of the security audit process.
8. The security audit report is automatically generated and stored in the designated Amazon S3 bucket. The automated setup helps maintain consistent security monitoring with minimal operational overhead.

After the initial deployment, the workflow provides ongoing security auditing and reporting for your AWS infrastructure without manual intervention.

Overview of AI agents

This pattern creates multiple AI agents, each with unique roles, goals, and tools:

- The **security analyst agent** collects and analyzes AWS resource information.
- The **penetration tester agent** identifies vulnerabilities in AWS resources.
- The **compliance expert agent** checks configurations against compliance standards.
- The **report writer agent** compiles findings into comprehensive reports.

These agents collaborate on a series of tasks, leveraging their collective skills to perform security audits and generate comprehensive reports. (The `config/agents.yaml` file outlines the capabilities and configurations of each agent in this crew.)

Security analysis processing consists of the following actions:

1. The security analyst agent examines the collected data about AWS resources such as the following:
   - Amazon Elastic Compute Cloud (Amazon EC2) instances and security groups
   - Amazon S3 buckets and configurations
   - IAM roles, policies, and permissions
   - Virtual private cloud (VPC) configurations and network settings
   - Amazon RDS databases and security settings
   - Lambda functions and configurations
   - Other AWS services within audit scope
2. The penetration tester agent identifies potential vulnerabilities.
3. The agents collaborate through the CrewAI framework to share findings.

Report generation consists of the following actions:

1. The report writer agent compiles findings from all other agents.
2. Security issues are organized by service, severity, and compliance impact.
3. Remediation recommendations are generated for each identified issue.
4. A comprehensive security audit report is created in markdown format and uploaded to the designated Amazon S3 bucket. Historical reports are preserved for compliance tracking and security posture improvement.

Logging and monitoring activities include:

- CloudWatch logs capture execution details and any errors.
- Lambda execution metrics are recorded for monitoring.

> **Note**
>
> The code for `aws-security-auditor-crew` is sourced from the GitHub [3P-Agentic_frameworks](https://github.com/aws-samples/3P-Agentic_frameworks) repository, available in the AWS Samples collection.

Availability and scale

You can expand the available agents to more than the four core agents. To scale with additional specialized agents, consider the following new agent types:

- A *threat intelligence specialist* agent can do the following:
  - Monitors external threat feeds and correlates with internal findings
  - Provides context on emerging threats relevant to your infrastructure
  - Prioritizes vulnerabilities based on active exploitation in the wild
- *Compliance framework* agents can focus on specific regulatory areas such as the following:
  - Payment Card Industry Data Security Standard (PCI DSS) compliance agent
  - Health Insurance Portability and Accountability Act of 1996 (HIPAA) compliance agent
  - System and Organization Controls 2 (SOC 2) compliance agent
  - General Data Protection Regulation (GDPR) compliance agent

By thoughtfully expanding the available agents, this solution can provide deeper, more specialized security insights while maintaining scalability across large AWS environments. For more information about an implementation approach, tool development, and scaling considerations, see Additional information.

Tools

AWS services

- Amazon Bedrock is a fully managed AI service that makes high-performing foundation models (FMs) available for use through a unified API.
- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts. In this pattern, it's used for scheduling and orchestrating agent workflows.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [AWS SDK for Python (Boto3)](#) is a software development kit that helps you integrate your Python application, library, or script with AWS services.
- [Amazon Simple Storage Service (Amazon S3)](#) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data. In this pattern, it provides object storage for agent artifacts and state management.

Other tools

- CrewAI is an open source Python-based framework for building multi-agent AI systems.
- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub deploy-crewai-agents-terraform repository.

Best practices

- Implement proper state management for Terraform by using an Amazon S3 backend with Amazon DynamoDB locking. For more information, see Backend best practices in *Best practices for using the Terraform AWS Provider*.
- Use workspaces to separate development, staging, and production environments.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.
- Enable detailed logging and monitoring through CloudWatch Logs.
- Implement retry mechanisms and error handling for agent operations.

Epics

Deploy CrewAI framework

Task	Description	Skills required

Clone the repository.	To clone this pattern’s repository on your local machine, run the following command:
	`git clone "git@github.com:aws-samples/deploy-crewai-agents-terraform.git"`
	`cd deploy-crewai-agents-terraform`
		DevOps engineer

Edit the environment variables.	To edit the environment variables, do the following: 1. Create a `terraform.tfvars` file from the example `terraform.tfvars.example` in the terraform directory. 2. Edit the environment variables to use your own information.	DevOps engineer

Create the infrastructure.	To create the infrastructure, run the following commands:
	`cd terraform`
	`terraform init`
		DevOps engineer

Task	Description	Skills required

	`terraform plan`
	
	Review the execution plan carefully. If the planned changes are acceptable, then run the following command:
	
	`terraform apply --auto-approve`	

Access CrewAI agents

Task	Description	Skills required
Access the agents.	The agents in the AWS Infrastructure Security Audit and Reporting crew are deployed as a Lambda function. To access the agents, use the following steps: 1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/. 2. On the **Functions** page, find and then select the function named `{project_name}` \function (as	DevOps engineer

Task	Description	Skills required
	defined in your Terraform variables). 3. From the function's page, you can take the following actions: • View configuration details. • Monitor execution metrics. • View CloudWatch logs. • Test the function manually.	

Task	Description	Skills required
(Optional) Configure manual execution of the agents.	The agents are configured to run automatically on a daily schedule (midnight UTC). However, you can trigger them manually by using the following steps: 1. In the Lambda console, select the function named `{project_name}` . 2. On the function's page, choose the **Test** tab. 3. Create a new test event with an empty JSON object `{}`. 4. To execute the event, choose **Test**. For more details, see [Testing Lambda functions in the console](#) in the Lambda documentation.	DevOps engineer

Task	Description	Skills required
Access agent logs for debugging.	The CrewAI agents are running in a Lambda environment with the necessary permissions to perform security audits and store reports in Amazon S3. The output is a markdown report that provides a comprehensive security analysis of your AWS infrastructure. To assist with detailed debugging of agent behavior, do the following: 1. In the AWS Management Console, navigate to CloudWatch Logs. 2. Find the log group for your Lambda function. 3. Look for log entries with agent names (for example, *Infrastructure mapping specialist* and *Exploratory security analyst*). 4. Review the logs for insights into the actions of each agent.	DevOps engineer

Task	Description	Skills required
View results of agent execution.	To view the results of an agent execution, do the following: 1. In the AWS Managemen t Console, navigate to Amazon S3. 2. Open the Amazon S3 bucket named `{project_name}-reports-{random_suffix}` (as defined in your Terraform variables). Reports are stored with timestamp-based filenames as follows: `security-audit-report-YYYY-MM-DD-HH-MM-SS.md`)	DevOps engineer

Task	Description	Skills required
Monitor agent execution.	To monitor the agents' execution through CloudWatch logs, do the following: 1. In the AWS Management Console, navigate to CloudWatch. 2. Go to **Log groups**. 3. Select the log group named `/aws/lambda/{project_name}-function` . 4. In **Log streams**, choose the most recent log stream to see detailed execution information.	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Customize agent behavior.</td><td>To modify the agents or their tasks, do the following:

1. Update the configuration files in your local repository:

• The following file defines each agent's role, capabilities, and settings: <code>src/aws_infrastructure_security_audit_and_reporting/config/agents.yaml</code>

• The following dile defines the tasks and workflows for the agents:

<code>src/aws_infrastructure_security_audit_and_reporting/config/tasks.yaml</code>

2. To repackage and update the Lambda function, use the following commands:

<code>cd terraform</code>

<code>terraform apply</code></td><td>DevOps engineer</td></tr>
</table>

Clean up resources

Task	Description	Skills required

Delete the created resources.	To delete all infrastructure created by this pattern, run the following command:
	`terraform plan -destroy`
	Warning
	The following command will permanently delete all resources created by this pattern. The command will prompt for confirmation before removing any resources.
	Review the destruction plan carefully. If the planned deletions are acceptable, then run the following command:
	`terraform destroy`
		DevOps engineer

Troubleshooting

Issue	Solution
Agent behavior	For information about this issue, see Test and troubleshoot agent behavior in the Amazon Bedrock documentation.
Lambda network issues	For information about these issues, see Troubleshoot networking issues in Lambda in the Lambda documentation.
IAM permissions	For information about these issues, see Troubleshoot IAM in the IAM documentation.

Related resources

AWS Blogs

- Build agentic systems with CrewAI and Amazon Bedrock

AWS documentation

- Amazon Bedrock documentation
- How Amazon Bedrock Agents works
- AWS Well-Architected Framework

Other resources

- CrewAI documentation
- CrewAI templates
- Terraform AWS Provider documentation

Additional information

This section contains information about an implementation approach, tool development, and scaling considerations related to the earlier discussion in Automation and scale.

Implementation approach

Consider the following approach to adding agents:

1. Agent configuration:
   - Add new agent definitions to the `config/agents.yaml` file.
   - Define specialized backstories, goals, and tools for each agent.
   - Configure memory and analysis capabilities based on agent specialty.
2. Task orchestration:
   - Update the `config/tasks.yaml` file to include new agent-specific tasks.
   - Create dependencies between tasks to help ensure proper information flow.
   - Implement parallel task execution where appropriate.

Technical implementation

Following is an addition to the `agents.yaml` file for a proposed Threat Intelligence Specialist agent:

```
Example new agent configuration in agents.yaml
threat_intelligence_agent:
 name: "Threat Intelligence Specialist"
 role: "Cybersecurity Threat Intelligence Analyst"
 goal: "Correlate AWS security findings with external threat intelligence"
 backstory: "Expert in threat intelligence with experience in identifying emerging
 threats and attack patterns relevant to cloud infrastructure."
verbose: true
allow_delegation: true
tools:
- "ThreatIntelligenceTool"
- "AWSResourceAnalyzer"
```

Tool development

With the CrewAI framework, you can take the following actions to enhance your security audit crew's effectiveness:

- Create custom tools for new agents.
- Integrate with external APIs for threat intelligence.
- Develop specialized analyzers for different AWS services.

Scaling considerations

When expanding your AWS Infrastructure Security Audit and Reporting system to handle larger environments or more comprehensive audits, address the following scaling factors:

- **Computational resources**
  - Increase Lambda memory allocation to handle additional agents.
  - Consider splitting agent workloads across multiple Lambda functions.
- **Cost management**
  - Monitor Amazon Bedrock API usage as agent count increases.
  - Implement selective agent activation based on audit scope.
- **Collaboration efficiency**
  - Optimize information sharing between agents.
  - Implement hierarchical agent structures for complex environments.
- **Knowledge base enhancement**
  - Provide agents with specialized knowledge bases for their domains.
  - Regularly update agent knowledge with new security best practices.

Deploy an AWS Glue job with an AWS CodePipeline CI/CD pipeline

Bruno Klein and Luis Henrique Massao Yamada, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern demonstrates how you can integrate AWS CodeCommit and AWS CodePipeline with AWS Glue, and use AWS Lambda to launch jobs as soon as a developer pushes their changes to a remote AWS CodeCommit repository.

When a developer submits a change to an extract, transform, and load (ETL) repository and pushes the changes to AWS CodeCommit, a new pipeline is invoked. The pipeline initiates a Lambda function that launches an AWS Glue job with these changes. The AWS Glue job performs the ETL task.

This solution is helpful in the situation where businesses, developers, and data engineers want to launch jobs as soon as changes are committed and pushed to the target repositories. It helps achieve a higher level of automation and reproducibility, therefore avoiding errors during the job launch and lifecycle.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Git installed on the local machine
- Amazon Cloud Development Kit (Amazon CDK) installed on the local machine
- Python installed on the local machine
- The code in the *Attachments* section

Limitations

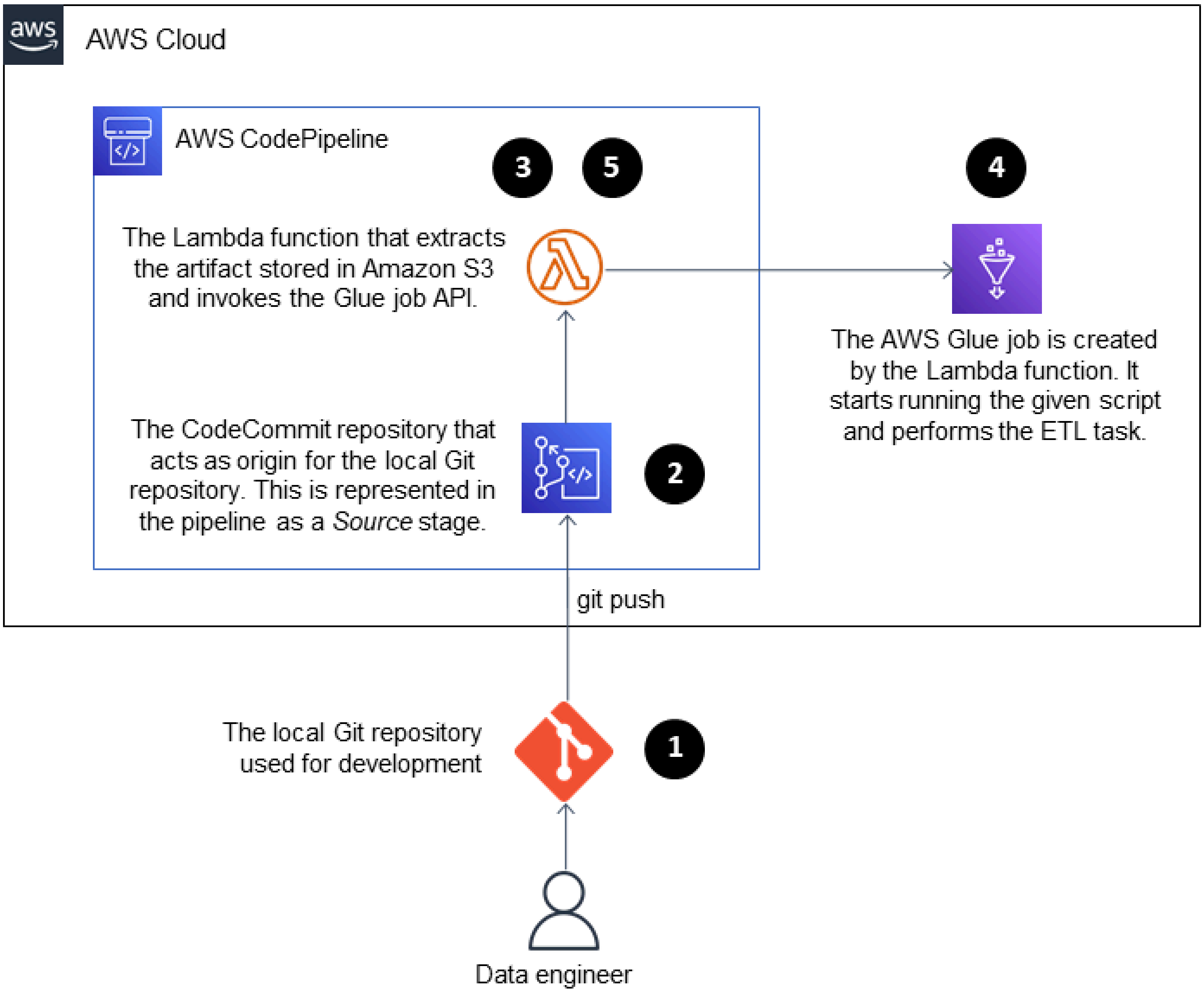
- The pipeline is finished as soon as the AWS Glue job is successfully launched. It does not wait for the conclusion of the job.
- The code provided in the attachment is intended for demo purposes only.

Architecture

Target technology stack

- AWS Glue
- AWS Lambda
- AWS CodePipeline
- AWS CodeCommit

Target architecture



The process consists of these steps:

1. The developer or data engineer makes a modification in the ETL code, commits, and pushes the change to AWS CodeCommit.
2. The push initiates the pipeline.
3. The pipeline initiates a Lambda function, which calls `codecommit:GetFile` on the repository and uploads the file to Amazon Simple Storage Service (Amazon S3).
4. The Lambda function launches a new AWS Glue job with the ETL code.
5. The Lambda function finishes the pipeline.

Automation and scale

The sample attachment demonstrates how you can integrate AWS Glue with AWS CodePipeline. It provides a baseline example that you can customize or extend for your own use. For details, see the *Epics* section.

Tools

- AWS CodePipeline – AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates.
- AWS CodeCommit – AWS CodeCommit is a fully managed source control service that hosts secure, Git-based repositories.
- AWS Lambda – AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers.
- AWS Glue – AWS Glue is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development.
- Git client – Git provides GUI tools, or you can use the command line or a desktop tool to check out the required artifacts from GitHub.
- AWS CDK – The AWS CDK is an open source software development framework that helps you define your cloud application resources by using familiar programming languages.

Epics

Deploy the sample code

Task	Description	Skills required
Configure the AWS CLI.	Configure the AWS Command Line Interface (AWS CLI) to target and authenticate with your current AWS account. For instructions, see the [AWS CLI documentation]().	Developer, DevOps engineer
Extract the sample project files.	Extract the files from the attachment to create a folder that contains the sample project files.	Developer, DevOps engineer

Deploy the sample code.	After you extract the files, run the following commands from the extract location to create a baseline example:
	`cdk bootstrap`
	`cdk deploy`
	`git init`
	`git remote add origin <code-commit-repository-url>`
	`git stage .`
	`git commit -m "adds sample code"`
	`git push --set-upstream origin main`
	After the last command, you can monitor the status of the pipeline and the AWS Glue job.
		Developer, DevOps engineer

Task	Description	Skills required
Customize the code.	Customize the code for the etl.py file in accordance with your business requirements. You can revise the ETL code, modify the pipeline stages, or extend the solution.	Data engineer

Related resources

- Getting started with the AWS CDK
- Adding jobs in AWS Glue
- Source action integrations in CodePipeline
- Invoke an AWS Lambda function in a pipeline in CodePipeline
- AWS Glue programming
- AWS CodeCommit GetFile API

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Deploy an Amazon EKS cluster from AWS Cloud9 using an EC2 instance profile

Sagar Panigrahi, Amazon Web Services

Summary

Notice: AWS Cloud9 is no longer available to new customers. Existing customers of AWS Cloud9 can continue to use the service as normal. Learn more

This pattern describes how to use AWS Cloud9 and AWS CloudFormation to create an Amazon Elastic Kubernetes Service (Amazon EKS) cluster that can be operated without enabling programmatic access for users in your Amazon Web Services (AWS) account.

AWS Cloud9 is a cloud-based integrated development environment (IDE) that helps you write, run, and debug your code by using a browser. AWS Cloud9 is used as a control center that provisions an Amazon EKS cluster by using Amazon Elastic Compute Cloud (Amazon EC2) instance profiles and AWS CloudFormation templates.

You can use this pattern if you don't want to create AWS Identity and Access Management (IAM) users and want to use IAM roles instead. Role-based access control (RBAC) regulates access to resources based on the roles of individual users. This pattern demonstrates how to update RBAC within an Amazon EKS cluster to allow access to a specific IAM role.

The pattern's setup also helps your DevOps team use AWS Cloud9 features to maintain and develop infrastructure as code (IaC) resources for creating Amazon EKS infrastructure.

Prerequisites and limitations

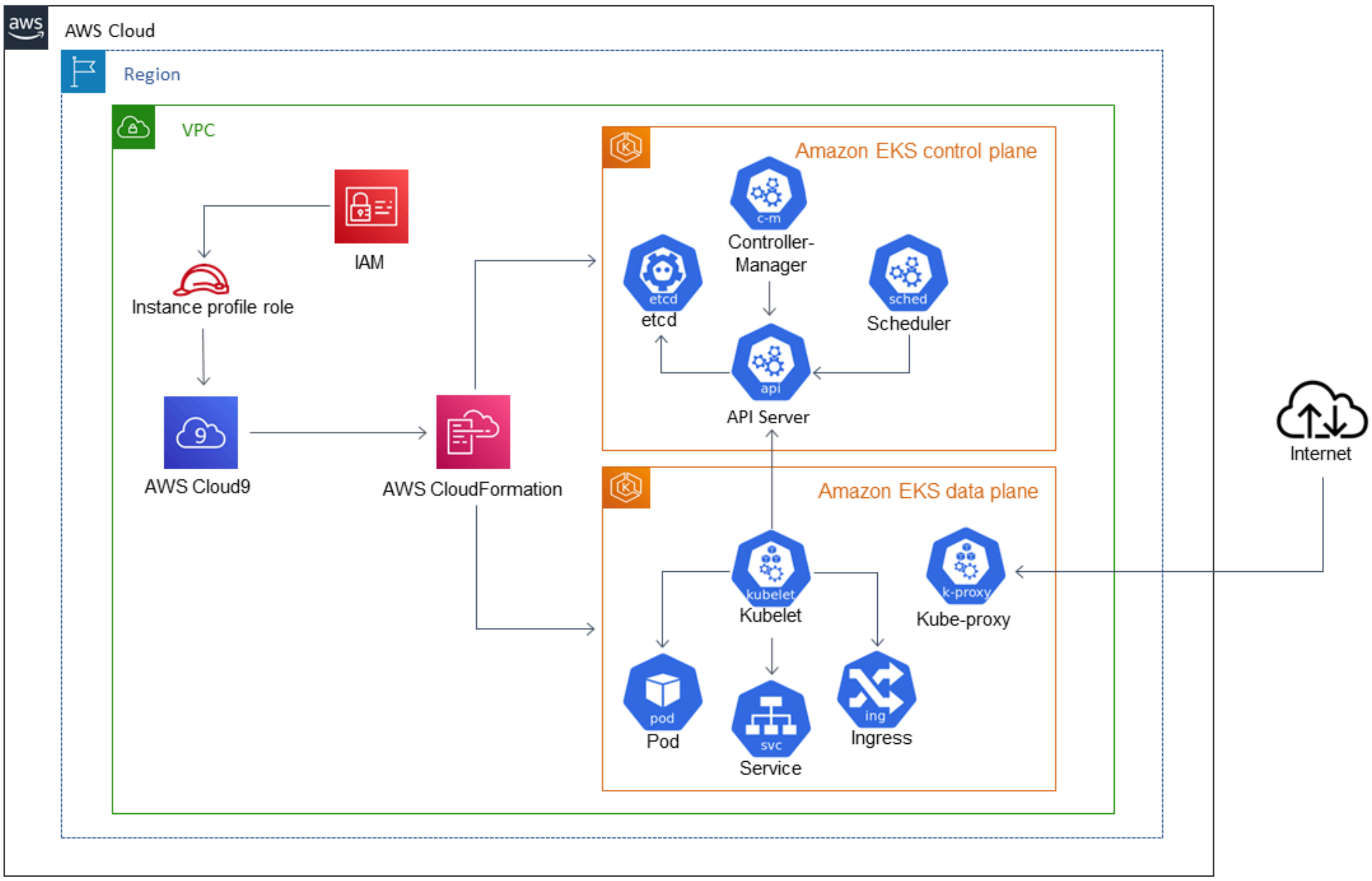
Prerequisites

- An active AWS account.
- Permissions to create IAM roles and policies for the account. The IAM role for the user must include the `AWSCloud9Administrator` policy. The `AWSServiceRoleForAmazonEKS` and `eksNodeRoles` roles must also be created because they are required to create an Amazon EKS cluster.
- Knowledge of Kubernetes concepts.

Limitations

- This pattern describes how to create a basic Amazon EKS cluster. For production clusters, you must update the AWS CloudFormation template.
- The pattern doesn't deploy additional Kubernetes components (for example, Fluentd, ingress controllers, or storage controllers).

Architecture



Technology stack

- AWS Cloud9
- AWS CloudFormation
- Amazon EKS
- IAM

Automation and scale

You can expand this pattern and incorporate it into continuous integration and continuous deployment (CI/CD) pipelines to automate the complete provisioning of Amazon EKS.

Tools

- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications.
- AWS Cloud9 – AWS Cloud9 offers a rich code-editing experience with support for several programming languages and runtime debuggers, and a built-in terminal.
- AWS CLI – AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services using commands in your command-line shell.
- Kubectl – `kubectl` is a command line utility that you can use to interact with an Amazon EKS cluster.

Epics

Create the IAM roles for the EC2 instance profile

Task	Description	Skills required
Create the IAM policy.	Sign in to the AWS Management Console, open the IAM console, choose **Policies**, and then choose **Create policy**. Choose the **JSON** tab and paste the contents from the policy-ro le-eks-instance-profile-for-cloud9.json file (attached). Resolve any security warnings, errors, or general warnings generated during the policy validation, and	Cloud administrator

Task	Description	Skills required
	then choose **Review policy**. Enter a **Name** for the policy. We recommend that you use `eks-instance-profile-for-cloud9` for the policy name. Review the policy **Summary** to see the permissions that are granted by your policy. Then choose **Create policy**.	
Create an IAM role using the policy.	On the IAM console, choose **Roles** and then choose **Create role**. Choose **AWS Service** and then choose **EC2** from the list. Choose **Next: Permissions** and search for the IAM policy that you created earlier. Choose the appropriate tags for your requirements. In the **Review** section, enter a name for the role. We recommend that you use `role-eks-instance-profile-for-cloud9` for the role name. Then choose **Create role**.	Cloud administrator

Create an IAM policy and role for the Amazon EKS RBAC

Task	Description	Skills required
Create the IAM policy.	On the IAM console, choose **Policies** and then choose **Create policy**. Choose the **JSON** tab and paste the contents from the policy-for-eks-rbac.json file (attached). Resolve any security warnings, errors, or general warnings generated during the policy validation, and then choose **Review policy**. Enter a **Name** for the policy. We recommend that you use `policy-for-eks-rbac` for the policy name. Review the policy **Summary** to see the permissions that are granted by your policy. Then choose **Create policy**.	Cloud administrator
Create an IAM role using the policy.	On the IAM console, choose **Roles** and then choose **Create role**. Choose **AWS Service** and then choose **EC2** from the list. Choose **Next: Permissions** and search for the IAM policy that you created earlier. Choose the appropriate tags for your requirements. In the **Review** section, enter a name for the role. We	Cloud administrator

Task	Description	Skills required
	recommend that you use `role-eks-admin-for-rbac` for the role name. Then choose **Create role**.	

Create the AWS Cloud9 environment

Task	Description	Skills required
Create the AWS Cloud9 environment.	Open the AWS Cloud9 console and choose **Create environment**. On the **Name environment** page, enter a name for your environment. We recommend that you use `eks-management-env` for the environment name. Configure the remaining settings according to your requirements and then choose **Next step**. On the **Review** page, choose **Create environment**. Wait while AWS Cloud9 creates your environment. This can take several minutes. For more information about the available configuration options, see [Creating an EC2 environment](#) in the AWS Cloud9 documentation.	Cloud administrator

Task	Description	Skills required
Remove the temporary IAM credentials for AWS Cloud9.	After your AWS Cloud9 environment is provisioned, choose **Settings** in the gear icon. Under **Preferences**, choose **AWS settings** and then choose **Credentials**. Turn off **AWS managed temporary credentials** and close the tab.	Cloud administrator
Attach the EC2 instance profile to the underlying EC2 instance.	Open the Amazon EC2 console and choose the EC2 instance that matches your environment in AWS Cloud9. If you used the name that we recommended, the EC2 instance is called `aws-cloud9-eks-management-env` . Choose the EC2 instance, choose **Actions**, and then choose **Instance settings**. Choose **Attach/replace IAM role**. Search for `role-eks-instance-profile-for-cloud9` or the name of the IAM role that you created earlier, and then choose **Apply**.	Cloud administrator

Create the Amazon EKS cluster

Task	Description	Skills required
Create the Amazon EKS cluster.	Download and open the eks-cfn.yaml (attached) template for AWS CloudFormation. Edit the template according to your requirements. Open the AWS Cloud9 environment and choose **New file**. Paste the AWS CloudFormation template that you created earlier into the field. We recommend that you use eks-cfn.yaml for the template name. In the AWS Cloud9 terminal, run the following command to create the Amazon EKS cluster: `aws cloudformation create-stack --stack-name eks-cluster --template-body file://eks-cfn.yaml --region <your_AWS_Region>` If the AWS CloudFormation call is successful, you receive the AWS CloudFormation stack's Amazon Resource Name (ARN) in your output.	Cloud administrator

Task	Description	Skills required
	The stack creation can take between 10 to 20 minutes.	
Verify the Amazon EKS cluster's status.	On the AWS CloudFormation console, open the **Stacks** page and then choose the stack name. The stack is created when the stack status code shows `CREATE_COMPLETE` . For more information, see Viewing AWS CloudFormation stack data and resources in the AWS CloudFormation documentation.	Cloud administrator

Access the Kubernetes resources in the Amazon EKS cluster

Task	Description	Skills required
Install kubectl in the AWS Cloud9 environment.	Install `kubectl` in your AWS Cloud9 environment by following the instructions from Installing kubectl in the Amazon EKS documentation.	Cloud administrator
Update the new Amazon EKS configuration in AWS Cloud9.	Run the following command in the AWS Cloud9 terminal to update the `kubeconfig` from the Amazon EKS cluster to the AWS Cloud9 environme nt:	Cloud administrator

Task	Description	Skills required
	`aws eks update-kubeconfig --name EKS-DEV2 --region <your_AWS_Region>` **Important** `EKS-DEV2` is the name of the Amazon EKS cluster in the AWS CloudFormation template that you used to create the cluster. Run the `kubectl get all -A` command to view all Kubernetes resources.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add the administrator IAM role to the Kubernetes RBAC.</td><td>Run the following command in your AWS Cloud9 terminal to open the RBAC configura tion map for Amazon EKS in edit mode:

<code>kubectl edit cm/aws-auth -n kube-system</code>

Append the following lines under the <code>mapRoles</code> section:

<pre>- groups:
- system:masters
rolearn: <ARN_of_IAM_role _from_second_epic>
username: eksadmin</pre>
Lint the YAML-formatted file to avoid syntax errors. Save the file using <code>vi</code> commands and then exit the file.

Note
By adding this section, you inform the Kubernetes RBAC that <code><ARN_of_IAM_role _from_second_epic></code> is to receive full administr ator access on the Amazon EKS cluster. This means that</td><td>Cloud administrator</td></tr>
</table>

Task	Description	Skills required
	the identified IAM role can carry out administrative actions on the Kubernetes cluster. AWS adds the existing section under `mapRoles` while the Amazon EKS cluster is provisioned.	

Related resources

References

- Modular and scalable Amazon EKS architecture (Quick Start)
- Managing users or IAM roles for your Amazon EKS cluster
- AWS CloudFormation template to create a new Amazon EKS control plane

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Deploy code in multiple AWS Regions using AWS CodePipeline, AWS CodeCommit, and AWS CodeBuild

Anand Krishna Varanasi, Amazon Web Services

Summary

This pattern demonstrates how to build infrastructure or architecture across multiple Amazon Web Services (AWS) Regions by using AWS CloudFormation. It includes continuous integration (CI)/ continuous deployment (CD) across multiple AWS Regions for faster deployments. The steps in this pattern have been tested for the creation of an AWS CodePipeline job to deploy to three AWS Regions as an example. You can change the number of Regions based on your use case.

Prerequisites and limitations

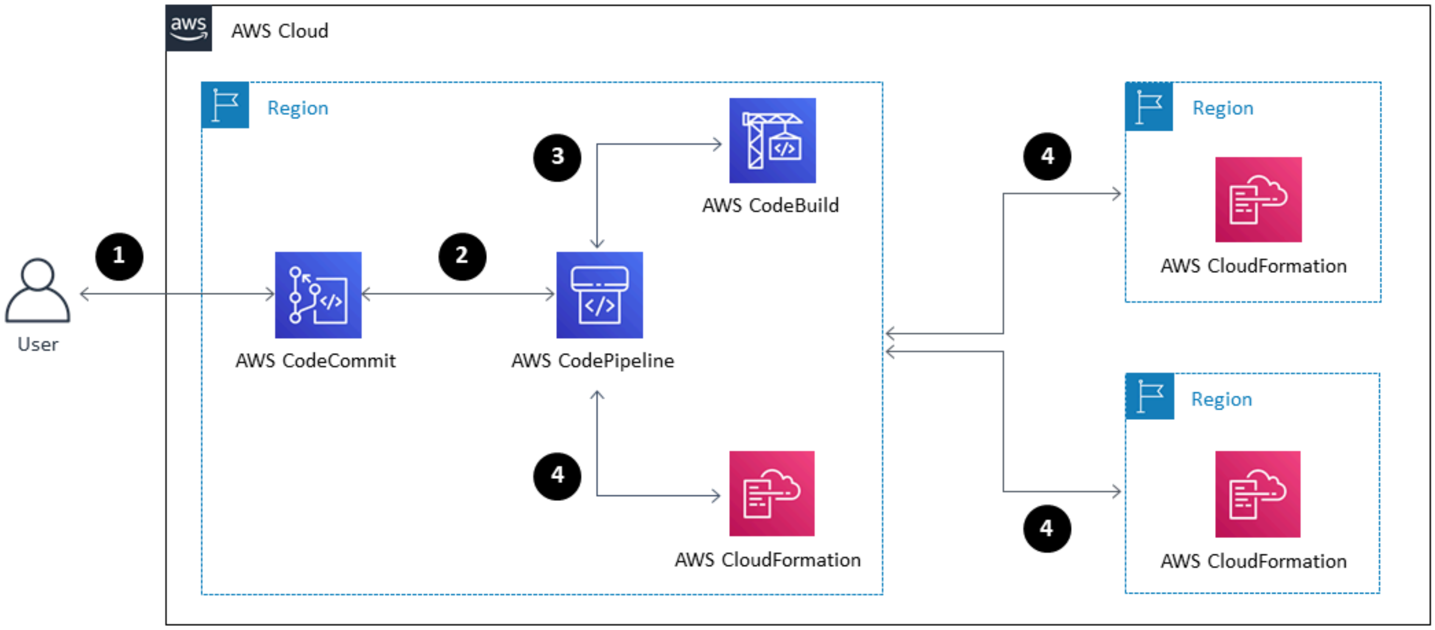
Prerequisites

- An active AWS account.
- - A CodeBuild role with the *AmazonS3FullAccess* and *CloudWatchFullAccess* policies. These policies give CodeBuild access to watch events of AWS CodeCommit through Amazon CloudWatch and to use Amazon Simple Storage Service (Amazon S3) as an artifact store.
  - An AWS CloudFormation role with the following policies, which give AWS CloudFormation, in the final Build stage, the ability to create or update AWS Lambda functions, push or watch Amazon CloudWatch logs, and to create and update change sets.
    - *AWSLambdaFullAccess*
    - *AWSCodeDeployFullAccess*
    - *CloudWatchFullAccess*
    - *AWSCloudFormationFullAccess*
    - *AWSCodePipelineFullAccess*

> **Note**
>
> Two AWS Identity and Access Management (IAM) roles for AWS CodeBuild and AWS CloudFormation with proper policies for CodeBuild to perform the CI tasks of testing, bundling, packaging the artifacts, and deploying to multiple AWS Regions in parallel.

Cross-check the policies created by CodePipeline to verify that CodeBuild and AWS CloudFormation have proper permissions in the CI and CD phases.

Architecture



This pattern's multiple-Region architecture and workflow comprise the following steps.

1. You send your code to a CodeCommit repository.
2. Upon receiving any code update or commit, CodeCommit invokes a CloudWatch event, which in turn starts a CodePipeline job.
3. CodePipeline engages the CI that is handled by CodeBuild. The following tasks are performed.
   - Testing of the AWS CloudFormation templates (optional)
   - Packaging of the AWS CloudFormation templates for each Region included in the deployment. For example, this pattern deploys in parallel to three AWS Regions, so CodeBuild packages the AWS CloudFormation templates into three S3 buckets, one in each specified Region. The S3 buckets are used by CodeBuild as artifact repositories only.
4. CodeBuild packages the artifacts as input for next Deploy phase, which runs in parallel in the three AWS Regions. If you specify a different number of Regions, CodePipeline will deploy to those Regions.

Tools

Tools

- [AWS CodePipeline](#) – CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software changes continuously.
- AWS CodeBuild – CodeBuild is a fully managed build service that compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- AWS CodeCommit – CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as source code and binary files) in the cloud.
- AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS.
- AWS Identity and Access Management – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers.

Code

The following sample code is for the `BuildSpec.yaml` file (Build phase).

```
---
artifacts:
discard-paths: true
files:
- packaged-first-region.yaml
- packaged-second-region.yaml
- packaged-third-region.yaml
phases:
build:
commands:
- echo "********BUILD PHASE - CF PACKAGING**********"
- "aws cloudformation package --template-file sam-template.yaml --s3-bucket
 $S3_FIRST_REGION --output-template-file packaged-first-region.yaml --region
 $FIRST_REGION"
```

```
- "aws cloudformation package --template-file sam-template.yaml --s3-bucket
 $S3_SECOND_REGION --output-template-file packaged-second-region.yaml --region
 $SECOND_REGION"
- "aws cloudformation package --template-file sam-template-anand.yaml --s3-bucket
 $S3_THIRD_REGION --output-template-file packaged-third-region.yaml --region
 $THIRD_REGION"
install:
commands:
- echo "********BUILD PHASE - PYTHON SETUP**********"
runtime-versions:
python: 3.8
post_build:
commands:
- echo "********BUILD PHASE - PACKAGING COMPLETION**********"
pre_build:
commands:
- echo "********BUILD PHASE - DEPENDENCY SETUP**********"
- "npm install --silent --no-progress"
- echo "********BUILD PHASE - DEPENDENCY SETUP DONE**********"
version: 0.2
```

Epics

Prepare the code and the CodeCommit repository

Task	Description	Skills required
Select the primary AWS Region for the deployment.	Sign in to your AWS account and choose the primary Region for the deployment. The CodeCommit repository will be in the primary Region.	DevOps
Create the CodeCommit repository.	Create the CodeCommit repository, and push the required code into it. The code generally includes the AWS CloudFormation or AWS SAM templates, Lambda code if any, and the CodeBuild `buildspec.yaml` files as	DevOps

Task	Description	Skills required
	input to the AWS CodePipel ine.	
Push the code into the CodeCommit repository.	In the *Attachments* section, download the code for this example, and then push the required code into it. Generally, the code can include AWS CloudForm ation or AWS SAM templates , Lambda code and the CodeBuild `buildspec .yaml` files as input to the pipeline.	DevOps

Source phase: Create the pipeline

Task	Description	Skills required
Create the CodePipeline job.	On the CodePipeline console, choose **Create pipeline**.	DevOps
Name the CodePipeline job and choose the service role setting.	Enter a name for the job, and keep the default service role setting so that CodePipel ine creates the role with the necessary policies attached.	DevOps
Specify the location for the artifact store.	Under **Advanced settings**, keep the default option so that CodePipeline creates an S3 bucket to use for code artifact storage. If you use an existing S3 bucket instead, the bucket must be in the	DevOps

Task	Description	Skills required
	primary Region that you specified in the first epic.	
Specify the encryption key.	Keep the default option, **Default AWS Managed Key**, or choose to use your own AWS Key Management Service (AWS KMS) customer managed key.	DevOps
Specify the source provider.	Under **Source provider**, choose **AWS CodeCommit**.	DevOps
Specify the repository.	Choose the CodeCommit repository that you created in the first epic. If you placed the code in a branch, choose the branch.	DevOps
Specify how code changes are detected.	Keep the default, **Amazon CloudWatch Events**, as the change trigger for CodeCommit to start the CodePipeline job.	DevOps

Build phase: Configure the pipeline

Task	Description	Skills required
Specify the build provider.	For the build provider, choose **AWS CodeBuild**.	DevOps
Specify the AWS Region.	Choose the primary Region, which you specified in the first epic.	DevOps

Build phase: Create and configure the project

Task	Description	Skills required
Create the project	Choose **Create project**, and enter a name for the project.	DevOps
Specify the environment image.	For this pattern demonstration, use the default CodeBuild managed image. You also have the option to use a custom Docker image if you have one.	DevOps
Specify the operating system.	Choose either Amazon Linux 2 or Ubuntu. **Note** Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs.	DevOps
Specify the service role.	Choose the role you created for CodeBuild before you started to create the CodePipeline job. (See the *Prerequisites* section.)	DevOps
Set additional options.	For **Timeout** and **Queued timeout**, keep the default values. For certificate, keep the default setting unless you have a custom certificate that you want to use.	DevOps

Task	Description	Skills required
Create the environment variables.	For each AWS Region that you want to deploy to, create environment variables by providing the S3 bucket name and the Region name (for example, us-east-1).	DevOps
Provide the buildspec file name, if it is not buildspec .yml.	Keep this field blank if the file name is the default, `buildspec.yaml` . If you renamed the buildspec file, enter the name here. Make sure it matches the name of the file that is in the CodeCommit repository.	DevOps
Specify logging.	To see logs for Amazon CloudWatch Events, keep the default setting. Or you can define any specific group or logger names.	DevOps

Skip the Deploy phase

Task	Description	Skills required
Skip the deploy phase and complete the creation of the pipeline.	When you set up the pipeline, CodePipeline allows you to create only one stage in the Deploy phase. To deploy to multiple AWS Regions, skip this phase. After the pipeline is created, you can	DevOps

Task	Description	Skills required
	add multiple Deploy phase stages.	

Deploy phase: Configure the pipeline for deployment to the first Region

Task	Description	Skills required
Add a stage to the Deploy phase.	Edit the pipeline and choose **Add stage** in the Deploy phase. This first stage is for the primary Region.	DevOps
Provide an action name for the stage.	Enter a unique name that reflects the first (primary) stage and Region. For example, enter **primary_<region>_deploy**.	DevOps
Specify the action provider.	For **Action provider**, choose AWS CloudFormation.	DevOps
Configure the Region for the first stage.	Choose the first (primary) Region, the same Region where CodePipeline and CodeBuild are set up. This is the primary Region where you want to deploy the stack.	DevOps
Specify the input artifact.	Choose **BuildArtifact**. This is the output of the build phase.	DevOps
Specify the action to take.	For **Action mode**, choose **Create or update a stack**.	DevOps
Enter a name for the CloudFormation stack.		DevOps

Task	Description	Skills required
Specify the template for the first Region.	Select the Region-specific package name that was packaged by CodeBuild and dumped into the S3 bucket for the first (primary) Region.	DevOps
Specify the capabilities.	Capabilities are required if the stack template includes IAM resources or if you create a stack directly from a template that contains macros. For this pattern, use CAPABILITY_IAM, CAPABILITY_NAMED_IAM, CAPABILITY_AUTO_EXPAND.	DevOps

Deploy phase: Configure the pipeline for deployment to the second Region

Task	Description	Skills required
Add the second stage to the Deploy phase.	To add a stage for the second Region, edit the pipeline and choose **Add stage** in the Deploy phase. Important: The process of creating the second Region is the same as that of the first Region, except for the following values.	DevOps
Provide an action name for the second stage.	Enter a unique name that reflects the second stage and the second Region.	DevOps

Task	Description	Skills required
Configure the Region for the second stage.	Choose the second Region where you want to deploy the stack.	DevOps
Specify the template for the second Region.	Select the Region-specific package name that was packaged by CodeBuild and dumped into the S3 bucket for the second Region.	DevOps

Deploy phase: Configure the pipeline for deployment to the third Region

Task	Description	Skills required
Add the third stage to the Deploy phase.	To add a stage for the third Region, edit the pipeline and choose **Add stage** in the Deploy phase. Important : The process of creating the second Region is the same as that of the previous two Regions, except for the following values.	DevOps
Provide an action name for the third stage.	Enter a unique name that reflects the third stage and the third Region.	DevOps
Configure the Region for the third stage.	Choose the third Region where you want to deploy the stack.	DevOps
Specify the template for the third Region.	Select the Region-specific package name that was packaged by CodeBuild and	DevOps

Task	Description	Skills required
	dumped into the S3 bucket for the third Region.	

Clean up the deployment

Task	Description	Skills required
Delete the AWS resources.	To clean up the deploymen t, delete the CloudFormation stacks in each Region. Then delete the CodeCommit, CodeBuild, and CodePipeline resources from the primary Region.	DevOps

Related resources

- What is AWS CodePipeline?
- AWS Serverless Application Model
- AWS CloudFormation
- AWS CloudFormation architecture structure reference for AWS CodePipeline

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Deploy artifact updates to on-premises Amazon EC2 instances using AWS CodePipeline

Akash Kumar and Sandeep Reddy Jogammagari, Amazon Web Services

Summary

This pattern provides code samples and steps to create a pipeline in the Amazon Web Services (AWS) Cloud and deploy updated artifacts to on-premises Amazon Elastic Compute Cloud (Amazon EC2) instances in AWS CodePipeline. The pattern is based on the continuous integration practice. This practice automates the building and testing of code with a Git-based version control system. In this pattern, you create and clone a code repository by using AWS CodeCommit. Then, you create a project and configure your source code by using AWS CodeBuild. Finally, you create your application and configure its target environment for on-premises EC2 instances by using AWS CodeDeploy.

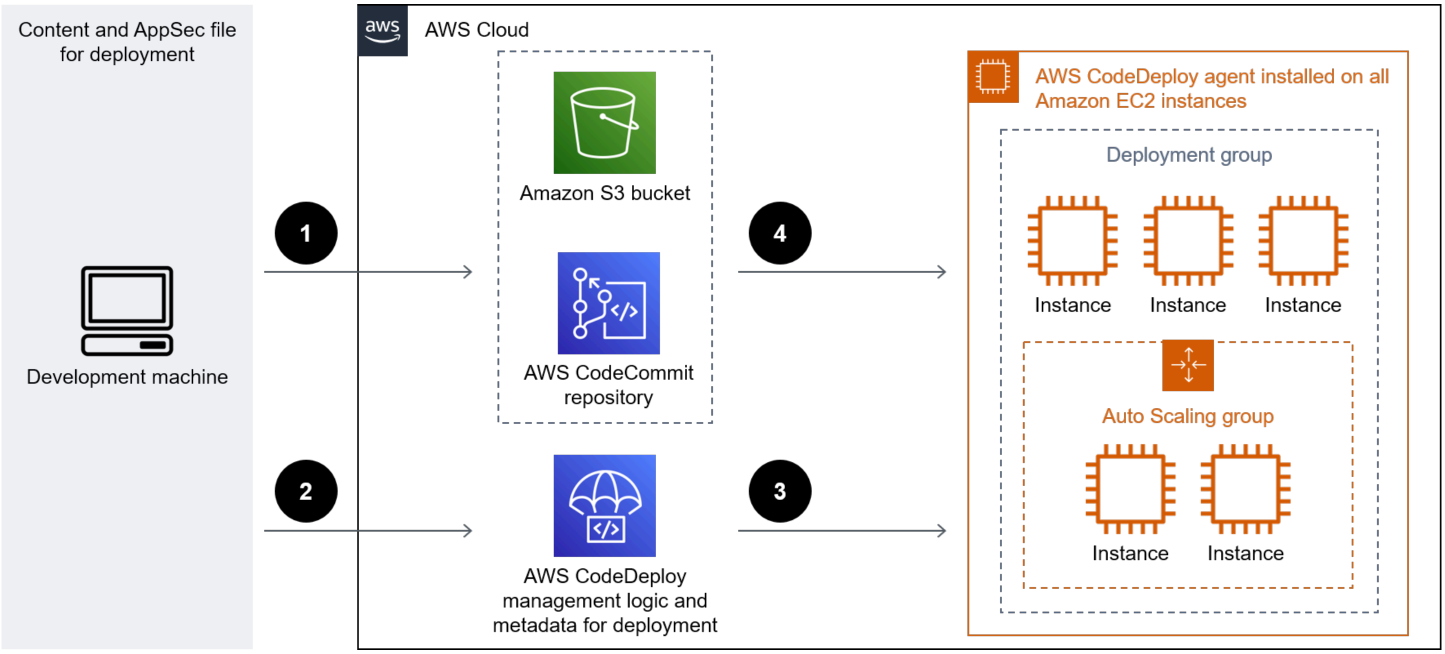
Prerequisites and limitations

Prerequisites

- An active AWS account
- User-defined tags to identify EC2 instances during deployment
- CodeDeploy agent, installed on EC2 instances
- Your required runtime software, installed on EC2 instances
- Amazon Corretto 8 for the Java Development Kit
- Apache Tomcat web server, installed
- Amazon CloudWatch Events (optional)
- A key pair to log in to the web server (optional)
- An Apache Maven application project for a web application

Architecture

The following diagram shows an example Java web application that's deployed to on-premises EC2 instances by using this pattern's architecture.



The diagram shows the following workflow:

1. The developer commits code changes to a private CodeCommit Git repository.
2. CodePipeline uses CodeBuild to initiate the build and add new artifacts that are ready for deployment in the Amazon Simple Storage Service (Amazon S3) bucket.
3. CodePipeline uses the CodeDeploy agent to pre-install any dependencies required for the deployment artifact changes.
4. CodePipeline uses the CodeDeploy agent to deploy the artifacts from the S3 bucket to target EC2 instances. If enabled, CloudWatch Events can automatically start the pipeline when a change occurs in the source code.

Technology stack

- CodeBuild
- CodeCommit
- CodeDeploy
- CodePipeline
- CloudWatch Events (optional)

Tools

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS CodeDeploy automates deployments to Amazon Elastic Compute Cloud (Amazon EC2) or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.

Code

This pattern includes the following attachments:

- `buildspec.yml` – This file specifies the actions that CodeBuild requires to build and create an artifact for deployment.
- `appspec.yml` – This file specifies the actions that CodeDeploy requires to create an application and configure a target environment for on-premises EC2 instances.
- `install_dependencies.sh` – This file installs dependencies for the Apache Tomcat web server.
- `start_server.sh` – This file starts the Apache Tomcat web server.
- `stop_server.sh` – This file stops the Apache Tomcat web server.

Epics

Set up the code repository

Task	Description	Skills required
Create the repository.	Create a CodeCommit repository.	AWS systems administrator

Task	Description	Skills required
Clone the repository.	Connect to the CodeCommit repository by cloning the repository.	App developer
Push the source code to the remote repository.	1. Create a commit to add the `buildspec.yml` and `appspec.yml` files to your local repository. 2. Push the commit from your local repository to the remote CodeCommit repository.	App developer

Create a CodeBuild project for the application

Task	Description	Skills required
Create a build project.	1. Sign in to the AWS Management Console, open the AWS CodeBuild console, and then choose **Create build project**. 2. For **Project name**, enter the name of your project. 3. For **Source provider**, choose **AWS CodeCommit**. 4. For **Repository**, choose the repository where you want to build the code pipeline. 5. For **Environment image**, choose **Managed image** or **Custom image**.	AWS administrator, App developer

Task	Description	Skills required
	6. For **Operating system**, choose **Amazon Linux 2**. **Note** Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs. 7. For **RunTime(s)**, choose **Standard**. 8. For **Image**, choose **aws/codebuild/amazonlinux2-aarch64-standard:2.0**. 9. For **Image version**, choose **Always use the latest image for this runtime version**. 10.For **Service role**, choose **New service role** or **Existing service role**. 11.For **Build specifications**, choose **Use a buildspec file** or **Insert build commands**. 12.(Optional) Choose **Add artifact to configure artifacts**. 13.(Optional) To upload build output logs to Amazon	

Task	Description	Skills required
	CloudWatch, choose **CloudWatch logs**. 14. Choose **Create build project**.	

Configure artifact deployment for on-premises EC2 instances

Task	Description	Skills required
Create the application.	1. Sign in to the AWS Management Console, open the [AWS CodeDeploy console](), and then choose **Create application**. 2. For **Application name**, enter a name for your application. 3. For **Compute platform**, choose **EC2/On-premises**. 4. Choose **Create application** and then choose **Create deployment group**. 5. For **Deployment group name**, enter a name. 6. **Note** Create a [service role]() for CodeDeploy. : The service role must have permissions to grant CodeDeploy access to your	AWS systems administrator, App developer

Task	Description	Skills required
	target environment. 7. For **Service role**, choose the service role that you created in step 6. 8. For **Deployment type**, choose either **In-place** or **Blue/green** based on your business requirements. 9. For **Environment configuration**, choose the options that meet your business requirements. 10. (Optional) [Create a target group]() for your load balancer separately in the Amazon EC2 console, and then go back to the **Create deployment group** page of the AWS CodeDeploy console to choose your load balancer and target group. 11. Choose **Create deployment group**.	

Set up the pipeline

Task	Description	Skills required
Create the pipeline.	1. Sign in to the AWS Management Console,	AWS systems administrator, App developer

Task	Description	Skills required
	open the [AWS CodePipeline console](), and then choose **Create pipeline**. 2. For **Pipeline name**, enter a name for the pipeline. 3. For **Service role**, choose **New service role** or **Existing service role**. 4. For **Role name**, enter a name for your role. 5. In the **Advanced settings** section, for **Artifact store**, choose **Default location** if you want Amazon S3 to create a bucket and store the artifacts in the bucket. To use an existing S3 bucket, choose **Custom location**. Choose **Next**. 6. For **Source provider**, choose **AWS CodeCommit**. 7. For **Repository name**, choose the repository that you cloned earlier. For **Branch name**, choose your source code branch. 8. For **Change detection options**, choose **Amazon CloudWatch Events (recommended)** or **AWS CodePipeline**. Choose **Next**.	

Task	Description	Skills required
	9. For **Build provider**, choose **AWS CodeBuild**. 10. For **Project Name**, choose the build project that you created in the *Create a CodeBuild project for the application* section of this pattern. 11. Choose your build options and then choose **Next**. 12. For **Deploy provider**, choose **AWS CodeDeploy**. 13. Choose an application name and deployment group, and then choose **Next**. 14. Choose **Create pipeline**.	

Related resources

- Working with repositories in AWS CodeCommit
- Working with build projects
- Working with applications in CodeDeploy
- Working with pipelines in CodePipeline

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Deploy workloads from Azure DevOps pipelines to private Amazon EKS clusters

Mahendra Revanasiddappa, Amazon Web Services

Summary

This pattern demonstrates how to implement continuous integration and continuous delivery (CI/CD) from Azure DevOps pipelines to private Amazon Elastic Kubernetes Service (Amazon EKS) clusters. It addresses a critical challenge faced by organizations that are enhancing their security posture by transitioning to private API server endpoints for their Amazon EKS clusters.

A public endpoint exposes the Kubernetes API server directly to the internet, creating a larger attack surface that malicious actors could potentially target. By switching to a private endpoint, access to the cluster's control plane is restricted to within the customer's virtual private cloud (VPC).

Although transitioning an Amazon EKS cluster to a private API endpoint significantly enhances security, it introduces connectivity challenges for external CI/CD platforms like Azure DevOps. The private endpoint is only accessible from within the cluster's VPC or peered networks. Therefore, standard Microsoft-hosted Azure DevOps agents, operating outside the AWS private network, can't reach the Kubernetes API server directly. This breaks typical deployment workflows that rely on tools like kubectl or Helm running on these agents because they fail to establish a connection to the cluster.

To overcome this problem, this pattern showcases an efficient approach by using self-hosted Azure DevOps agents within private Amazon EKS clusters. This solution offers superior cost optimization, operational efficiency, and scalability while preserving security requirements. This approach particularly benefits enterprises seeking to streamline their multi-cloud DevOps processes without compromising on performance or security.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2.13.17 or later, installed.
- kubectl version 1.25.1 or later, installed.

- A private Amazon EKS cluster version 1.24 or later created, with permissions to create namespaces, secrets, and deployments.
- Worker nodes in an Amazon EKS cluster with outbound connectivity to the internet so that the Azure DevOps agent running on them can connect to Azure DevOps agent pool.
- GitHub account created.
- An Azure DevOps project with access to configure service connections, which are authenticated connections between Azure Pipelines and external or remote services, created.
- The AWS Toolkit for Azure DevOps version 1.15 or later installed for the Azure DevOps project described in the previous point. For installation instructions, see AWS Toolkit for Azure DevOps in Visual Studio Marketplace.

Limitations

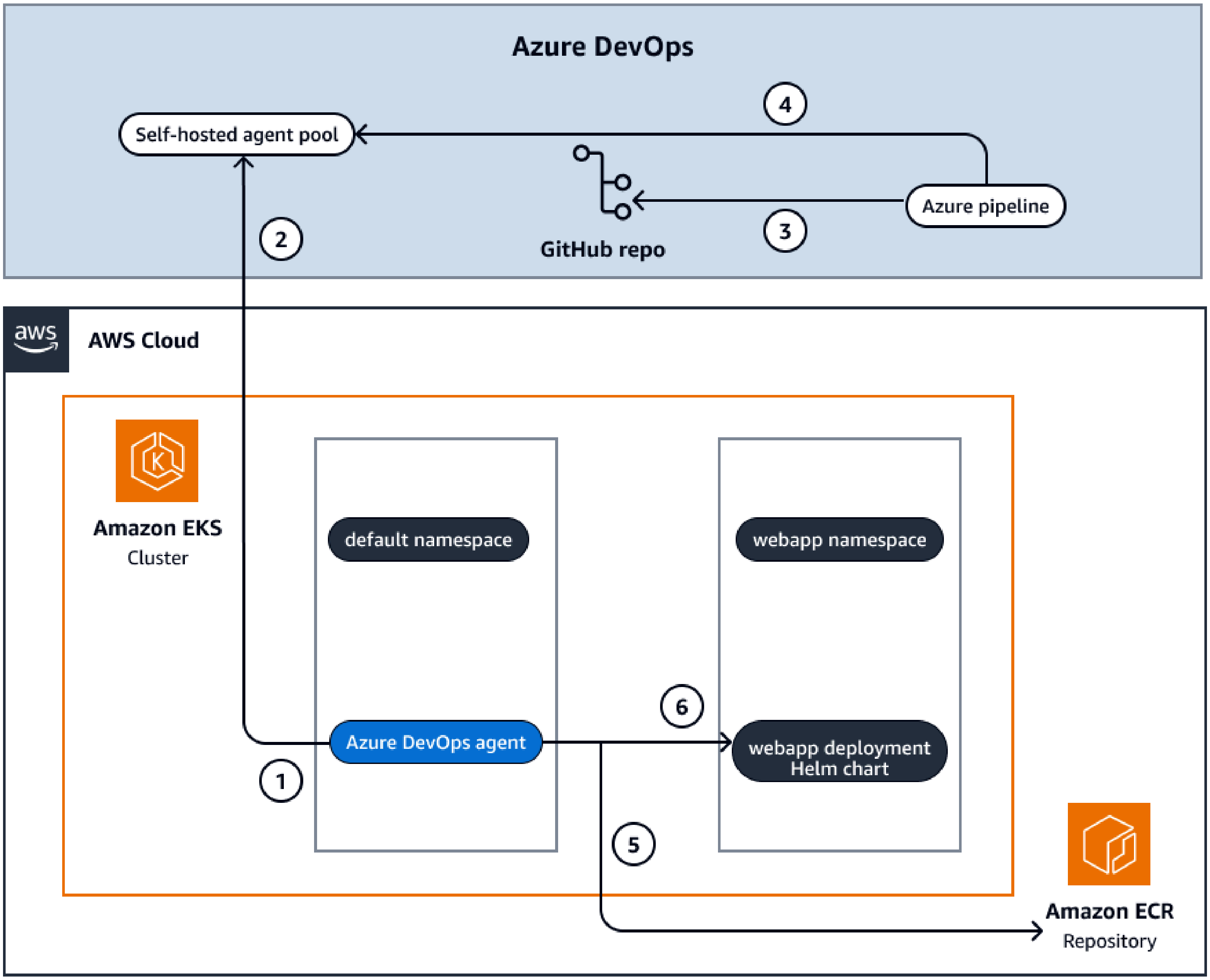
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

This pattern creates the following:

- **Amazon ECR repository** - The Amazon Elastic Container Registry (Amazon ECR) repository stores the Docker image with the Azure DevOps agent and the sample app that is deployed.
- **Azure DevOps agent pool** - An Azure DevOps self-hosted agent pool registers the agent running on the private Amazon EKS cluster.
- **IAM role** - An AWS Identity and Access Management (IAM) role for the Azure service connection to provide required access to the agent that's running on a private Amazon EKS cluster.
- **Azure DevOps service connection** - A service connection in an Azure DevOps account to use the IAM role that provides the required access for the pipeline jobs to access AWS services.

The following diagram shows the architecture of deploying a self-hosted Azure DevOps agent on a private Amazon EKS cluster and deploying a sample application on the same cluster.



The diagram shows the following workflow:

1. Deploy a self-hosted Azure DevOps agent as a deployment inside an Amazon EKS cluster.
2. An Azure DevOps agent connects to the agent pool on an Azure DevOps account using a personal access token (PAT) for authentication.
3. Azure Pipelines configures a pipeline to deploy by using code from a GitHub repository.
4. The pipeline runs on the agent from the agent pool that was configured in the pipeline configuration. The Azure DevOps agent gets the job information of the pipeline by constantly polling to the Azure DevOps account.
5. The Azure DevOps agent builds a Docker image as part of the pipeline job and pushes the image to the Amazon ECR repository.

6. The Azure DevOps agent deploys the sample application on a private Amazon EKS cluster in a namespace called `webapp`.

Tools

Tools

- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- kubectl is a command-line interface that helps you run commands against Kubernetes clusters.

Code repository

- The code for this pattern is available in the GitHub deploy-kubernetes-resources-to-amazon-eks-using-azure-devops repository.

Best practices

- For Amazon EKS, see the Amazon EKS Best Practices Guide.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Create a service connection

Task	Description	Skills required
Find the Azure DevOps organization GUID.	Sign in to your Azure DevOps account, and then use the following URL to find the organization GUID: `https://dev.azure.com/{DevOps_Org_ID}/_apis/projectCollections?api-version=6.0` In the URL, replace `{DevOps_org_ID}` with your Azure DevOps organization ID.	AWS DevOps
Configure an IdP in the AWS account.	To configure an Identity provider (IdP) in the AWS account for an Azure service connection, use the following steps: 1. Sign in to the AWS Management Console, and open the IAM console at https://console.aws.amazon.com/iam/. 2. In the left pane, choose **Identity providers**. 3. Choose **Add Provider**. 4. Choose **OpenID Connect** as the **Provider type**. 5. For **Provider URL**, enter the Azure DevOps issuer URL. Each tenant of Azure	AWS DevOps

Task	Description	Skills required
	DevOps has a unique `OrganizationGUID` that typically uses the following format: `https://vstoken.dev.azure.com/{OrganizationGUID}` Replace `{OrganizationGUID}` with your Azure DevOps organization ID. 6. For **Audience**, enter **api://AzureADTokenExchange**. This is a fixed value for Azure DevOps. 7. Choose **Add Provider**. 8. Take note of the ARN of the newly created provider for your use in the next task. For more details, see [How to federate into AWS from Azure DevOps using OpenID Connect]().	

Task	Description	Skills required

Create an IAM policy in the AWS account.	To create an IAM policy to provide the required permissions to the IAM role used by the Azure DevOps pipeline, use the following steps:
	
	1. In the IAM console, in the left pane, choose **Policies**.
	2. Choose **Create Policy**.
	3. For **Specify Permissions**, in **Policy editor**, select **JSON**. Replace the default JSON policy with the following JSON:
	
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Sid": "Statement1",`
	`            "Effect": "Allow",`
	`            "Action": [`
	`                "ecr:*",`
	`                "eks:DescribeCluster",`
	`                "eks:ListClusters"`
	`            ],`
	`            "Resource": "*"`
		AWS DevOps

Task	Description	Skills required

	``
	        }
	    ]
	}
	``
	4. Choose **Next.**
	5. For **Policy name**, enter a name for the IAM policy. This pattern uses the name **ADO-policy**.
	6. Choose **Create policy**.	

Task	Description	Skills required
Create an IAM role in the AWS account.	To configure an IAM role in the AWS account for the Azure service connection, use the following steps: 1. In the IAM console, in the left pane, choose **Roles**. 2. Choose **Create role**. 3. For **Trusted entity type**, select **Web Identity**. 4. Select the correct IdP from the dropdown list. The IdP name starts with `vstoken.dev.azure.com/{OrganizationGUID}` . 5. In the **Audience** dropdown list, select **api://AzureADTokenExchange**. 6. To limit this role to only one service connection, add a condition. Under **Condition**, select **Add condition**, and for **Key**, select **vstoken.dev.azure.com/{OrganizationGUID}:sub**. In **Condition**, select **StringEquals**. For **Value**, use the following format: `sc://{OrganizationName}/{ProjectName}/{ServiceConnectionName}` . For `ServiceCo`	AWS DevOps

Task	Description	Skills required

	nnectionName , use **aws-sc.** You will create this service connection in the next task.
	7. Choose **Next**.
	8. For **Add permissions**, select **ADO-policy**, which is the policy that you created in a previous task.
	9. Choose **Next**, and for **Role name**, enter **ado-role**. For **Select Trusted Entities**, use the following trust policy:
	`{`
	`  "Version": "2012-10-17",`
	`  "Statement": [`
	`    {`
	`      "Effect": "Allow",`
	`      "Principal": {`
	`        "Federated": "arn:aws:iam::{account_id}:oidc-provider/vstoken.dev.azure.com/{OrganizationGUID}"`
	`      },`
	`      "Action": "sts:AssumeRoleWithWebIdentity",`
	`      "Condition": {`
	`        "StringEquals": {`
	`          "vstoken.dev.azure.com/{Org`	

Task	Description	Skills required

	``
	anizationGUID}:aud
	": "api://AzureADToke
	nExchange",
	          "vstoken.
	dev.azure.com/{Org
	anizationGUID}:sub
	": "sc://{Organizatio
	nName}/{ProjectNam
	e}/{ServiceConnect
	ionName}"
	        }
	      }
	    }
	  ]
	}
	``
	
	In the policy, provide your information for the following placeholders:
	
	• `{account_id}` - AWS account ID
	• `{OrganizationGUID}` - Azure DevOps Organization GUID
	• `{OrganizationName}` - Azure DevOps Organization name
	• `{ProjectName}` - Azure DevOps project name
	• `{ServiceConnectionName}` - Azure DevOps service connection name. Use **aws-sc.** You will create this service connection in the next task.	

Task	Description	Skills required
Create a service connection in the Azure DevOps account.	To configure an Azure service connection, use the following steps: 1. In your Azure DevOps project, select **Project settings**, **Service connections**. 2. Choose **New service connection**, select the type of service connection as `aws`, and then select **Next**. 3. For **Role to Assume**, enter the arn for the IAM role **ado-role.** You created **ado-role** in the previous task *Create an IAM role in the AWS account*. 4. Select the **Use OIDC** checkbox. 5. For **Service connection name**, enter **aws-sc** in task properties. 6. Choose **Save**. For more details, see [Create a service connection](#) in the Microsoft documentation.	AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add IAM role to Amazon EKS configuration file.</td><td>The IAM role must have the necessary permissions to perform the required operations on the Amazon EKS cluster. Because it's a pipeline role, the IAM role must be able to manage almost all types of resources on the cluster. Therefore, the <code>system:masters</code> group permission is appropriate for this role.

To add the required configuration to the <code>aws-auth ConfigMap</code> within Kubernetes, use the following code:

<pre>- groups:
  - system:masters
  rolearn: arn:aws:iam::{account_id}:role/ADO-role
  username: ADO-role</pre>
Replace <code>{account_id}</code> with your AWS account ID.

For more information, see How Amazon EKS works with IAM in the Amazon EKS documentation.</td><td>AWS DevOps</td></tr>
</table>

Create an agent pool

Task	Description	Skills required
Create a self-hosted agent pool.	To configure a self-hosted agent pool in the Azure DevOps account, use the following steps: 1. Sign in to your Azure DevOps account organization. 2. Choose **Azure DevOps Organization**. 3. Choose your Azure DevOps project. 4. Choose **Project settings**. 5. Choose **Agent pools**. 6. Choose **Add pool**. 7. Select **Self-hosted**. 8. For **Name**, enter **eks-agent**. 9. Select the **Grant access permission to all pipelines** checkbox. 10. Choose **Create**. For more details, see [Create and manage agent pools] in the Microsoft documentation.	

Build Azure DevOps agent image and push to Amazon ECR

Task	Description	Skills required
Create an Amazon ECR repository.	The Docker images that are used to deploy the Azure DevOps agent and sample application (`webapp`) on the private Amazon EKS cluster must be stored in an Amazon ECR repository. To create an Amazon ECR repository, use the following steps: 1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories. 2. From the navigation bar, choose the **AWS Region** to create your repository in. 3. On the **Repositories** page, choose **Private repositories**, and then choose **Create repository**. 4. For **Repository name**, enter **webapp**. For the sample application in this pattern to work, the Amazon ECR repository name must use **webapp**. If you use a different name for the repository, see [Troubleshooting](#).	AWS DevOps

Task	Description	Skills required
	For more details, see Creating an Amazon ECR private repository to store images in the Amazon ECR documentation.	

Task	Description	Skills required

Create a Dockerfile to build the Azure DevOps agent.	Create a Dockerfile to build the Docker image that has the Azure DevOps agent installed. Store the following content in a file named `Dockerfile` :
	
	`FROM ubuntu:22.04`
	`ENV TARGETARCH="linux-x64"`
	`RUN apt update && apt upgrade -y && apt install -y curl git jq libicu70 unzip wget`
	
	`RUN curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
	`RUN unzip awscliv2.zip`
	`RUN ./aws/install`
	`RUN rm -rf aws awscliv2.zip`
	
	`RUN curl -sSL https://get.docker.com/ | sh`
	
	`RUN curl -sL https://aka.ms/InstallAzureCLIDeb | bash`
	`RUN mkdir -p azp`
	`WORKDIR /azp/`
	
	`COPY ./start.sh ./`
	`RUN chmod +x ./start.sh`
	
	`RUN useradd -m -d /home/agent agent`
		AWS DevOps

Task	Description	Skills required

	`RUN chown -R agent:agent /azp /home/agent`
	`RUN groupadd -f docker`
	`RUN usermod -aG docker agent`
	`USER agent`
	
	`ENTRYPOINT [ "./start.sh" ]`	

Create script for the Azure DevOps agent.	To create the `start.sh` script, use the following steps: 1. Go to the procedure [Create and build the Dockerfile](#) in the Microsoft documentation, and scroll to **step 5. Save the following content to** `~/azp-agent-in-docker/start.sh` **, making sure to use Unix-style (LF) line endings**. 2. Copy the script's content, and save it in a file named `start.sh` in the same directory as the Dockerfile.	AWS DevOps

Task	Description	Skills required

Build a Docker image with the Azure DevOps agent.	To create a Docker image to install the Azure DevOps agent, use the Dockerfile that you created earlier to build the image. In the same directory where the Dockerfile is stored, run the following commands:
	`aws ecr get-login-password --region region | docker login --username AWS --password-stdin aws_account_id.dkr.ecr.region.amazonaws.com`
	
	`docker build --platform linux/amd64 -t ado-agent:latest .`
	
	`docker tag ado-agent:latest aws_account_id.dkr.ecr.region.amazonaws.com/webapp:latest`
	
	`docker push aws_account_id.dkr.ecr.region.amazonaws.com/webapp:latest`
	
	Replace `aws_account_id` and `region` with your AWS account ID and AWS Region.
		AWS DevOps

Deploy the Azure DevOps agent to a private Amazon EKS cluster

Task	Description	Skills required
Generate an Azure personal access token.	The agent running on the private Amazon EKS cluster requires a personal access token (PAT) so that it can authenticate with the Azure DevOps account. To generate a PAT, use the following steps: 1. Sign in with the user account you plan to use in your Azure DevOps organization (`https://dev.azure.com/{Your_Organization}`). 1. From your home page, open your user settings, and then select **Personal access tokens**. 2. Choose **New Token.** 3. Enter the **Name** for the token. 4. Choose **Show all scopes.** 5. For **Agent Pools**, select the **Read & manage** checkbox. 6. Choose **Create.** 7. To create a secret on the private Amazon EKS cluster, use the following configuration:	AWS DevOps

Task	Description	Skills required

	<pre>apiVersion: v1
	kind: Secret
	metadata:
	  name: azdevops-pat
	  namespace: default
	type: Opaque
	stringData:
	  AZP_TOKEN: <PAT
	 Token></pre> 1. Store the configuration in a file named `ado-secret.yaml` . Replace `<PAT Token>` with the personal access token that you just created. To create the secret, run the following command: <pre>kubectl create -f ado-
	secret.yaml</pre> For more details, see Register an agent using a personal access token (PAT) in the Microsoft documentation.	

Task	Description	Skills required

Use the Kubernetes manifest file for agent deployment.	To deploy the Azure DevOps agent on the private Amazon EKS cluster, copy the following manifest file and store the file as `agent-deployment.yaml` :
	``apiVersion: apps/v1
	kind: Deployment
	metadata:
	  name: azure-pipelines-agent-eks
	  labels:
	    app: azure-pipelines-agent
	spec:
	  replicas: 1
	  selector:
	    matchLabels:
	      app: azure-pipelines-agent
	  template:
	    metadata:
	      labels:
	        app: azure-pipelines-agent
	    spec:
	      containers:
	      - name: docker
	        image: docker:dind
	        securityContext:
	          privileged: true
	        volumeMounts:
	        - name: shared-workspace
	          mountPath: /workspace```
		AWS DevOps

Task	Description	Skills required

	``
	        - name: dind-
	storage
	          mountPath: /
	var/lib/docker
	        env:
	        - name:
	 DOCKER_TLS_CERTDIR
	          value: ""
	      - name: azure-pip
	elines-agent
	        image: aws_accou
	nt_id.dkr.ecr.regi
	on.amazonaws.com/w
	ebapp:latest
	        env:
	        - name: AZP_URL
	          value: "<Azure
	 account URL>"
	        - name: AZP_POOL
	          value: "eks-
	agent"
	        - name:
	 AZP_TOKEN
	          valueFrom:
	            secretKey
	Ref:
	              name:
	 azdevops-pat
	              key:
	 AZP_TOKEN
	        - name:
	 AZP_AGENT_NAME
	          valueFrom:
	            fieldRef:
	              fieldPath
	: metadata.name
	        - name:
	 DOCKER_HOST
	          value: tcp://
	localhost:2375
	        volumeMounts:
	``	

Task	Description	Skills required

	``
	        - mountPath: /
	workspace
	          name: shared-
	workspace
	      volumes:
	      - name: dind-stor
	age
	        emptyDir: {}
	      - name: shared-wo
	rkspace
	        emptyDir: {}
	``
	Replace `aws_account_id` and `<Azure account URL>` with your AWS account ID and Azure DevOps account URL.	

Deploy the agent on the private Amazon EKS cluster.	To deploy the Azure Devops agent on the private Amazon EKS cluster, use the following command: ``` kubectl create -f agent- deployment.tf ```	AWS DevOps

Task	Description	Skills required

Verify the agent is running.	To verify that the Azure DevOps agent is running, use the following command:
	`kubectl get deploy azure-pipelines-agent-eks`
	The expected output should be similar to the following:
	`NAME READY UP-TO-DATE AVAILABLE AGE`
	`azure-pipelines-agent-eks 1/1 1 1 58s`
	Make sure that the READY column shows 1/1.
		AWS DevOps

Task	Description	Skills required
Verify the agent is registered with the Azure DevOps agent pool.	To verify that the agent is deployed on the private Amazon EKS cluster and is registered with the agent pool `eks-agent`, use the following steps: 1. Sign in to your Azure DevOps organization (`https://dev.azure.com/{Your_Organization}`). 2. Choose **Project settings**. 3. Choose **Agent pools**. 4. Select the **eks-agent** pool, and then check the **Agents** tab. You should see one agent listed with a **Status** of **Online**, and the name of the agent should start with **azure-pipelines-agent-eks-***.	AWS DevOps

Deploy sample application

Task	Description	Skills required
Fork the sample application repository to your GitHub account.	Fork the following AWS Samples repository to your GitHub account: https://github.com/aws-samples/deploy-kubernetes-re	AWS DevOps

Task	Description	Skills required
	sources-to-amazon-eks-using-azure-devops	

Task	Description	Skills required
Create a pipeline.	To create a pipeline in your Azure DevOps account, use the following steps: 1. Sign in with the user account you plan to use in your Azure DevOps organization (`https://dev.azure.com/{Your_Organization}`). 2. Navigate to your project and pipelines console. 3. Choose **New Pipeline.** 4. For **Where is your code**, choose **GitHub.** 5. Provide the credentials required for the pipeline to connect to your GitHub account 6. Select the repository **deploy-kubernetes-resources-to-amazon-eks-using-azure-devops**. 7. For **Configure your pipeline,** select **Existing Azure Pipelines YAML file**. 8. For **Select an existing YAML file,** select **main** for **Branch** and **azure_pipelines.yaml** for **path**. 9. Choose **Continue**. 10For **Review your pipeline YAML**, replace the	AWS DevOps

Task	Description	Skills required

	input parameter values for `awsRegion` and `awsEKSClusterName` with your information:
	
	`pool:`
	`  name: eks-agent`
	`#pool: self-hosted # If you are running self-hosted Azure DevOps Agents`
	
	`stages:`
	`# Refering the pipeline template, input parameter that are not specified will be added with defaults`
	`- template: ./pipeline_templates/main_template.yaml`
	`  parameters:`
	`    serviceConnectionName: aws-sc`
	`    awsRegion: <your region>`
	`    awsEKSClusterName: <name of your EKS cluster>`
	`    projectName: webapp`
	
	1. Choose **RUN**.	

Task	Description	Skills required
Verify that the sample application deployed.	After the pipeline completes , verify the successful deployment of the sample application by checking both the Amazon ECR repository and the Amazon EKS cluster. To verify artifacts in the Amazon ECR repository, use the following steps: 1. Navigate to the webapp Amazon ECR repository. 2. Confirm the presence of the following new artifacts: • Docker image – <date>.<build_number>-image • Helm chart – <date>.<build_number>-helm For example, 20250501.1-image and 20250501.1-helm. To verify deployment on the private Amazon EKS cluster in the namespace webapp, use the following command: kubectl get deploy -n webapp	AWS DevOps

Task	Description	Skills required

	The expected output is as follows:
	`NAME      READY   UP-TO-DATE   AVAILABLE`
	`webapp   1/1     1            1`
	Note: If this is your first pipeline run, you might need to authorize the service connection and agent pool. Look for permission requests in the Azure DevOps pipeline interface, and approve them to proceed.	

Troubleshooting

Issue	Solution
Pipeline fails when Amazon ECR repository name doesn't match `webapp`	The sample application expects the Amazon ECR repository name to match the `projectName: webapp` parameter in `azure_pipeline.yml` . To resolve this issue, rename your Amazon ECR repository to `webapp`, or update the following: • Rename the `webapp` directory in your forked GitHub repository to match your Amazon ECR repository name.

Issue	Solution
	• Update the `projectName` parameter in `azure_pipeline.yml` to match your Amazon ECR repository name.
Error: Kubernetes cluster unreachable: the server has asked for the client to provide credentials	If you encounter this error in the "Pull and Deploy Helm Chart" step in your Azure pipeline, the root cause typically stems from an incorrect IAM role configuration in your Amazon EKS cluster's `aws-auth ConfigMap` . To resolve this issue, check the following: • Verify your `aws-auth ConfigMap` configuration. • Check your Amazon EKS cluster's authentic ation settings: Open the Amazon EKS console, **Cluster details**, **Access configura tion**. Make sure that **Authentication mode** is set to **EKS API and ConfigMap** (not just **EKS API**).

Related resources

AWS Blogs

- How to federate into AWS from Azure DevOps using OpenID Connect

AWS services documentation

- Amazon ECR
- Amazon EKS

Microsoft documentation

- What is Azure DevOps?
- What is Azure Pipelines?

Execute Amazon Redshift SQL queries by using Terraform

Sylvia Qi and Aditya Ambati, Amazon Web Services

Summary

Using infrastructure as code (IaC) for the deployment and management of Amazon Redshift is a prevalent practice within DevOps. IaC facilitates the deployment and configuration of various Amazon Redshift resources, such as clusters, snapshots, and parameter groups. However, IaC doesn't extend to the management of database resources like tables, schemas, views, and stored procedures. These database elements are managed through SQL queries and are not directly supported by IaC tools. Although solutions and tools exist for managing these resources, you might prefer not to introduce additional tools into your technology stack.

This pattern outlines a methodology that uses Terraform to deploy Amazon Redshift database resources, including tables, schemas, views, and stored procedures. The pattern distinguishes between two types of SQL queries:

- **Nonrepeatable queries –** These queries are executed once during the initial Amazon Redshift deployment to establish the essential database components.
- **Repeatable queries –** These queries are immutable and can be rerun without impacting the database. The solution uses Terraform to monitor changes in repeatable queries and apply them accordingly.

For more details, see *Solution walkthrough* in Additional information.

Prerequisites and limitations

Prerequisites

You must have an active AWS account and install the following on your deployment machine:

- AWS Command Line Interface (AWS CLI)
- An AWS CLI profile configured with Amazon Redshift read/write permissions
- Terraform version 1.6.2 or later
- Python3
- Boto3

Limitations

- This solution supports a single Amazon Redshift database because Terraform only allows for the creation of one database during cluster creation.
- This pattern doesn't include tests to validate changes to the repeatable queries before applying them. We recommend that you incorporate such tests for enhanced reliability.
- To illustrate the solution, this pattern provides a sample `redshift.tf` file that uses a local Terraform state file. However, for production environments, we strongly recommend that you employ a remote state file with a locking mechanism for enhanced stability and collaboration.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

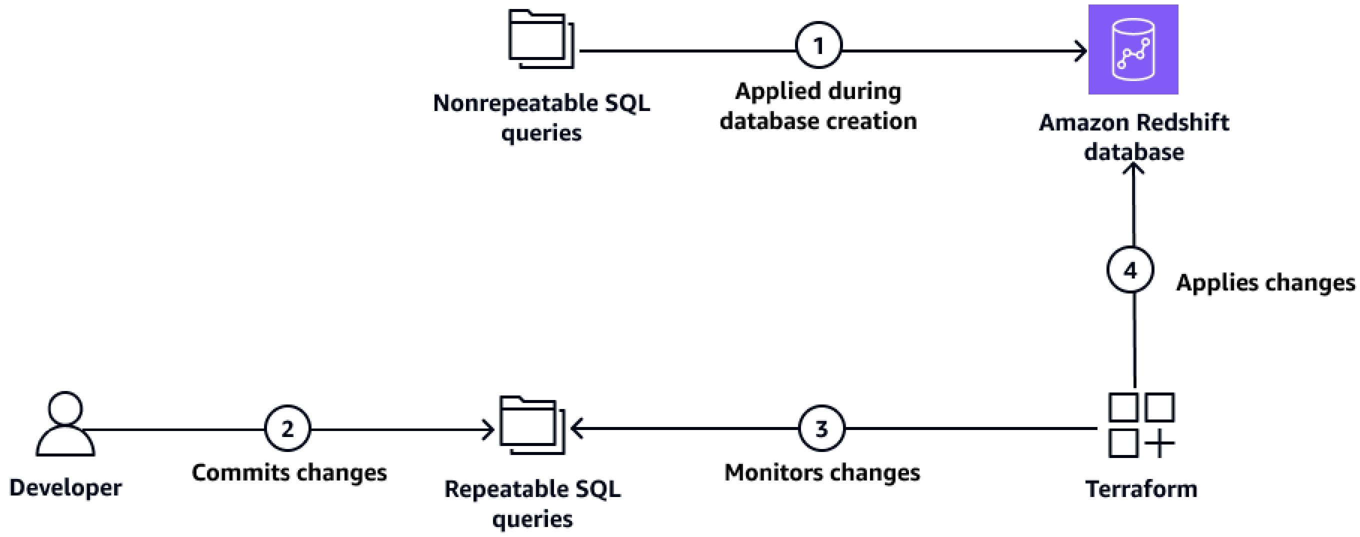
This solution is developed and tested on Amazon Redshift patch 179.

Code repository

The code for this pattern is available in the GitHub amazon-redshift-sql-deploy-terraform repository.

Architecture

The following diagram illustrates how Terraform manages the Amazon Redshift database resources by handling both nonrepeatable and repeatable SQL queries.



The diagram shows the following steps:

1. Terraform applies nonrepeatable SQL queries during the initial Amazon Redshift cluster deployment.
2. The developer commits changes to the repeatable SQL queries.
3. Terraform monitors changes in the repeatable SQL queries.
4. Terraform applies repeatable SQL queries to the Amazon Redshift database.

The solution provided by this pattern is built based on the Terraform module for Amazon Redshift. The Terraform module provisions an Amazon Redshift cluster and database. To enhance the module, we used `terraform_data` resources, which invokes a custom Python script to execute SQL queries using the Amazon Redshift ExecuteStatement API operation. As a result, the module can do the following:

- Deploy any number of database resources by using SQL queries after the database is provisioned.
- Monitor continuously for changes in the repeatable SQL queries and apply those changes using Terraform.

For more details, see *Solution walkthrough* in Additional information.

Tools

AWS services

- Amazon Redshift is a fully managed petabyte-scale data warehouse service in the AWS Cloud.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.
- Python is a general-purpose programming language that's used in this pattern to execute SQL queries.

Best practices

- Amazon Redshift best practices
- Using the Amazon Redshift Data API to interact with Amazon Redshift clusters

Epics

Deploy the solution using Terraform

Task	Description	Skills required
Clone the repository.	To clone the Git repository containing the Terraform code for provisioning an Amazon Redshift cluster, use the following command. `git clone https://github.com/aws-samples/amazon-redshift-sql-deploy-terraform.git`	DevOps engineer
Update the Terraform variables.	To customize the Amazon Redshift cluster deployment according to your specific requirements, update the	DevOps engineer

Task	Description	Skills required

	following parameters in the `terraform.tfvars` file.
	``
	region
	  = "<AWS_REGION>"
	cluster_identifier
	        = "<REDSHIF
	T_CLUSTER_IDENTIFI
	ER>"
	node_type
	   = "<REDSHIFT_NODE_TY
	PE>"
	number_of_nodes
	         = "<REDSHIF
	T_NODE_COUNT>"
	database_name
	         = "<REDSHIF
	T_DB_NAME>"
	subnet_ids
	   = "<REDSHIFT_SUBNET_
	IDS>"
	vpc_security_group_id
	s = "<REDSHIF
	T_SECURITY_GROUP_I
	DS>"
	run_nonrepeatable_quer
	ies = true
	run_repeatable_queries
	    = true
	sql_path_bootstrap
	        = "<BOOTSTR
	AP_SQLS_PATH>"
	sql_path_nonrepeatabl
	e = "<NON-REP
	EATABLE_SQLS_PATH>"
	sql_path_repeatable
	       = "<REPEATA
	BLE_SQLS_PATH>"
	sql_path_finalize
	         = "<FINALIZ
	E_SQLS_PATH>"
	``	

Task	Description	Skills required

	<pre>create_random_password
	    = false
	master_username
	         = "<REDSHIF
	T_MASTER_USERNAME>"</pre>	

Deploy the resources using Terraform.	1. To prepare for the deployment process, use the following command to initialize Terraform within the cloned repository.
	<pre>terraform init</pre>2. To preview the changes that Terraform will apply to the infrastructure, use the following command to create an execution plan.
	<pre>terraform plan -var-
	file terraform.tfvars</pre>3. To provision the Amazon Redshift cluster and associated resources, use the following command to apply the Terraform execution plan.
	<pre>terraform apply -var-
	file terraform.tfvars</pre>
		DevOps engineer

Task	Description	Skills required

(Optional) Execute additional SQL queries.	The sample repository provides several SQL queries for demo purposes. To execute your own SQL queries, add them to the following folders:
	
	`/bootstrap`
	
	`/nonrepeatable`
	
	`/repeatable`
	
	`/finalize`	

Monitor the execution of SQL statements

Task	Description	Skills required
Monitor the deployment of SQL statements.	You can monitor the results of the SQL executions to an Amazon Redshift cluster. For examples of output that show a failed and a successfu l SQL execution, see *Example SQL statements* in Additional information.	DBA, DevOps engineer
Clean up resources.	To delete all the resources deployed by Terraform, run the following command. `terraform destroy`	DevOps engineer

Validate the results

Task	Description	Skills required
Validate the data in the Amazon Redshift cluster.	1. Sign in to the AWS Management Console, and open the Amazon Redshift console. 2. On the navigation menu, choose **Clusters**. Choose the relevant cluster name in the list. 3. Follow the instructions in [Querying a database using the Amazon Redshift query editor v2](#) in the Amazon Redshift documentation.	DBA, AWS DevOps

Related resources

AWS documentation

- Amazon Redshift provisioned clusters
- Troubleshooting issues for Amazon Redshift Data API

Other resources

- Command: apply (Terraform documentation)

Additional information

Solution walkthrough

To use the solution, you must organize your Amazon Redshift SQL queries in a specific way. All SQL queries must be stored in files with a `.sql` extension.

In the code example provided with this pattern, the SQL queries are organized in the following folder structure. You can modify the code (`sql-queries.tf` and `sql-queries.py`) to work with any structure that fits your unique use case.

```
/bootstrap
     |- Any # of files
     |- Any # of sub-folders
/nonrepeatable
     |- Any # of files
     |- Any # of sub-folders
/repeatable
     /udf
          |- Any # of files
          |- Any # of sub-folders
     /table
          |- Any # of files
          |- Any # of sub-folders
     /view
          |- Any # of files
          |- Any # of sub-folders
     /stored-procedure
          |- Any # of files
          |- Any # of sub-folders
/finalize
     |- Any # of files
     |- Any # of sub-folders
```

Given the preceding folder structure, during Amazon Redshift cluster deployment, Terraform executes the queries in the following order:

1. `/bootstrap`
2. `/nonrepeatable`
3. `/repeatable`
4. `/finalize`

The `/repeatable` folder contains four subfolders: `/udf`, `/table`, `/view`, and `/stored-procedure`. These subfolders indicate the order in which Terraform executes the SQL queries.

The Python script that executes the SQL queries is `sql-queries.py`. First, the script reads all the files and subfolders of a specific source directory, for example, the `sql_path_bootstrap` parameter. Then the script executes the queries by calling the Amazon Redshift ExecuteStatement API operation. You might have one or more SQL queries in a file. The following code snippet shows the Python function that executes SQL statements stored in a file against an Amazon Redshift cluster.

```
def execute_sql_statement(filename, cluster_id, db_name, secret_arn, aws_region):
    """Execute SQL statements in a file"""
    redshift_client = boto3.client(
        'redshift-data', region_name=aws_region)
    contents = get_contents_from_file(filename),
    response = redshift_client.execute_statement(
        Sql=contents[0],
        ClusterIdentifier=cluster_id,
        Database=db_name,
        WithEvent=True,
        StatementName=filename,
        SecretArn=secret_arn
    )
    ...
```

The Terraform script `sql-queries.tf` creates the terraform_data resources that invoke the `sql-queries.py` script. There is a `terraform_data` resource for each of the four folders: `/bootstrap`, `/nonrepeatable`, `/repeatable`, and `/finalize`. The following code snippet shows the `terraform_data` resource that execute the SQL queries in the `/bootstrap` folder.

```
locals {
  program               = "${path.module}/sql-queries.py"
  redshift_cluster_name = try(aws_redshift_cluster.this[0].id, null)
}

resource "terraform_data" "run_bootstrap_queries" {
  count      = var.create && var.run_nonrepeatable_queries && (var.sql_path_bootstrap !
= "") && (var.snapshot_identifier == null) ? 1 : 0
  depends_on = [aws_redshift_cluster.this[0]]

  provisioner "local-exec" {
    command = "python3 ${local.program} ${var.sql_path_bootstrap}
 ${local.redshift_cluster_name} ${var.database_name} ${var.redshift_secret_arn}
 ${local.aws_region}"
  }
}
```

You can control whether to run these queries by using the following variables. If you don't want to run queries in `sql_path_bootstrap`, `sql_path_nonrepeatable`, `sql_path_repeatable`, or `sql_path_finalize`, set their values to `""`.

```
run_nonrepeatable_queries = true
run_repeatable_queries    = true
sql_path_bootstrap        = "src/redshift/bootstrap"
sql_path_nonrepeatable    = "src/redshift/nonrepeatable"
sql_path_repeatable       = "src/redshift/repeatable"
sql_path_finalize         = "src/redshift/finalize"
```

When you run `terraform apply`, Terraform considers the `terraform_data` resource added after the script is completed, regardless of the results of the script. If some SQL queries failed, and you want to rerun them, you can manually remove the resource from the Terraform state, and run `terraform apply` again. For example, the following command removes the `run_bootstrap_queries` resource from the Terraform state.

```
terraform state rm module.redshift.terraform_data.run_bootstrap_queries[0]
```

The following code example shows how the `run_repeatable_queries` resource monitors changes in the `repeatable` folder by using the [sha256 hash](). If any file within the folder is updated, Terraform marks the entire directory for an update. Then, Terraform runs the queries in the directory again during the next `terraform apply`.

```
resource "terraform_data" "run_repeatable_queries" {
  count      = var.create_redshift && var.run_repeatable_queries &&
 (var.sql_path_repeatable != "") ? 1 : 0
  depends_on = [terraform_data.run_nonrepeatable_queries]

  # Continuously monitor and apply changes in the repeatable folder
  triggers_replace = {
    dir_sha256 = sha256(join("", [for f in fileset("${var.sql_path_repeatable}",
 "**") : filesha256("${var.sql_path_repeatable}/${f}")]))
  }

  provisioner "local-exec" {
    command = "python3 ${local.sql_queries} ${var.sql_path_repeatable}
 ${local.redshift_cluster_name} ${var.database_name} ${var.redshift_secret_arn}"
  }
}
```

To refine the code, you can implement a mechanism to detect and apply changes only to the files that have been updated within the `repeatable` folder, rather than applying changes to all files indiscriminately.

Example SQL statements

The following output shows a failed SQL execution, along with an error message.

```
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): Executing:
 ["/bin/sh" "-c" "python3 modules/redshift/sql-queries.py src/redshift/nonrepeatable
 testcluster-1 db1 arn:aws:secretsmanager:us-east-1:XXXXXXXXXXXX:secret:/redshift/
master_user/password-8RapGH us-east-1"]
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec):
 -------------------------------------------------------------------
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): src/redshift/
nonrepeatable/table/admin/admin.application_family.sql
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec):
 -------------------------------------------------------------------
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): Status:
 FAILED
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): SQL execution
 failed.
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec): Error
 message: ERROR: syntax error at or near ")"
module.redshift.terraform_data.run_nonrepeatable_queries[0] (local-exec):   Position:
 244
module.redshift.terraform_data.run_nonrepeatable_queries[0]: Creation complete after 3s
 [id=ee50ba6c-11ae-5b64-7e2f-86fd8caa8b76]
```

The following output shows a successful SQL execution.

```
module.redshift.terraform_data.run_bootstrap_queries[0]: Provisioning with 'local-
exec'...
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): Executing:
 ["/bin/sh" "-c" "python3 modules/redshift/sql-queries.py src/redshift/bootstrap
 testcluster-1 db1 arn:aws:secretsmanager:us-east-1:XXXXXXXXXXXX:secret:/redshift/
master_user/password-8RapGH us-east-1"]
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec):
 -------------------------------------------------------------------
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): src/redshift/
bootstrap/db.sql
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec):
 -------------------------------------------------------------------
```

```
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): Status: FINISHED
module.redshift.terraform_data.run_bootstrap_queries[0] (local-exec): SQL execution
 successful.
module.redshift.terraform_data.run_bootstrap_queries[0]: Creation complete after 2s
 [id=d565ef6d-be86-8afd-8e90-111e5ea4a1be]
```

Export AWS Backup reports from across an organization in AWS Organizations as a CSV file

Aromal Raj Jayarajan and Purushotham G K, Amazon Web Services

Summary

This pattern shows how to export AWS Backup job reports from across an organization in AWS Organizations as a CSV file. The solution uses AWS Lambda and Amazon EventBridge to categorize AWS Backup job reports based on their status, which can help when configuring status-based automations.

AWS Backup helps organizations centrally manage and automate data protection across AWS services, in the cloud, and on premises. However, for AWS Backup jobs configured within AWS Organizations, consolidated reporting is available only in the AWS Management Console of each organization's management account. Bringing this reporting outside of the management account can reduce the effort required for auditing and increase the scope for automations, notifications, and alerting.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An active organization in AWS Organizations that includes at least a management account and a member account
- AWS Backup configured at the organization level in AWS Organizations (for more information, see Automate centralized backup at scale across AWS services using AWS Backup on the AWS Blog)
- Git, installed and configured on your local machine

Limitations

The solution provided in this pattern identifies AWS resources that are configured for AWS Backup jobs only. The report can't identify AWS resources that aren't configured for backup through AWS Backup.

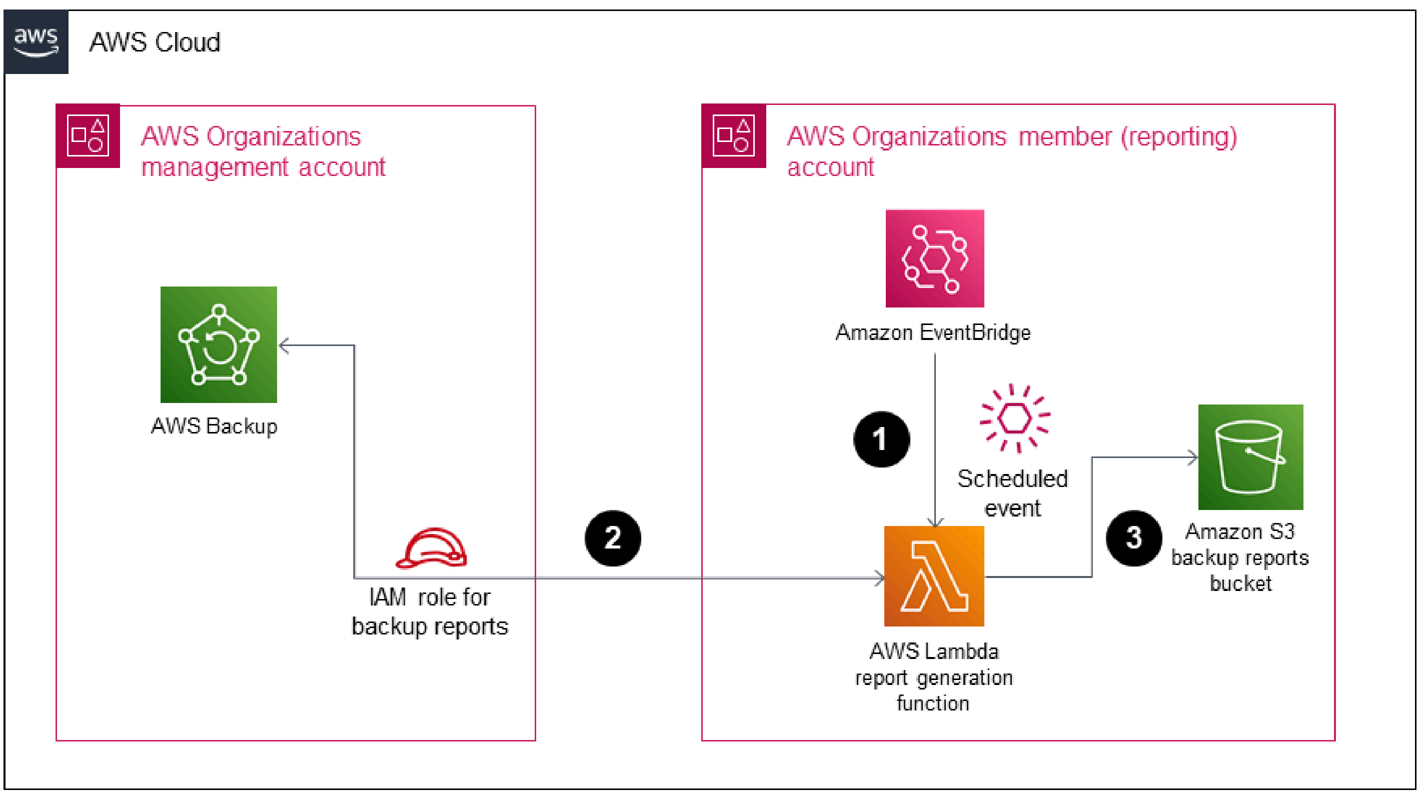
Architecture

Target technology stack

- AWS Backup
- AWS CloudFormation
- Amazon EventBridge
- AWS Lambda
- AWS Security Token Service (AWS STS)
- Amazon Simple Storage Service (Amazon S3)
- AWS Identity and Access Management (IAM)

Target architecture

The following diagram shows an example workflow for exporting AWS Backup job reports from across an organization in AWS Organizations as a CSV file.



The diagram shows the following workflow:

1. A scheduled EventBridge event rule invokes a Lambda function in the member (reporting) AWS account.
2. The Lambda function then uses AWS STS to assume an IAM role that has the permissions required to connect to the management account.
3. The Lambda function then does the following:
   - Requests the consolidated AWS Backup jobs report from the AWS Backup service
   - Categorizes the results based on AWS Backup job status
   - Converts the response to a CSV file
   - Uploads the results to an Amazon S3 bucket in the reporting account within folders that are labeled based on their creation date

Tools

Tools

- AWS Backup is a fully managed service that helps you centralize and automate data protection across AWS services, in the cloud, and on premises.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code

The code for this pattern is available in the GitHub aws-backup-report-generator repository.

Best practices

- Security best practices for Amazon S3 (*Amazon S3 User Guide*)
- Best practices for working with AWS Lambda functions (*AWS Lambda Developer Guide*)
- Best practices for the management account (*AWS Organizations User Guide*)

Epics

Deploy the solution components

Task	Description	Skills required
Clone the GitHub repository.	Clone the GitHub aws-backup-report-generator repository by running the following command in a terminal window: `git clone https://github.com/aws-samples/aws-backup-report-generator.git` For more information, see Cloning a repository in the GitHub Docs.	AWS DevOps, DevOps engineer
Deploy the solution components in the member (reporting) AWS account.	1. In the member (reporting) account, sign in to the AWS Management Console and then open the CloudFormation console. 2. Choose **Create stack**, and then choose **With new resources (standard)**. 3. On the **Create stack** page, in the **Specify template**	DevOps engineer, AWS DevOps

Task	Description	Skills required
	section, choose **Upload a template file**. 4. Select **Choose file**. Then, navigate to the root folder of the cloned GitHub repository on your local workstation and choose **template-reporting.yaml**. 5. Choose **Open**, and then choose **Next**. 6. On the **Specify stack details** page, for **Stack name**, enter a name for your CloudFormation stack. 7. For **ManagementAccountID**, enter the AWS account ID for your organization's management account in AWS Organizations. 8. Choose **Next**. 9. On the **Configure Stack Options** page, choose **Next**. 10. On the **Review** page, select the check box to acknowledge that you reviewed the configuration. 11. Choose **Create stack**. The stack shows **CREATE_COMPLETE** status when the solution components are deployed in the member (reporting) account.	

Test the solution

Task	Description	Skills required
Make sure that the EventBridge rule runs prior to testing.	Make sure that the EventBridge rule runs by waiting at least 24 hours, or by increasing the report frequency in the CloudFormation template's **template-reporting.yml** file. **To increase the report frequency** 1. Open the **template-reporting.yml** file in the cloned repository. 2. In the event rule with the logical ID **'LambdaSchedule'**, find the **'Schedule Expression'**. 3. Edit the **'Schedule Expression'** key so that it includes a valid cron expression. For example, the following cron expression schedules the event rule to run every five minutes: `"cron (* /5 * * * *)"`	AWS DevOps, DevOps engineer
Check the Amazon S3 bucket for the generated report.	1. In the member (reporting) account, sign in to the AWS Management Console and then open the [CloudFormation console]().	AWS DevOps, DevOps engineer

Task	Description	Skills required
	2. In the **Stacks** pane, select the name of the stack you created. Then, choose the **Resources** tab. 3. In the **Resources** pane, in the **Logical ID** column, find **BackupReportS3Bucket**. Then, open the associated Amazon S3 bucket in a new tab by selecting the link in the **Physical ID** column next to that logical ID. 4. Make sure that the bucket contains a report that's generated in the following format: **BackupReports/<yyyy>/<mm>/<dd>/BackupReport-<BACKUP JOB STATUS>-<dd>-<Mon>-<yyyy>.csv**	

Clean up your resources

Task	Description	Skills required
Delete the solution components from the member (reporting) account.	1. In the member (reporting) account, open the solution's Amazon S3 bucket. For instructions, see steps 2-4 in the **Check the S3 bucket for the generated report** story of the **Test the solution** section of this pattern.	AWS DevOps, DevOps engineer

Task	Description	Skills required
	2. Delete the contents of the bucket and empty the bucket. For instructions, see [Emptying a bucket](#) in the *Amazon S3 User Guide*. 3. In the member (reporting) account, sign in to the AWS Management Console and then open the [CloudFormation console](#). 4. In the **Stacks** pane, select the checkbox next to the name of the stack you created. Then, choose **Delete**.	
Delete the solution components from the management account.	1. In the management account, sign in to the AWS Management Console and then open the [CloudFormation console](#). 2. In the **Stacks** pane, select the checkbox next to the name of the stack you created. Then, choose **Delete**.	AWS DevOps, DevOps engineer

Related resources

- [Tutorial: Using AWS Lambda with scheduled events](#) (AWS Lambda documentation)
- [Creating scheduled events to run AWS Lambda functions](#) (AWS SDK for JavaScript documentation)
- [IAM tutorial: Delegate access across AWS accounts using IAM roles](#) (IAM documentation)

- AWS Organizations terminology and concepts (AWS Organizations documentation)
- Creating report plans using the AWS Backup console (AWS Backup documentation)
- Create an audit report (AWS Backup documentation)
- Creating on-demand reports (AWS Backup documentation)
- What is AWS Backup? (AWS Backup documentation)
- Automate centralized backup at scale across AWS services using AWS Backup (AWS blog post)

Export tags for a list of Amazon EC2 instances to a CSV file

Sida Ju and Pac Joonhyun, Amazon Web Services

Summary

This pattern shows how to programmatically export tags for a list of Amazon Elastic Compute Cloud (Amazon EC2) instances to a CSV file.

By using the example Python script provided, you can reduce how long it takes to review and categorize your Amazon EC2 instances by specific tags. For example, you could use the script to quickly identify and categorize a list of instances that your security team has flagged for software updates.

Prerequisites and limitations

Prerequisites

- Python 3 installed and configured
- AWS Command Line Interface (AWS CLI) installed and configured

Limitations

The example Python script provided in this pattern can search Amazon EC2 instances based on the following attributes only:

- Instance IDs
- Private IPv4 addresses
- Public IPv4 addresses

Tools

- Python is a general-purpose computer programming language.
- virtualenv helps you create isolated Python environments.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.

Code repository

The example Python script for this pattern is available in the GitHub search-ec2-instances-export-tags repository.

Epics

Install and configure the prerequisites

Task	Description	Skills required
Clone the GitHub repository.	**Note:** If you receive errors when running AWS CLI commands, make sure that you're using the most recent AWS CLI version. Clone the GitHub search-ec2-instances-export-tags repository by running the following Git command in a terminal window: `git clone https://github.com/aws-samples/search-ec2-instances-export-tags.git`	DevOps engineer
Install and activate virtualenv.	1. Install virtualenv by running the following command: `python3 -m pip install virtualenv`	DevOps engineer

Task	Description	Skills required

	2. Create a new virtual environment by running the following command:
	`python3 -m venv env`
	3. Activate the new virtual environment by running the following command:
	`source env/bin/activate`
	
	For more information, see the [virtualenv User Guide]().	

Install dependencies.	1. Open the code directory by running the following command in the terminal:
	`cd search-ec2-instances-export-tags`
	2. Install the `requirements.txt` file by running the following pip command:
	`pip3 install -r requirements.txt`
		DevOps engineer

Task	Description	Skills required
Configure an AWS named profile.	If you haven't already, configure an AWS named profile that includes the required credentials to run the script. To create a named profile, run the aws configure command. For more information, see Using named profiles in the AWS CLI documentation.	DevOps engineer

Configure and run the Python script

Task	Description	Skills required
Create the input file.	Create an input file that contains a list of the Amazon EC2 instances that you want the script to search and export tags for. You can list instance IDs, private IPv4 addresses, or public IPv4 addresses. **Important** Make sure that each Amazon EC2 instance is listed on its own line in the input file. **Input file example**	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

```
i-0547c351bdfe85b9f
54.157.194.156
172.31.85.33
54.165.198.144
i-0b6223b5914111a4b
172.31.85.44
54.165.198.145
172.31.80.219
172.31.94.199
```

</td><td></td></tr>
</table>

Task	Description	Skills required
Run the Python script.	Run the script by running the following command in the terminal: `python search_instances.py -i INPUTFILE -o OUTPUTFILE -r REGION [-p PROFILE]` **Note** Replace `INPUTFILE` with the name of your input file. Replace `OUTPUTFILE` with the name you want to give the CSV output file. Replace `REGION` with the AWS Region that your Amazon EC2 resources are in. If you're using an AWS named profile, replace `PROFILE` with the named profile that you're using. To get a list of supported parameters and their description, run the following command:	DevOps engineer

Task	Description	Skills required
	`python search_instances.py -h` For more information and to see an output file example, see the `README.md` file in the GitHub search-ec2-instances-export-tags repository.	

Related resources

- Configuring the AWS CLI (*AWS CLI User Guide*)

Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere

Lucas Nation and Freddie Wilson, Amazon Web Services

Summary

Many organizations use AWS Config managed rules to evaluate the compliance of their Amazon Web Services (AWS) resources against common best practices. However, these rules can be time consuming to maintain and this pattern helps you leverage Troposphere, a Python library, to generate and manage AWS Config managed rules.

The pattern helps you to manage your AWS Config managed rules by using a Python script to convert a Microsoft Excel spreadsheet containing AWS managed rules into an AWS CloudFormation template. Troposphere acts as the infrastructure as code (IaC) and this means that you can update the Excel spreadsheet with managed rules, instead of using a JSON or YAML-formatted file. You then use the template to launch an AWS CloudFormation stack that creates and updates the managed rules in your AWS account.

The AWS CloudFormation template defines each AWS Config managed rule by using the Excel spreadsheet and helps you to avoid manually creating individual rules in the AWS Management Console. The script defaults each managed rule's parameters to an empty dictionary and the scope's `ComplianceResourceTypes` defaults from `THE_RULE_IDENTIFIER.template file`. For more information about the rule identifier, see Creating AWS Config managed rules with AWS CloudFormation templates in the AWS Config documentation.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Familiarity with using AWS CloudFormation templates to create AWS Config managed rules. For more information about this, see Creating AWS Config managed rules with AWS CloudFormation templates in the AWS Config documentation.
- Python 3, installed and configured. For more information about this, see the Python documentation.
- An existing integrated development environment (IDE).

- Identify your organizational units (OUs) in a column in the sample `excel_config_rules.xlsx` Excel spreadsheet (attached).

Epics

Customize and configure the AWS Config managed rules

Task	Description	Skills required
Update the sample Excel spreadsheet.	Download the sample `excel_config_rules.xlsx` Excel spreadsheet (attached) and label as `Implemented` the AWS Config managed rules that you want to use. Rules marked as `Implemented` will be added to the AWS CloudFormation template.	Developer

(Optional) Update the config_rules_params.json file with AWS Config rule parameters.	Some AWS Config managed rules require parameters and should be passed to the Python script as a JSON file by using the `--param-file` option. For example, the `access-keys-rotated` managed rule uses the following `maxAccessKeyAge` parameter:
	
	`{`
	`"access-keys-rotated": {`
	`"InputParameters": {`
		Developer

Task	Description	Skills required

	``
	"maxAccessKeyAge": 90
	            }
	        }
	    }
	``
	In this sample parameter, the `maxAccessKeyAge` is set to 90 days. The script reads the parameter file and adds any `InputParameters` that it finds.	

Task	Description	Skills required
(Optional) Update the config_rules_params.json file with AWS Config ComplianceResourceTypes.	By default, the Python script retrieves the `ComplianceResourceTypes` from AWS defined templates. If you want to override the scope of a specific AWS Config managed rule, then you need to pass it to the Python script as a JSON file using the `--param-file` option. For example, the following sample code shows how the `ComplianceResourceTypes` for `ec2-volume-inuse-check` is set to the `["AWS::EC2::Volume"]` list: `{ "ec2-volume-inuse-check": { "Scope": { "ComplianceResourceTypes": [ "AWS::EC2::Volume" ] } } }`	Developer

Run the Python script

Task	Description	Skills required
Install the pip packages from the requirements.txt file.	Download the `requirements.txt` file (attached) and run the following command in your IDE to install the Python packages: `pip3 install -r requirements.txt`	Developer
Run the Python script.	1. Download the `aws_config_rules.py` file (attached) to your local machine. 2. **Note** Run the - `python3 aws_config_rules.py --ou <OU_NAME>` command. : --ou defines which OU column to choose in the Excel spreadsheet. You can also add the following optional parameters: • `--config-rule-option` – Defines the rules	Developer

Task	Description	Skills required
	to choose from the Excel spreadsheet. The default is the `Implemented` parameter. • `--excel-file` – The path for the Excel spreadsheet. The default is `aws_config_rules.xlsx`. • `--param-file` – The path of the parameter JSON file. The default is `config_rules_params.json`. • `--max-execution-frequency` – Defines how often the AWS Config managed rules are evaluated. The choices are `One_Hour`, `Three_Hours`, `Six_Hours`, `Twelve_Hours`, or `TwentyFour_Hours`. The default is `TwentyFour_Hours`.	

Deploy the AWS Config managed rules

Task	Description	Skills required
Launch the AWS CloudFormation stack.	1. Sign in to the AWS Management Console, open the AWS CloudForm	Developer

Task	Description	Skills required
	ation console, and then choose **Create stack.** 2. On the **Specify template** page, choose **Upload a template file** and then upload your AWS CloudFormation template. 3. Specify a stack name and then choose **Next**. 4. Specify tags and then choose **Next**. 5. Choose **Create stack**.	

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Give SageMaker notebook instances temporary access to a CodeCommit repository in another AWS account

Helge Aufderheide, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. [Learn more]

This pattern shows how to grant Amazon SageMaker notebook instances and users temporary access to an AWS CodeCommit repository that's in another AWS account. This pattern also shows how you can grant granular permissions for specific actions each entity can perform on each repository.

Organizations often store CodeCommit repositories in a different AWS account than the account that hosts their development environment. This multi-account setup helps control access to the repositories and reduces the risk of their accidental deletion. To grant these cross-account permissions, it's a best practice to use AWS Identity and Access Management (IAM) roles. Then, predefined IAM identities in each AWS account can temporarily assume the roles to create a controlled chain of trust across the accounts.

> **Note**
>
> You can apply a similar procedure to grant other IAM identities cross-account access to a CodeCommit repository. For more information, see [Configure cross-account access to an AWS CodeCommit repository using roles] in the *AWS CodeCommit User Guide*.

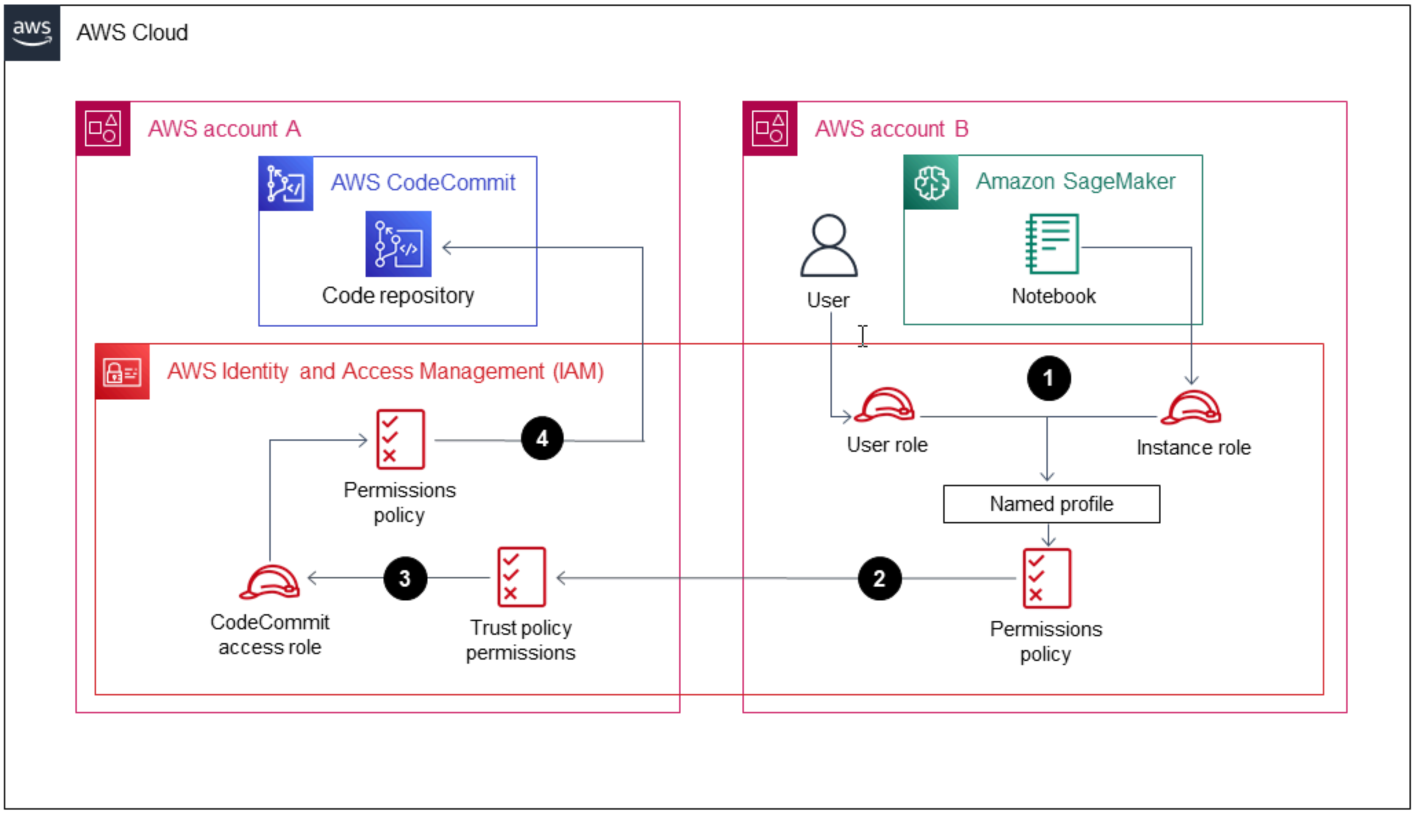
Prerequisites and limitations

Prerequisites

- An active AWS account with a CodeCommit repository (*account A*)
- A second active AWS account with a SageMaker notebook instance (*account B*)
- An AWS user with sufficient permissions to create and modify IAM roles in account A
- A second AWS user with sufficient permissions to create and modify IAM roles in account B

Architecture

The following diagram shows an example workflow for granting a SageMaker notebook instance and users in one AWS account cross-account access to a CodeCommit repository:



The diagram shows the following workflow:

1. The AWS user role and SageMaker notebook instance role in account B assume a [named profile](#).
2. The named profile's permissions policy specifies a CodeCommit access role in account A that the profile then assumes.
3. The CodeCommit access role's trust policy in account A allows the named profile in account B to assume the CodeCommit access role.
4. The CodeCommit repository's IAM permissions policy in account A allows the CodeCommit access role to access the CodeCommit repository.

Technology stack

- CodeCommit
- Git

- IAM
- pip
- SageMaker

Tools

- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Git is a distributed version-control system for tracking changes in source code during software development.
- git-remote-codecommit is a utility that helps you push and pull code from CodeCommit repositories by extending Git.
- pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes.

Best practices

When you set permissions with IAM policies, make sure that you grant only the permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

When implementing this pattern, make sure that you do the following:

- Confirm that IAM principles have only the permissions required to perform specific, needed actions within each repository. For example, it's recommended to allow approved IAM principles to push and merge changes to specific repository branches, but only request merges to protected branches.
- Confirm that IAM principles are assigned different IAM roles based on their respective roles and responsibilities for each project. For example, a developer will have different access permissions than a release manager or AWS Administrator.

Epics

Configure the IAM roles

Task	Description	Skills required
Configure the CodeCommit access role and permissions policy.	**Note** To automate the manual setup process documented in this epic, you can use an AWS CloudFormation template. In the account that contains the CodeCommit repository (*account A*), do the following: 1. Create an IAM role that can be assumed by the SageMaker notebook instance role in account B. 2. Create an IAM policy that grants access to the repository and attach the policy to the role. For testing purposes only, choose the AWSCodeCommitPowerUser AWS managed policy. This policy grants all CodeCommit permissions except the ability to delete resources.	General AWS, AWS DevOps

Task	Description	Skills required
	3. Modify the role's trust policy so that account B is listed as a trusted entity. **Important** Before moving this setup into your production environment, it's a best practice to write your own IAM policy that applies least-privilege permissions. For more information, see the **Additional information** section of this pattern.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Grant the SageMaker notebook instance's role in account B permissions to assume the CodeCommit access role in account A.</td><td>In the account that contains the SageMaker notebook instance’s IAM role (account B), do the following:

Create an IAM policy that allows an IAM role or user to assume the CodeCommi t access role in account A.
Example IAM permissions policy that allows an IAM role or user to assume a cross-account role
<pre><code>{
  "Version":
 "2012-10-17",
  "Statement": [
      {
    "Sid": "VisualEd
itor0",
    "Effect": "Allow",
    "Action":
 "sts:AssumeRole",
    "Resource":
 "arn:aws:iam:::acc
ountA_ID:role/acco
untArole_ID"
  }
}</code></pre>
Attach the policy to your SageMaker notebook instance’s role in account B.
Have the SageMaker notebook instance’s role in account B assume the
</td><td>General AWS, AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	CodeCommit access role in account A. **Note** To view your repository's Amazon Resource Name (ARN), see [View CodeCommit repository details](#) in the *AWS CodeCommit User Guide*.	

Set up your SageMaker notebook instance in account B

Task	Description	Skills required
Set up a user profile on the AWS SageMaker notebook instance to assume the role in account A.	**Important** [Make sure that you have the latest version of the AWS Command Line Interface (AWS CLI) installed](#). In the account that contains the SageMaker notebook instance (*account B)*, do the following: 1. Sign in to the AWS Management Console	General AWS, AWS DevOps

Task	Description	Skills required

	and open the SageMaker console.
	2. Access your SageMaker notebook instance. The Jupyter interface opens.
	3. Choose **New**, and then choose **Terminal**. A new terminal window opens in your Jupyter environment.
	4. Navigate to the SageMaker notebook instance's **~/.aws/config** file. Then, add a user profile to the file by entering the following statement:
	`------.aws/config-`
	`-------------`
	`[profile remoterep`
	`ouser]`
	`role_arn = arn:aws:i`
	`am::<ID of Account`
	` A>:role/<rolename>`
	`role_session_name =`
	` remoteaccesssession`
	`region = eu-west-1`
	`credential_source  =`
	` Ec2InstanceMetadata`
	`----------------`
	`------------------`	

Install the git-remote-codecommit utility.	Follow the instructions in Step 2: Install git-remote-codecommit in the *AWS CodeCommit User Guide*.	Data scientist

Access the repository

Task	Description	Skills required
Access the CodeCommit repository by using Git commands or SageMaker.	**To use Git** IAM principals that assume the SageMaker notebook instance's role in account B can now run Git commands to access the CodeCommit repository in account A. For example, users can run commands such as `git clone`, `git pull`, and `git push`. For instructions, see [Connect to an AWS CodeCommit repository](#) in the *AWS CodeCommit User Guide*. For information about how to use Git with CodeCommit, see [Getting started with AWS CodeCommit](#) in the *AWS CodeCommit User Guide*. **To use SageMaker** To use Git from the SageMaker console, you must allow Git to retrieve credentials from your CodeCommit repository. For instructions, see [Associate a CodeCommit repository in a different AWS account with a notebook](#)	Git, bash console

Task	Description	Skills required
	instance in the SageMaker documentation.	

Related resources

- Configure cross-account access to an AWS CodeCommit repository using roles (AWS CodeCommit documentation)
- IAM tutorial: Delegate access across AWS accounts using IAM roles (IAM documentation)

Additional information

Restricting CodeCommit permissions to specific actions

To restrict the actions that an IAM principal can take in the CodeCommit repository, modify the actions that are allowed in the CodeCommit access policy.

For more information about CodeCommit API operations, see CodeCommit permissions reference in the *AWS CodeCommit User Guide*.

> **Note**
>
> You can also edit the AWSCodeCommitPowerUser AWS managed policy to fit your use case.

Restricting CodeCommit permissions to specific repositories

To create a multitenant environment where more than one code repository is accessible to only specific users, do the following:

1. Create multiple CodeCommit access roles in account A. Then, configure each access role's trust policy to allow specific users in account B to assume the role.
2. Restrict what code repositories that each role can assume by adding a **"Resource"** condition to each CodeCommit access role's policy.

Example "Resource" condition that restricts an IAM principal's access to a specific CodeCommit repository

```
"Resource" : [<REPOSITORY_ARN>,<REPOSITORY_ARN> ]
```

Note

To help identify and differentiate multiple code repositories in the same AWS account, you can assign different prefixes to the repositories' names. For example, you can name code repositories with prefixes that align to different developer groups, such as **myproject-subproject1-repo1** and **myproject-subproject2-repo1**. Then, you can create an IAM role for each developer group based on their assigned prefixes. For example, you could create a role named **myproject-subproject1-repoaccess** and grant it access to all of the code repositories that include the prefix **myproject-subproject1**.

Example "Resource" condition that refers to a code repository ARN that includes a specific prefix

```
"Resource" : arn:aws:codecommit:<region>:<account-id>:myproject-subproject1-*
```

Implement a GitHub Flow branching strategy for multi-account DevOps environments

Mike Stephens and Abhilash Vinod, Amazon Web Services

Summary

When managing a source code repository, different branching strategies affect the software development and release processes that development teams use. Examples of common branching strategies include Trunk, GitHub Flow, and Gitflow. These strategies use different branches, and the activities performed in each environment are different. Organizations that are implementing DevOps processes would benefit from a visual guide to help them understand the differences between these branching strategies. Using this visual in your organization helps development teams align their work and follow organizational standards. This pattern provides this visual and describes the process of implementing a GitHub Flow branching strategy in your organization.

This pattern is part of a documentation series about choosing and implementing DevOps branching strategies for organizations with multiple AWS accounts. This series is designed to help you apply the correct strategy and best practices from the outset, to streamline your experience in the cloud. GitHub Flow is just one possible branching strategy that your organization can use. This documentation series also covers Trunk and Gitflow branching models. If you haven't done so already, we recommend that you review Choosing a Git branching strategy for multi-account DevOps environments prior to implementing the guidance in this pattern. Please use due diligence to choose the right branching strategy for your organization.

This guide provides a diagram that shows how an organization might implement the GitHub Flow strategy. It is recommended that you review the AWS Well-Architected DevOps Guidance to review best practices. This pattern includes recommended tasks, steps, and restrictions for each step in the DevOps process.

Prerequisites and limitations

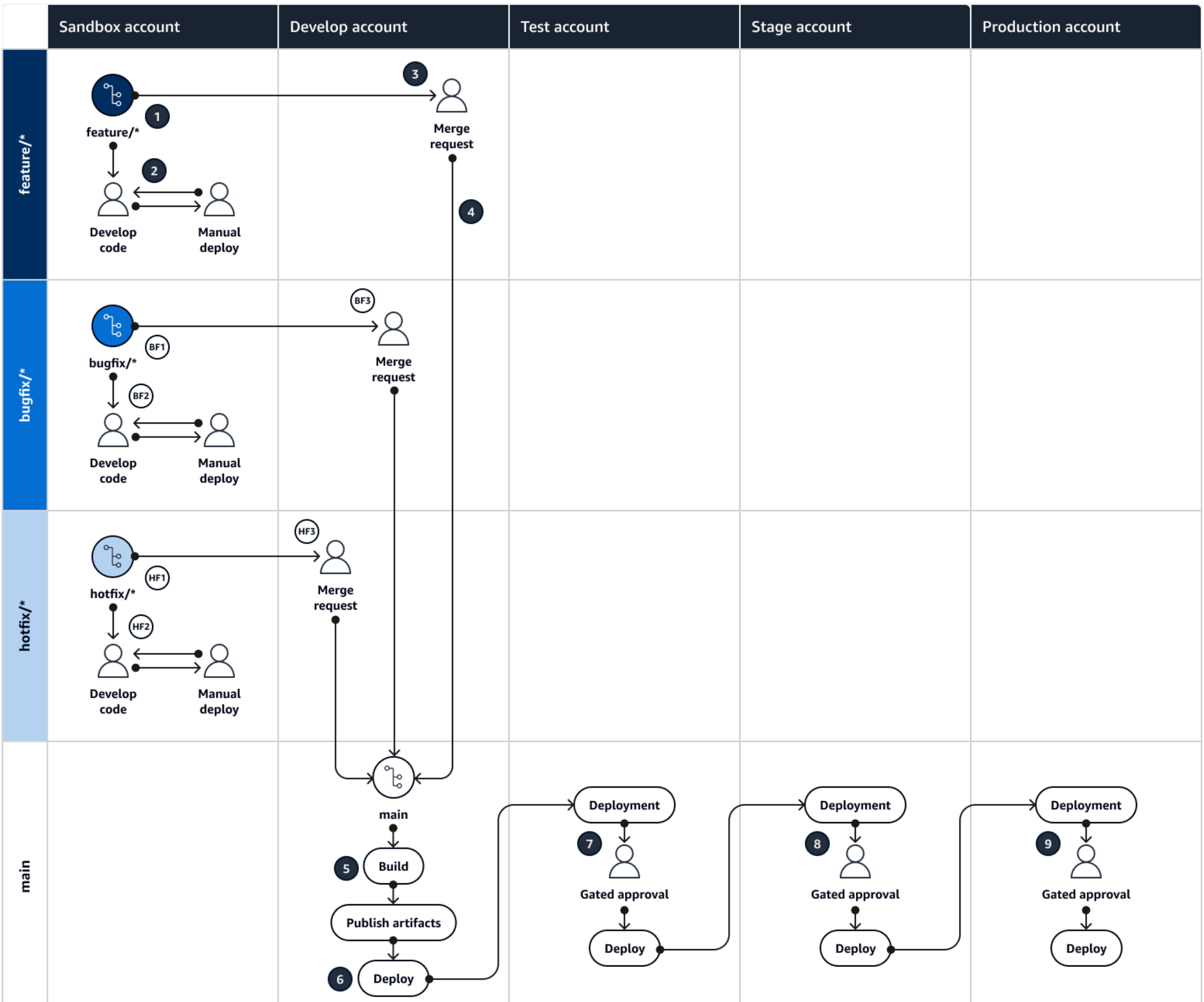
Prerequisites

- Git, installed. This is used as a source code repository tool.
- Draw.io, installed. This application is used to view and edit the diagram.

Architecture

Target architecture

The following diagram can be used like a [Punnett square](https://en.wikipedia.org/wiki/Punnett_square) (Wikipedia). You line up the branches on the vertical axis with the AWS environments on the horizontal axis to determine what actions to perform in each scenario. The numbers indicate the sequence of the actions in the workflow. This example takes you from a `feature` branch through deployment in production.



For more information about the AWS accounts, environments, and branches in a GitHub Flow approach, see [Choosing a Git branching strategy for multi-account DevOps environments](https://docs.aws.amazon.com/prescriptive-guidance/latest/choosing-git-branch-approach/).

Automation and scale

Continuous integration and continuous delivery (CI/CD) is the process of automating the software release lifecycle. It automates much or all of the manual processes traditionally required to get new code from an initial commit into production. A CI/CD pipeline encompasses the sandbox, development, testing, staging, and production environments. In each environment, the CI/CD pipeline provisions any infrastructure that is needed to deploy or test the code. By using CI/CD, development teams can make changes to code that are then automatically tested and deployed. CI/CD pipelines also provide governance and guardrails for development teams by enforcing consistency, standards, best practices, and minimal acceptance levels for feature acceptance and deployment. For more information, see Practicing Continuous Integration and Continuous Delivery on AWS.

AWS offers a suite of developer services that are designed to help you build CI/CD pipelines. For example, AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. AWS CodeBuild compiles source code, runs tests, and produces ready-to-deploy software packages. For more information, see Developer Tools on AWS.

Tools

AWS services and tools

AWS provides a suite of developer services that you can use to implement this pattern:

- AWS CodeArtifact is a highly scalable, managed artifact repository service that helps you store and share software packages for application development.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeDeploy automates deployments to Amazon Elastic Compute Cloud (Amazon EC2) or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.

Other tools

- Draw.io Desktop is an application for making flowcharts and diagrams. The code repository contains templates in .drawio format for Draw.io.
- Figma is an online design tool designed for collaboration. The code repository contains templates in .fig format for Figma.

Code repository

This source file for the diagram in this pattern is available in the GitHub Git Branching Strategy for GitHub Flow repository. It includes files in PNG, draw.io, and Figma formats. You can modify these diagrams to support your organization's processes.

Best practices

Follow the best practices and recommendations in AWS Well-Architected DevOps Guidance and Choosing a Git branching strategy for multi-account DevOps environments. These help you effectively implement GitHub Flow-based development, foster collaboration, improve code quality, and streamline the development process.

Epics

Reviewing the GitHub Flow workflows

Task	Description	Skills required
Review the standard GitHub Flow process.	1. In the sandbox environme nt, the developer creates a `feature` branch from the `main` branch and uses the naming pattern `feature/<ticket>_<initials>_<short description>` . 2. The developer adds one or more commits to the `feature` branch, each representing a discrete change or improvement.	DevOps engineer

Task	Description	Skills required
	3. The developer opens a merge request (MR) to merge the changes into the `main` branch. This initiates a review process. 4. During the review process, developers discuss the code changes and provide feedback. The goal is to ensure that the changes are of high quality and meet the project's standards. 5. After the developer creates the merge request, an automated build process starts and deploys the changes in the `feature` branch to the development environment. 6. Automated tests verify the integrity and quality of the changes encapsulated in the merge request. A successful build, successful deployment, and successful testing are required to complete the merge request. 7. When the review process is complete, the changes are merged into the `main` branch.	

Task	Description	Skills required
	8. An approver manually approves the deployment of the release artifacts to the testing environment. 9. An approver manually approves the deployment of the release artifacts to the staging environment. 10. An approver manually approves the deployment of the release artifacts to the production environme nt.	

Task	Description	Skills required
Review the bugfix GitHub Flow process.	1. The developer creates a `bugfix` branch from the `main` branch and uses the naming pattern `bugfix/<ticket number>_<developer initials>_<descriptor>` . 2. The developer fixes the issue, commits the fix, and builds the `bugfix` branch. 3. The developer opens a merge request to merge the `bugfix` branch into the `main` branch. This initiates a review process. 4. During the review process, developers discuss the code changes and provide feedback. 5. Upon review completion and approval, the developer completes the merge request of the `bugfix` branch into the `main` branch. 6. An approver manually approves the deployment of the release artifacts to higher environments.	DevOps engineer

Task	Description	Skills required
Review the hotfix GitHub Flow process.	GitHub Flow is designed to enable continuous delivery, where code changes are frequently and reliably deployed to higher environme nts. The key is that every `feature` branch is deployabl e at any time. `Hotfix` branches, which are akin to `feature` or `bugfix` branches, can follow the same process as either of these other branches. However, given their urgency, hotfixes typically have a higher priority. Depending on the team's policies and the immediacy of the situation, certain steps in the process could be expedited. For instance, code reviews for hotfixes might be fast-trac ked. Therefore, while the hotfix process parallels the feature or bugfix process, the urgency surrounding hotfixes may warrant modifications in the procedural adherence . It's crucial to establish guidelines about managing hotfixes to make sure that they are handled efficiently and securely.	DevOps engineer

Troubleshooting

Issue	Solution
Branch conflicts	A common issue that can occur with the GitHub Flow model is where a hotfix needs to occur in production but a corresponding change needs to occur in a `feature`, `bugfix`, or `hotfix` branch where the same resources are being modified. We recommend that you frequently merge changes from `main` into lower branches to avoid significant conflicts when you merge to `main`.
Team maturity	GitHub Flow encourages daily deploymen ts to higher environments, embracing true continuous integration and continuous delivery (CI/CD). It is imperative that the team has the engineering maturity to build features and create automation tests for them. The team must perform an exhaustive merge request review before changes are approved. This fosters a robust engineering culture that promotes quality, accountability, and efficienc y in the development process.

Related resources

This guide doesn't include training for Git; however, there are many high-quality resources available on the internet if you need this training. We recommend that you start with the [Git documentation] site.

The following resources can help you with your GitHub Flow branching journey in the AWS Cloud.

AWS DevOps guidance

- AWS DevOps Guidance

- AWS Deployment Pipeline Reference Architecture
- What is DevOps?
- DevOps resources

GitHub Flow guidance

- GitHub Flow Quickstart Tutorial (GitHub)
- Why GitHub Flow?

Other resources

- Twelve-factor app methodology (12factor.net)

Implement a Gitflow branching strategy for multi-account DevOps environments

Mike Stephens, Stephen DiCato, Abhilash Vinod, and Tim Wondergem, Amazon Web Services

Summary

When managing a source code repository, different branching strategies affect the software development and release processes that development teams use. Examples of common branching strategies include Trunk, Gitflow, and GitHub Flow. These strategies use different branches, and the activities performed in each environment are different. Organizations that are implementing DevOps processes would benefit from a visual guide to help them understand the differences between these branching strategies. Using this visual in your organization helps development teams align their work and follow organizational standards. This pattern provides this visual and describes the process of implementing a Gitflow branching strategy in your organization.

This pattern is part of a documentation series about choosing and implementing DevOps branching strategies for organizations with multiple AWS accounts. This series is designed to help you apply the correct strategy and best practices from the outset, to streamline your experience in the cloud. Gitflow is just one possible branching strategy that your organization can use. This documentation series also covers Trunk and GitHub Flow branching models. If you haven't done so already, we recommend that you review Choosing a Git branching strategy for multi-account DevOps environments prior to implementing the guidance in this pattern. Please use due diligence to choose the right branching strategy for your organization.

This guide provides a diagram that shows how an organization might implement the Gitflow strategy. It is recommended that you review the AWS Well-Architected DevOps Guidance to review best practices. This pattern includes recommended tasks, steps, and restrictions for each step in the DevOps process.

Prerequisites and limitations

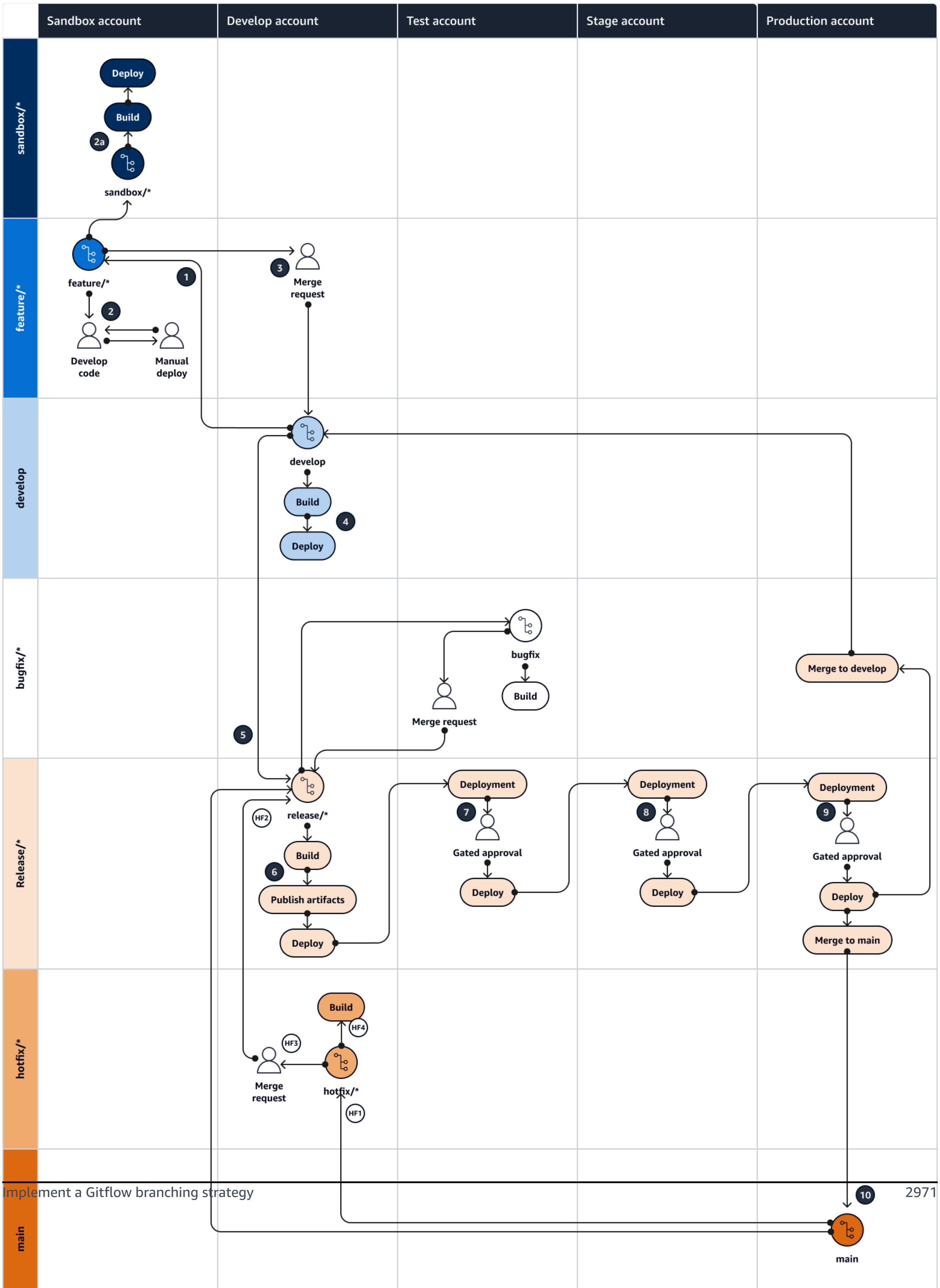
Prerequisites

- Git, installed. This is used as a source code repository tool.
- Draw.io, installed. This application is used to view and edit the diagram.
- (Optional) Gitflow plugin, installed.

Architecture

Target architecture

The following diagram can be used like a Punnett square (Wikipedia). You line up the branches on the vertical axis with the AWS environments on the horizontal axis to determine what actions to perform in each scenario. The numbers indicate the sequence of the actions in the workflow. This example takes you from a feature branch through deployment in production.



Sandbox account
Develop account
Test account
Stage account
Production account
sandbox/*
feature/*
develop
bugfix/*
Release/*
hotfix/*
main
Deploy
Build
2a
sandbox/*
feature/*
1
2
3
Merge request
Develop code
Manual deploy
develop
Build
4
Deploy
bugfix
Build
Merge request
5
Merge to develop
release/*
HF2
Build
6
Publish artifacts
Deploy
Deployment
7
Gated approval
Deploy
Deployment
8
Gated approval
Deploy
Deployment
9
Gated approval
Deploy
Merge to main
Build
HF4
HF3
Merge request
hotfix/*
HF1
10
main

For more information about the AWS accounts, environments, and branches in a Gitflow approach, see Choosing a Git branching strategy for multi-account DevOps environments.

Automation and scale

Continuous integration and continuous delivery (CI/CD) is the process of automating the software release lifecycle. It automates much or all of the manual processes traditionally required to get new code from an initial commit into production. A CI/CD pipeline encompasses the sandbox, development, testing, staging, and production environments. In each environment, the CI/CD pipeline provisions any infrastructure that is needed to deploy or test the code. By using CI/CD, development teams can make changes to code that are then automatically tested and deployed. CI/CD pipelines also provide governance and guardrails for development teams by enforcing consistency, standards, best practices, and minimal acceptance levels for feature acceptance and deployment. For more information, see Practicing Continuous Integration and Continuous Delivery on AWS.

AWS offers a suite of developer services that are designed to help you build CI/CD pipelines. For example, AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. AWS CodeBuild compiles source code, runs tests, and produces ready-to-deploy software packages. For more information, see Developer Tools on AWS.

Tools

AWS services and tools

AWS provides a suite of developer services that you can use to implement this pattern:

- AWS CodeArtifact is a highly scalable, managed artifact repository service that helps you store and share software packages for application development.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeDeploy automates deployments to Amazon Elastic Compute Cloud (Amazon EC2) or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.

Other tools

- Draw.io Desktop is an application for making flowcharts and diagrams. The code repository contains templates in .drawio format for Draw.io.
- Figma is an online design tool designed for collaboration. The code repository contains templates in .fig format for Figma.
- (Optional) Gitflow plugin is a collection of Git extensions that provide high-level repository operations for the Gitflow branching model.

Code repository

This source file for the diagram in this pattern is available in the GitHub Git Branching Strategy for GitFlow repository. It includes files in PNG, draw.io, and Figma formats. You can modify these diagrams to support your organization's processes.

Best practices

Follow the best practices and recommendations in AWS Well-Architected DevOps Guidance and Choosing a Git branching strategy for multi-account DevOps environments. These help you effectively implement Gitflow-based development, foster collaboration, improve code quality, and streamline the development process.

Epics

Reviewing the Gitflow workflows

Task	Description	Skills required
Review the standard Gitflow process.	1. In the sandbox environment, the developer creates a `feature` branch from the `develop` branch and uses the naming pattern `feature/<ticket>_<initials>_<short description>` . 2. The developer develops code and deploys the code to the sandbox environme	DevOps engineer

Task	Description	Skills required
	nt iteratively in order to complete the ticket. **Note** The developer can optionally create a `sandbox` branch to run an automated build or deploy pipeline in the sandbox environme nt. 3. The developer creates a merge request from the `feature` branch into the `develop` branch by using a squash merge. 4. A continuous integration and continuous delivery (CI/CD) pipeline automatic ally builds and deploys the `develop` branch to the development environment. 5. (Optional) A developer integrates additional `feature` branches into the develop branch prior to continuing with release activities. 6. When you are ready to release the features in the `develop` branch, the developer creates a	

Task	Description	Skills required
	`release` branch named `release/v<number>` from the `develop` branch. 7. The developer builds the release branch, which publishes artifacts for reuse across other environments. 8. An approver manually approves the deployment of the release artifacts to the testing environment. 9. An approver manually approves the deployment of the release artifacts to the staging environment. 10. An approver manually approves the deployment of the release artifacts to the production environme nt. 11. The developer merges the `release` branch into the `main` branch. Ideally, the developer uses an automated script to perform a fast-forward merge. Do not use a squash merge. 12. The developer merges the `release` branch into the `develop` branch. Ideally, the developer uses	

Task	Description	Skills required
	an automated script to perform a fast-forward merge. Do not use a squash merge.	

Task	Description	Skills required
Review the hotfix Gitflow process.	1. The developer creates a `hotfix` branch from the `main` branch and uses the naming pattern `hotfix/<ticket>_<initials>_<short description>` . 2. The developer creates a `release` branch from the `main` branch and names it `release/v<number>` . 3. The developer fixes the issue, commits the fix, and builds the `hotfix` branch. 4. The developer creates a merge request from the `hotfix` branch into the `release/v<number>` branch by using a squash merge. 5. The developer builds the `release` branch, which publishes artifacts for reuse across other environments. 6. An approver manually approves the deployment of the release artifacts to the testing environment. 7. An approver manually approves the deployment of the release artifacts to the staging environment.	DevOps engineer

Task	Description	Skills required
	8. An approver manually approves the deployment of the release artifacts to the production environme nt. 9. The developer merges the `release` branch into the `main` branch. Ideally, the developer uses an automated script to perform a fast-forward merge. Do not use a squash merge. 10. The developer merges the `release` branch into the `develop` branch. Ideally, the developer uses an automated script to perform a fast-forward merge. Do not use a squash merge. 11. If a conflict is detected, developers receive an alert and resolve the conflict with a merge request.	

Task	Description	Skills required
Review the bugfix Gitflow process.	1. The developer creates a `bugfix` branch from the current `release/v<number>` branch and uses the naming pattern `bugfix/<ticket number>_<developer initials>_<descriptor>`. 2. The developer fixes the issue, commits the fix, and builds the `bugfix` branch. 3. The developer creates a merge request from the `bugfix` branch into the `release/v<number>` branch by using a squash merge. 4. The developer builds the `release` branch, which publishes artifacts for reuse across other environments. 5. An approver manually approves the deployment of the release artifacts to the Test environment. 6. An approver manually approves the deployment of the release artifacts to the Stage environment. 7. An approver manually approves the deployment	DevOps engineer

Task	Description	Skills required
	of the release artifacts to the Production environme nt. 8. The developer merges the `release` branch into the `main` branch. Ideally, the developer uses an automated script to perform a fast-forward merge. Do not use a squash merge. 9. The developer merges the `release` branch into the `develop` branch. Ideally, the developer uses an automated script to perform a fast-forward merge. Do not use a squash merge. 10.If a conflict is detected, developers receive an alert and resolve the conflict with a merge request.	

Troubleshooting

Issue	Solution
Branch conflicts	A common issue that can occur with the Gitflow model is where a hotfix needs to occur in production but a corresponding change needs to occur in a lower environme nt, where another branch is modifying the

Issue	Solution
	same resources. We recommend that you have only a single release branch active at a time. If you have more than one active at a time, the changes in the environments might collide, and you might be unable to move a branch forward to production.
Merging	Releases should be merged back into main and develop as soon as possible to consolidate work back into the primary branches.
Squash merging	Only use a squash merge when you are merging from a `feature` branch to a `develop` branch. Using squash merges in higher branches causes difficulty when merging changes back down to lower branches.

Related resources

This guide doesn't include training for Git; however, there are many high-quality resources available on the internet if you need this training. We recommend that you start with the Git documentation site.

The following resources can help you with your Gitflow branching journey in the AWS Cloud.

AWS DevOps guidance

- AWS DevOps Guidance
- AWS Deployment Pipeline Reference Architecture
- What is DevOps?
- DevOps resources

Gitflow guidance

- The original Gitflow blog (Vincent Driessen blog post)

- Gitflow workflow (Atlassian)
- Gitflow on GitHub: How to use Git Flow workflows with GitHub Based Repos (YouTube video)
- Git Flow Init Example (YouTube video)
- The Gitflow Release Branch from Start to Finish (YouTube video)

Other resources

Twelve-factor app methodology (12factor.net)

Implement a Trunk branching strategy for multi-account DevOps environments

Mike Stephens and Rayjan Wilson, Amazon Web Services

Summary

When managing a source code repository, different branching strategies affect the software development and release processes that development teams use. Examples of common branching strategies include Trunk, GitHub Flow, and Gitflow. These strategies use different branches, and the activities performed in each environment are different. Organizations that are implementing DevOps processes would benefit from a visual guide to help them understand the differences between these branching strategies. Using this visual in your organization helps development teams align their work and follow organizational standards. This pattern provides this visual and describes the process of implementing a Trunk branching strategy in your organization.

This pattern is part of a documentation series about choosing and implementing DevOps branching strategies for organizations with multiple AWS accounts. This series is designed to help you apply the correct strategy and best practices from the outset, to streamline your experience in the cloud. Trunk is just one possible branching strategy that your organization can use. This documentation series also covers GitHub Flow and Gitflow branching models. If you haven't done so already, we recommend that you review Choosing a Git branching strategy for multi-account DevOps environments prior to implementing the guidance in this pattern. Please use due diligence to choose the right branching strategy for your organization.

This guide provides a diagram that shows how an organization might implement the Trunk strategy. It is recommended that you review the official AWS Well-Architected DevOps Guidance to review best practices. This pattern includes recommended tasks, steps, and restrictions for each step in the DevOps process.

Prerequisites and limitations

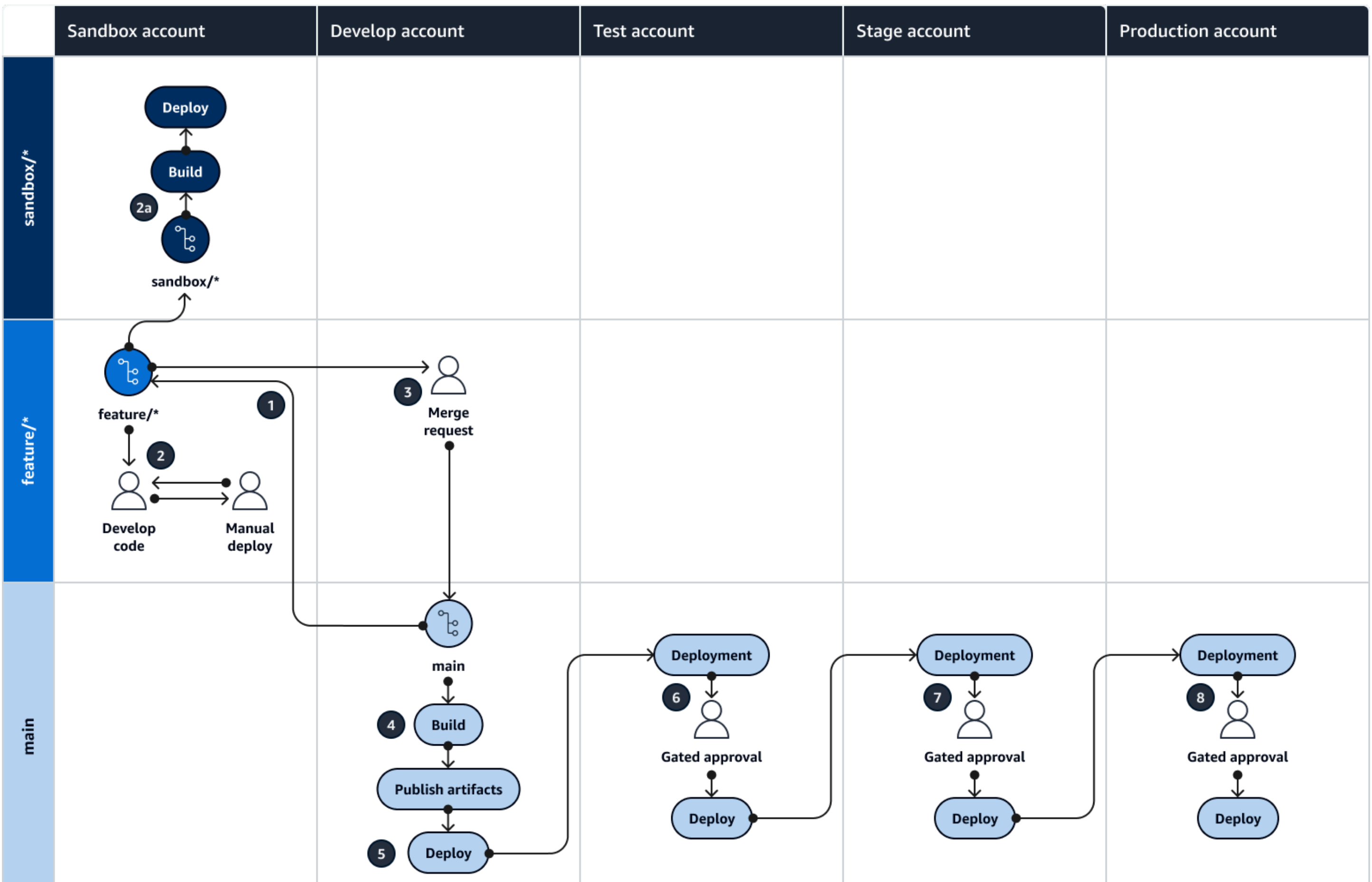
Prerequisites

- Git, installed. This is used as a source code repository tool.
- Draw.io, installed. This application is used to view and edit the diagram.

Architecture

Target architecture

The following diagram can be used like a [Punnett square](https://en.wikipedia.org/wiki/Punnett_square) (Wikipedia). You line up the branches on the vertical axis with the AWS environments on the horizontal axis to determine what actions to perform in each scenario. The numbers indicate the sequence of the actions in the workflow. This example takes you from a `feature` branch through deployment in production.



For more information about the AWS accounts, environments, and branches in a Trunk approach, see [Choosing a Git branching strategy for multi-account DevOps environments](#).

Automation and scale

Continuous integration and continuous delivery (CI/CD) is the process of automating the software release lifecycle. It automates much or all of the manual processes traditionally required to get new code from an initial commit into production. A CI/CD pipeline encompasses the sandbox, development, testing, staging, and production environments. In each environment, the CI/CD pipeline provisions any infrastructure that is needed to deploy or test the code. By using CI/CD, development teams can make changes to code that are then automatically tested and deployed. CI/CD pipelines also provide governance and guardrails for development teams by enforcing consistency, standards, best practices, and minimal acceptance levels for feature acceptance and deployment. For more information, see Practicing Continuous Integration and Continuous Delivery on AWS.

AWS offers a suite of developer services that are designed to help you build CI/CD pipelines. For example, AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. AWS CodeBuild compiles source code, runs tests, and produces ready-to-deploy software packages. For more information, see Developer Tools on AWS.

Tools

AWS services and tools

AWS provides a suite of developer services that you can use to implement this pattern:

- AWS CodeArtifact is a highly scalable, managed artifact repository service that helps you store and share software packages for application development.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeDeploy automates deployments to Amazon Elastic Compute Cloud (Amazon EC2) or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.

Other tools

- Draw.io Desktop – An application for making flowcharts and diagrams.
- Figma is an online design tool designed for collaboration. The code repository contains templates in .fig format for Figma.

Code repository

This source file for the diagram in this pattern is available in the GitHub Git Branching Strategy for Trunk repository. It includes files in PNG, draw.io, and Figma formats. You can modify these diagrams to support your organization's processes.

Best practices

Follow the best practices and recommendations in AWS Well-Architected DevOps Guidance and Choosing a Git branching strategy for multi-account DevOps environments. These help you effectively implement Trunk-based development, foster collaboration, improve code quality, and streamline the development process.

Epics

Reviewing the Trunk workflow

Task	Description	Skills required
Review the standard Trunk process.	1. In the sandbox environment, the developer creates a `feature` branch from the `main` branch and uses the naming pattern `feature/<ticket>_<initials>_<short description>` . 2. The developer develops code and deploys the code to the sandbox environment iteratively in order to complete the ticket. **Note** The developer can optionally create a `sandbox` branch to run an automated build or deploy	DevOps engineer

Task	Description	Skills required
	pipeline in the sandbox environme nt. 3. The developer creates a merge request from the `feature` branch into the `main` branch by using a squash merge. 4. A continuous integration and continuous delivery (CI/CD) pipeline automatic ally builds and publishes the artifacts from the `main` branch to the development environment. 5. An approver manually approves the deploymen t of the release artifacts to the development environment. 6. An approver manually approves the deployment of the release artifacts to the testing environment. 7. An approver manually approves the deployment of the release artifacts to the staging environment. 8. An approver manually approves the deployment of the release artifacts to	

Task	Description	Skills required
	the production environme nt.	

Troubleshooting

Issue	Solution
Branch conflicts	A common issue that can occur with the Trunk model is where a hotfix needs to occur in production but a corresponding change needs to occur in a `feature` branch, where the same resources are being modified. We recommend that you frequently merge changes from `main` into lower branches to avoid significant conflicts on merge to `main`.

Related resources

This guide doesn't include training for Git; however, there are many high-quality resources available on the internet if you need this training. We recommend that you start with the Git documentation site.

The following resources can help you with your Trunk branching journey in the AWS Cloud.

AWS DevOps guidance

- AWS DevOps Guidance
- AWS Deployment Pipeline Reference Architecture
- What is DevOps?
- DevOps resources

Trunk guidance

- Trunk Based Development

Other resources

- Twelve-factor app methodology (12factor.net)

Implement centralized custom Checkov scanning to enforce policy before deploying AWS infrastructure

Benjamin Morris, Amazon Web Services

Summary

This pattern provides a GitHub Actions framework for writing custom Checkov policies in one repository that can be reused across a GitHub organization. By following this pattern, an information security team can write, add, and maintain custom policies based on company requirements. The custom policies can be pulled into all pipelines in the GitHub organization automatically. This approach can be used to enforce company standards for resources before the resources are deployed.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A GitHub organization using GitHub Actions
- AWS infrastructure deployed with either HashiCorp Terraform or AWS CloudFormation

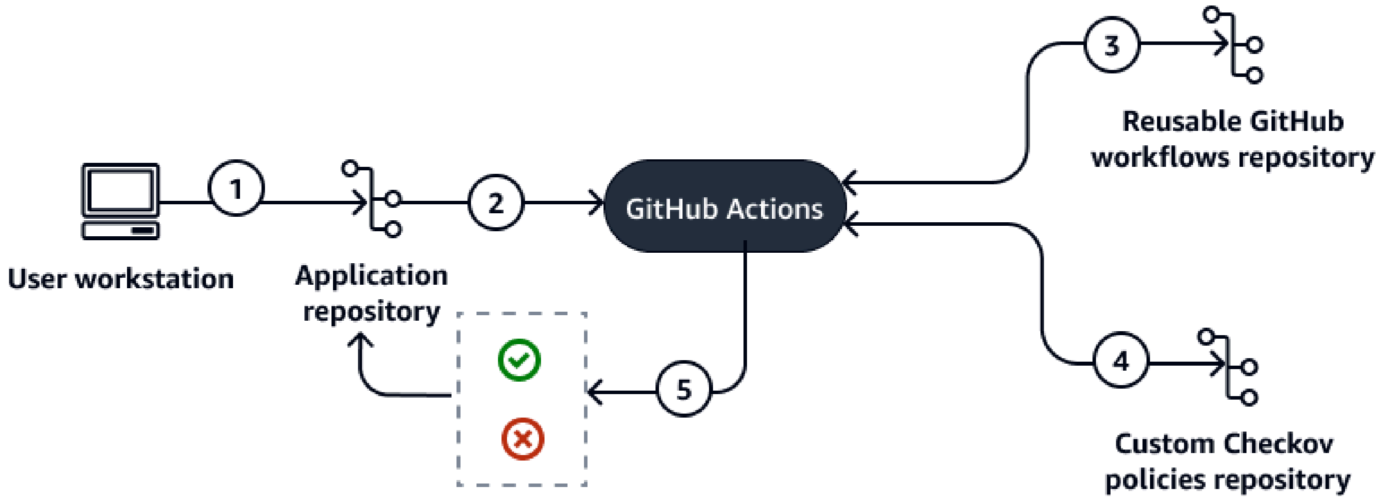
Limitations

- This pattern is written for GitHub Actions. However, it can be adapted to similar continuous integration and continuous delivery (CI/CD) frameworks such as GitLab. No specific paid version of GitHub is required.
- Some AWS services aren't available in all AWS Regions. For Region availability, see [Service endpoints and quotas] in the AWS documentation, and choose the link for the service.

Architecture

This pattern is designed to be deployed as a GitHub repository that contains a GitHub reusable workflow and custom Checkov policies. The reusable workflow can scan both Terraform and CloudFormation infrastructure as code (IaC) repositories.

The following diagram shows the **Reusable GitHub workflows repository** and **Custom Checkov policies repository** as separate icons. However, you can implement these repositories either as separate repositories or a single repository. The example code uses a single repository, with files for workflows (`.github/workflows`) and files for custom policies (`custom_policies` folder and the `.checkov.yml` config file) in the same repository.



The diagram shows the following workflow:

1. A user creates a pull request in a GitHub repository.
2. Pipeline workflows start in GitHub Actions, including a reference to a Checkov reusable workflow.
3. The pipeline workflow downloads the referenced Checkov reusable workflow from an external repository and runs that Checkov workflow by using GitHub Actions.
4. The Checkov reusable workflow downloads the custom policies from an external repository.
5. The Checkov reusable workflow evaluates the IaC in the GitHub repository against both built-in and custom Checkov policies. The Checkov reusable workflow passes or fails based on whether security issues are found.

Automation and scale

This pattern allows for central management of Checkov configuration, so that policy updates can be applied in one location. However, this pattern does require that each repository use a workflow that contains a reference to the central reusable workflow. You can add this reference manually or use scripts to push the file to the `.github/workflows` folder for each repository.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions. Checkov can scan CloudFormation.

Other tools

- Checkov is a static code analysis tool that checks IaC for security and compliance misconfigurations.
- GitHub Actions is integrated into the GitHub platform to help you create, share, and run workflows within your GitHub repositories. You can use GitHub Actions to automate tasks such as building, testing, and deploying your code.
- Terraform is an IaC tool from HashiCorp that helps you create and manage cloud and on-premises resources. Checkov can scan Terraform.

Code repository

The code for this pattern is available in the GitHub centralized-custom-checkov-sast repository.

Best practices

- To maintain a consistent security posture, align your company's security policies with the Checkov policies.
- In the early phases of implementing Checkov custom policies, you can use the soft-fail option in your Checkov scan to allow IaC with security issues to be merged. As the process matures, switch from the soft-fail option to the hard-fail option.

Epics

Create a central Checkov repository for custom policies

Task	Description	Skills required
Create a central Checkov repository.	Create a repository to store custom Checkov policies that will be used within the organization.	DevOps engineer

Task	Description	Skills required
	For a quick start, you can copy the contents of this pattern's GitHub centralized-custom-checkov-sast repository into your central Checkov repository.	
Create a repository for reusable workflows.	If a repository for reusable workflows already exists, or you plan to include reusable workflow files in the same repository as the custom Checkov policies, you can skip this step. Create a GitHub repository to hold reusable workflows. Other repositories' pipelines will reference this repository.	DevOps engineer

Create reusable and example Checkov workflows

Task	Description	Skills required
Add a reusable Checkov workflow.	Create a reusable Checkov GitHub Actions workflow (YAML file) in the reusable workflows repository. You can adapt this reusable workflow from the workflow file provided in this pattern. An example of a change that you might want to make is to change the reusable	DevOps engineer

Task	Description	Skills required
	workflow to use the soft-fail option. Setting `soft-fail` to `true` allows the job to complete successfully even if there is a failed Checkov scan. For instructions, see [Hard and soft fail](#) in the Checkov documentation.	
Add an example workflow.	Add an example Checkov workflow that references the `reusable` workflow. This will provide a template for how to reuse the `reusable` workflow. In the example repository, `checkov-source.yaml` is the reusable workflow and `checkov-scan.yaml` is the example that consumes `checkov-source`. For more details about writing an example Checkov workflow, see [Additional information](#).	DevOps engineer

Associate company policies to Checkov custom policies

Task	Description	Skills required
Determine policies that can be enforced with Checkov.	1. Review company policies that are related to infrastructure security and which	Security and Compliance

Task	Description	Skills required
	requirements should be in place. 2. Determine which requirements can be implemented by using Checkov custom policies. 3. Create a naming convention that maps the policy control to the Checkov custom policy. Typically, Checkov custom policies have an identifier with a Checkov name, the policy source (custom), and a policy number (for example, `CKV2_CUSTOM_123`). For more details about creating Checkov custom policies, see Custom Policies Overview in the Checkov documentation.	
Add Checkov custom policies.	Convert the identified company policies to custom Checkov policies in the central repository. You can write simple Checkov policies in either Python or YAML.	Security

Implement centralized Checkov custom policies

Task	Description	Skills required
Add the Checkov reusable workflow to all repositories.	At this point, you should have an example Checkov workflow that references the reusable workflow. Copy the sample Checkov workflow that references the reusable workflow to each repository that requires it.	DevOps engineer
Create a mechanism to ensure that Checkov runs before merges.	To ensure that the Checkov workflow gets run for every pull request, create a [status check](https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/collaborating-on-repositories-with-code-quality-features/about-status-checks) that requires a successful Checkov workflow before pull requests can be merged. GitHub allows you to require that specific workflows run before pull requests can be merged.	DevOps engineer
Create an organization-wide PAT, and share it as a secret.	If your GitHub organization is publicly visible, you can skip this step. This pattern requires that the Checkov workflow be able to download custom policies from the custom policy repository in your GitHub organization. You must provide permissio ns such that the Checkov workflow can access those repositories.	DevOps engineer

Task	Description	Skills required
	To do this, create a personal access token (PAT) with permissions to read organizat ion repositories. Share this PAT with repositories, either as an organization-wide secret (if on a paid plan) or a secret in each repository (free version). In the sample code, the default name for the secret is `ORG_PAT`.	

Task	Description	Skills required
(Optional) Protect the Checkov workflow files from modification.	To protect the Checkov workflow files from unwanted changes, you can use a `CODEOWNERS` file. The `CODEOWNERS` file is typically deployed in the root of the directory. For example, to require approvals from your GitHub organization's `secEng` group when the `checkov-scan.yaml` file is modified, append the following to a repository's `CODEOWNERS` file: `[Checkov] .github/workflows/checkov-scan.yaml @myOrg/secEng` A `CODEOWNERS` file is specific to the repository it lives in. To protect the Checkov workflow used by the repository, you must add (or update) a `CODEOWNERS` file in each repository. For more information about protecting Checkov workflow files, see [Additional information](). For more information about `CODEOWNERS` files, see the official documentation for	DevOps engineer

Task	Description	Skills required
	your CI/CD provider (such as GitHub).	

Related resources

- Checkov Custom Policies Overview
- CloudFormation Configuration Scanning
- GitHub Actions Reusable Workflows

Additional information

Writing Checkov workflow files

When writing `checkov-scan.yaml`, consider when you want it to run. The top-level `on` key determines when the workflow runs. In the example repository, the workflow will run when there is a pull request targeting the `main` branch (and any time that pull request's source branch is modified). The workflow can also be run as required because of the `workflow_dispatch` key.

You can change the workflow trigger conditions based on how often you want the workflow to run. For example, you could change the workflow to run every time code is pushed to any branch by replacing `pull_request` with `push` and removing the `branches` key.

You can modify the example workflow file that you created within an individual repository. For example, you could adjust the target branch's name from `main` to `production` if a repository is structured around a `production` branch.

Protecting Checkov workflow files

Checkov scanning provides useful information about potential security misconfiguration. However, some developers might perceive it to be a barrier to their productivity and attempt to remove or disable the scanning workflow.

There are several ways to address this problem, including better messaging about the long-term value of security scanning and clearer documentation about how to deploy secure infrastructure. These are important "soft" approaches to DevSecOps collaboration that can be seen as the solution to this problem's root cause. However, you can also use technical controls such as a `CODEOWNERS` file as guardrails to help keep developers on the right path.

Testing pattern in a sandbox

To test this pattern in a sandbox environment, follow these steps:

1. Create a new GitHub organization. Create a token with read-only access to all repositories in the organization. Because this token is for a sandbox environment, not a paid environment, you will not be able to store this token in an organization-wide secret.
2. Create a `checkov` repository to hold the Checkov configuration and a `github-workflows` repository to hold the reusable workflow configuration. Populate the repositories with the contents of the example repository.
3. Create an application repository, and copy and paste the `checkov-scan.yaml` workflow to its `.github/workflows` folder. Add a secret to the repository that contains the PAT you created for organization read-only access. The default secret is `ORG_PAT`.
4. Create a pull request that adds some Terraform or CloudFormation code to the application repository. Checkov should scan and return a result.

Implement AI-powered Kubernetes diagnostics and troubleshooting with K8sGPT and Amazon Bedrock integration

Ishwar Chauthaiwale, Muskan ., and Prafful Gupta, Amazon Web Services

Summary

This pattern demonstrates how to implement AI-powered Kubernetes diagnostics and troubleshooting by integrating K8sGPT with the Anthropic Claude v2 model available on Amazon Bedrock. The solution provides natural language analysis and remediation steps for Kubernetes cluster issues through a secure bastion host architecture. By combining K8sGPT Kubernetes expertise with Amazon Bedrock advanced language capabilities, DevOps teams can quickly identify and resolve cluster problems. With these capabilities, it's possible to reduce mean time to resolution (MTTR) by up to 50 percent.

This cloud-native pattern leverages Amazon Elastic Kubernetes Service (Amazon EKS) for Kubernetes management. The pattern implements security best practices through proper AWS Identity and Access Management (IAM) roles and network isolation. This solution is particularly valuable for organizations who want to streamline their Kubernetes operations and enhance their troubleshooting capabilities with AI assistance.

Prerequisites and limitations

Prerequisites

- An active AWS account with appropriate permissions
- AWS Command Line Interface (AWS CLI) installed and configured
- An Amazon EKS cluster
- Access to Anthropic Claude 2 model on Amazon Bedrock
- A bastion host with required security group settings
- K8sGPT installed

Limitations

- K8sGPT analysis is limited by the context window size of the Claude v2 model.
- Amazon Bedrock API rate limits apply based on your account quotas.

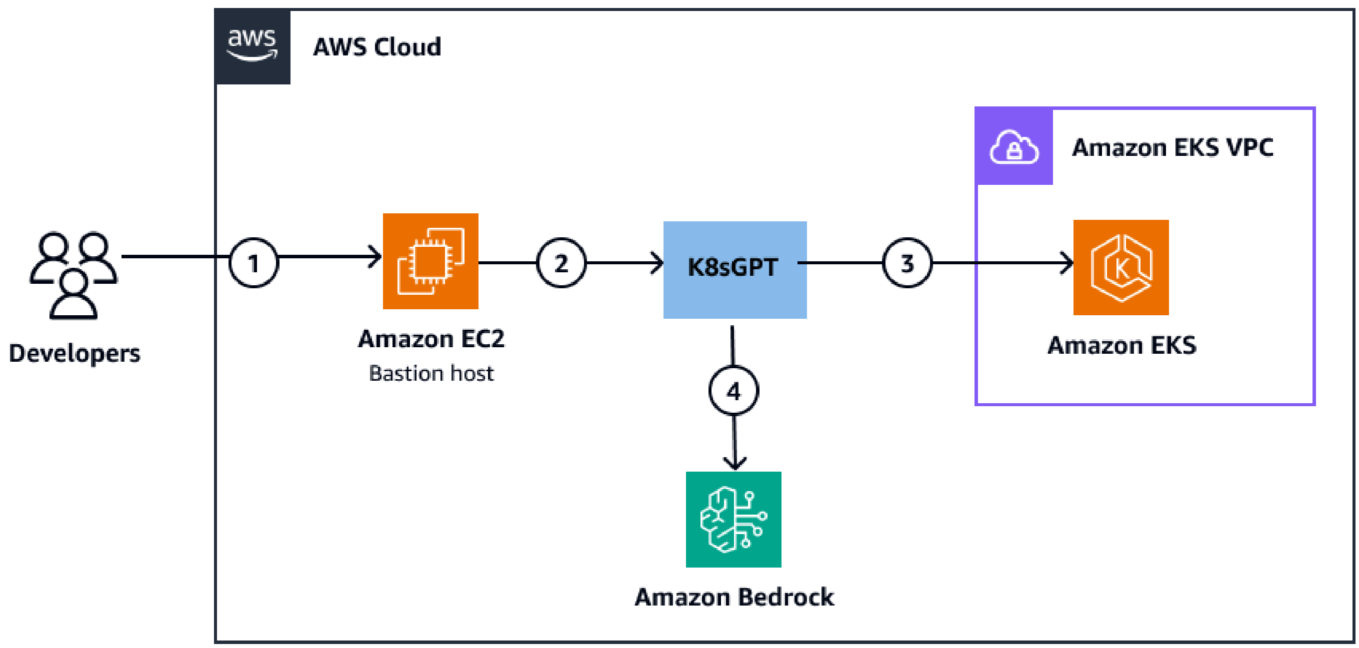
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- Amazon EKS version 1.31 or later
- Claude 2 model on Amazon Bedrock
- K8sGPT v0.4.2 or later

Architecture

The following diagram shows the architecture for AI-powered Kubernetes diagnostics using K8sGPT integrated with Amazon Bedrock in the AWS Cloud.



The architecture shows the following workflow:

1. Developers access the environment through a secure connection to the bastion host. This Amazon EC2 instance serves as the secure entry point and contains the K8sGPT command line interface (CLI) installation and required configurations.

2. The bastion host, configured with specific IAM roles, establishes secure connections to both the Amazon EKS cluster and the Amazon Bedrock endpoints. K8sGPT is installed and configured on the bastion host to perform Kubernetes cluster analysis.
3. Amazon EKS manages the Kubernetes control plane and worker nodes, providing the target environment for K8sGPT analysis. The service runs across multiple Availability Zones within a virtual private cloud (VPC), which helps to provide high availability and resilience. Amazon EKS supplies operational data through the Kubernetes API, enabling comprehensive cluster analysis.
4. K8sGPT sends analysis data to Amazon Bedrock, which provides the Claude v2 foundation model (FM) for natural language processing. The service processes K8sGPT analysis to generate human-readable explanations and offers detailed remediation suggestions based on identified issues. Amazon Bedrock operates as a serverless AI service with high availability and scalability.

Note

Throughout this workflow, IAM controls access between components through roles and policies, managing authentication for the bastion host, Amazon EKS, and Amazon Bedrock interactions. IAM implements the principle of least privilege and enables secure cross-service communication throughout the architecture.

Automation and scale

K8sGPT operations can be automated and scaled across multiple Amazon EKS clusters through various AWS services and tools. This solution supports continuous integration and continuous deployment (CI/CD) integration using Jenkins, GitHub Actions, or AWS CodeBuild for scheduled analysis. The K8sGPT operator enables continuous in-cluster monitoring with automated issue detection and reporting capabilities. For enterprise-scale deployments, you can use Amazon EventBridge to schedule scans and trigger automated responses with custom scripts. AWS SDK integration enables programmatic control across large fleet of clusters.

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command line shell.

- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Other tools

- K8sGPT is an open source AI-powered tool that transforms Kubernetes management. It acts as a virtual site reliability engineering (SRE) expert, automatically scanning, diagnosing, and troubleshooting Kubernetes cluster issues. Administrators can interact with K8sGPT using natural language and get clear, actionable insights about cluster state, pod crashes, and service failures. The tool's built-in analyzers detect a wide range of issues, from misconfigured components to resource constraints, and provide easy-to-understand explanations and solutions.

Best practices

- Implement secure access controls by using AWS Systems Manager Session Manager for bastion host access.
- Make sure that K8sGPT authentication uses dedicated IAM roles with least privilege permissions for Amazon Bedrock and Amazon EKS interactions . For more information, see Grant least privilege and Security best practices in the IAM documentation.
- Configure resource tagging, enable Amazon CloudWatch logging for audit trails, and implement data anonymization for sensitive information.
- Maintain regular backups of K8sGPT configurations while setting up automated scanning schedules during off-peak hours to minimize operational impact.

Epics

Add Amazon Bedrock to AI backend provider list.

Task	Description	Skills required
Set Amazon Bedrock as the AI backend provider for K8sGPT.	To set Amazon Bedrock as the AI backend provider for K8sGPT, use the following AWS CLI command:	AWS DevOps

Task	Description	Skills required
	``` k8sgpt auth add -b amazonbedrock \ -r us-west-2 \ -m anthropic.claude-v 2 \ -n endpoint-name ``` The example command uses `us-west-2` for the AWS Region. However, you can select another Region, provided that both the Amazon EKS cluster and the corresponding Amazon Bedrock model are available and enabled in that selected Region. To check that `amazonbedrock` is added to the AI backend provider list and is in the `Active` state, run the following command: ``` k8sgpt auth list ``` Following is an example of the expected output of this command: ``` Default: > openai Active: > amazonbedrock Unused: > openai > localai ```	

Task	Description	Skills required

	<pre>> ollama
	> azureopenai
	> cohere
	> amazonsagemaker
	> google
	> noopai
	> huggingface
	> googlevertexai
	> oci
	> customrest
	> ibmwatsonxai</pre>	

Scan resources using a filter

Task	Description	Skills required

View a list of available filters.	To see the list of all available filters, use the following AWS CLI command:
	<pre>k8sgpt filters list</pre>Following is an example of the expected output of this command:
	<pre>Active:
	> Deployment
	> ReplicaSet
	> PersistentVolumeCl
	aim
	> Service
	> CronJob
	> Node
	> MutatingWebhookCon
	figuration
	> Pod
	> Ingress</pre>
		AWS DevOps

Task	Description	Skills required
	``` > StatefulSet > ValidatingWebhookC onfiguration ```	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Scan a pod in a specific namespace by using a filter.</td><td>
This command is useful for targeted debugging of specific pod issues within a Kubernetes cluster, using Amazon Bedrock AI capabilities to analyze and explain the problems it finds.
To scan a pod in a specific namespace by using a filter, use the following AWS CLI command:
<pre><code>k8sgpt analyze --backend
 amazonbedrock --
explain --filter Pod -n
 default</code></pre>
Following is an example of the expected output of this command:
<pre><code>100% |#################
##################
##################
###| (1/1, 645 it/s)

AI Provider: amazonbed
rock

0: Pod default/c
rashme()
- Error: the last
 termination reason
 is Error container
=crashme pod=crashme
Error: The pod named
 crashme terminated</code></pre>
</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	``` because the container named crashme crashed. Solution: Check logs for crashme pod to identify reason for crash. Restart pod or redeploy application to resolve crash. ```	

Task	Description	Skills required

Scan a deployment in a specific namespace by using a filter.	This command is useful for identifying and troubleshooting deployment-specific issues, particularly when the actual state doesn't match the desired state.
	
	To scan a deployment in a specific namespace by using a filter, use the following AWS CLI command:
	
	`k8sgpt analyze --backend amazonbedrock --explain --filter Deployment -n default`
	
	Following is an example of the expected output of this command:
	
	`100% |#################################################################| (1/1, 10 it/min)`
	
	`AI Provider: amazonbedrock`
	
	`0: Deployment default/nginx()`
	`- Error: Deployment default/nginx has 1 replicas but 2 are available`
	`Error: The Deployment named nginx in the default namespace has`
		AWS DevOps

Task	Description	Skills required
	``` 1 replica specified but 2 pod replicas are running. Solution: Check if any other controlle rs like ReplicaSet or StatefulSet have created extra pods. Delete extra pods or adjust replica count to match available pods. ```	
Scan a node in a specific namespace by using a filter.	To scan a node in a specific namespace by using a filter, use the following AWS CLI command: ``` k8sgpt analyze --backend amazonbedrock -- explain --filter Node -n default ``` Following is an example of the expected output of this command: ``` AI Provider: amazonbed rock No problems detected ```	AWS DevOps

Analyze detailed outputs

Task	Description	Skills required

Get detailed outputs.	To get detailed outputs, use the following AWS CLI command:
	```k8sgpt analyze --backend amazonbedrock --explain --ouput json```
	Following is an example of the expected output of this command:
	```{ "provider": "amazonbedrock", "errors": null, "status": "ProblemDetected", "problems": 1, "results": [{ "kind": "Pod", "name": "default/crashme", "error": [{ "Text": "the last termination reason is Error container=crashme pod=crashme", "KubernetesDoc": "", "Sensitive": [] }],```
		AWS DevOps

Task	Description	Skills required
	`"details": " Error: The pod named crashme terminated because the container named crashme crashed. \nSolution: Check logs for crashme pod to identify reason for crash. Restart pod or redeploy application to resolve crash.", "parentObject": "" } ] }`	

Check problematic pods.	To check for specific problematic pods, use the following AWS CLI command:
	`kubectl get pods --all-namespaces | grep -v Running`
	Following is an example of the expected output of this command:
	`NAMESPACE NAME READY STATUS RESTARTS AGE`
	`default crashme 0/1 CrashLoopBackOff 260(91s ago) 21h`
		AWS DevOps

Task	Description	Skills required

Get application-specific insights.	This command is particularly useful when:
	
	• You want to focus on a specific application in your cluster.
	• You use labels effectively to organize your Kubernetes resources.
	• You need to quickly check the health of a particular application component.
	
	To get application-specific insights, use the following command:
	
	`k8sgpt analyze --backend amazonbedrock --explain -L app=nginx -n default`
	
	Following is an example of the expected output of this command:
	
	`AI Provider: amazonbedrock`
	
	`No problems detected`	

Related resources

AWS Blogs

- Automate Amazon EKS troubleshooting using an Amazon Bedrock agentic workflow
- Use K8sGPT and Amazon Bedrock for simplified Kubernetes cluster maintenance

AWS documentation

- AWS CLI commands: create-cluster and describe-cluster
- Get started with Amazon EKS (Amazon EKS documentation)
- Security best practices in IAM (IAM documentation)

Other resources

- K8sGPT

Automatically detect changes and initiate different CodePipeline pipelines for a monorepo in CodeCommit

Helton Ribeiro, Petrus Batalha, and Ricardo Morais, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

Notice: AWS Cloud9 is no longer available to new customers. Existing customers of AWS Cloud9 can continue to use the service as normal. Learn more

This pattern helps you automatically detect changes to the source code of a monorepo-based application in AWS CodeCommit and then initiate a pipeline in AWS CodePipeline that runs the continuous integration and continuous delivery (CI/CD) automation for each microservice. This approach means that each microservice in your monorepo-based application can have a dedicated CI/CD pipeline, which ensures better visibility, easier sharing of code, and improved collaboration, standardization, and discoverability.

The solution described in this pattern doesn't perform any dependency analysis among the microservices inside the monorepo. It only detects changes in the source code and initiates the matching CI/CD pipeline.

The pattern uses AWS Cloud9 as the integrated development environment (IDE) and AWS Cloud Development Kit (AWS CDK) to define an infrastructure by using two AWS CloudFormation stacks: `MonoRepoStack` and `PipelinesStack`. The `MonoRepoStack` stack creates the monorepo in AWS CodeCommit and the AWS Lambda function that initiates the CI/CD pipelines. The `PipelinesStack` stack defines your pipeline infrastructure.

Important

This pattern’s workflow is a proof of concept (PoC). We recommend that you use it only in a test environment. If you want to use this pattern’s approach in a production environment, see Security best practices in IAM in the AWS Identity and Access Management (IAM) documentation and make the required changes to your IAM roles and AWS services.

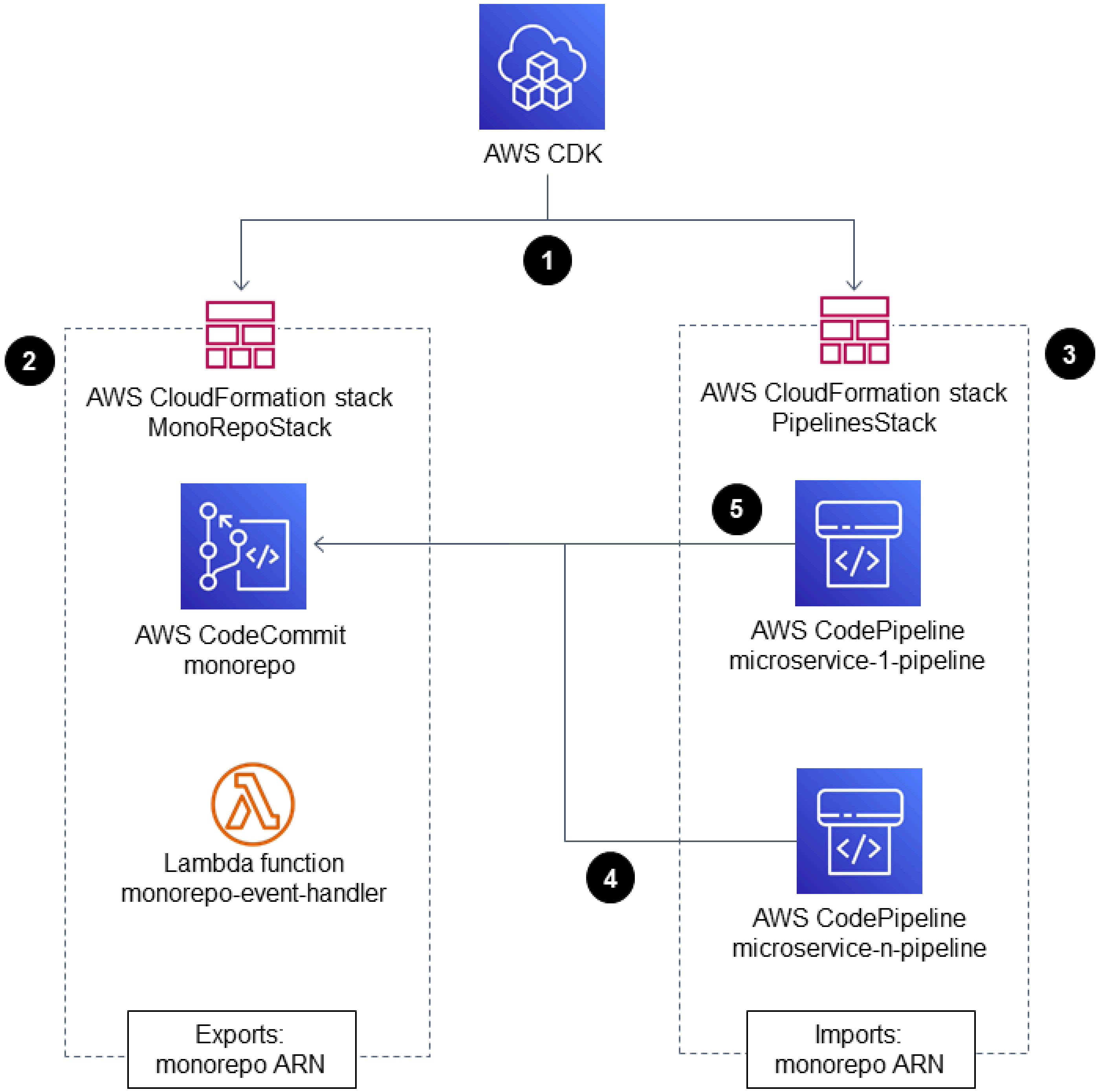
Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
- Python 3 and `pip`, installed on your local machine. For more information, see the Python documentation.
- AWS CDK, installed and configured. For more information, see Getting started with the AWS CDK in the AWS CDK documentation.
- An AWS Cloud9 IDE, installed and configured. For more information, see Setting up AWS Cloud9 in the AWS Cloud9 documentation.
- The GitHub AWS CodeCommit monorepo multi-pipeline triggers repository, cloned on your local machine.
- An existing directory containing application code that you want to build and deploy with CodePipeline.
- Familiarity and experience with DevOps best practices on the AWS Cloud. To increase your familiarity with DevOps, you can use the pattern Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9 on the AWS Prescriptive Guidance website.

Architecture

The following diagram shows how to use the AWS CDK to define an infrastructure with two AWS CloudFormation stacks: `MonoRepoStack` and `PipelinesStack`.



The diagram shows the following workflow:

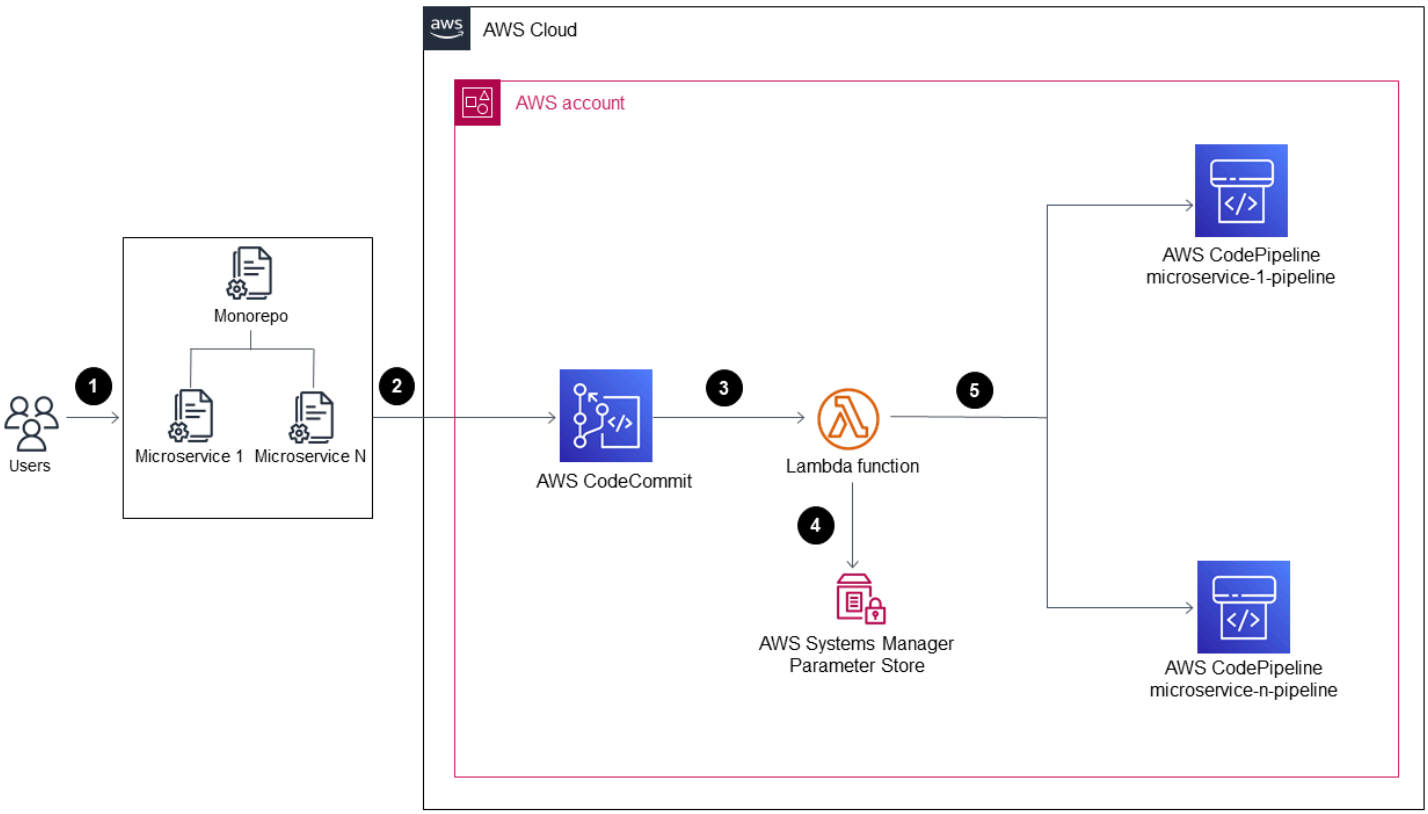
1. The bootstrap process uses the AWS CDK to create the AWS CloudFormation stacks `MonoRepoStack` and `PipelinesStack`.
2. The `MonoRepoStack` stack creates the CodeCommit repository for your application and the `monorepo-event-handler` Lambda function that is initiated after each commit.

3. The `PipelinesStack` stack creates the pipelines in CodePipeline that are initiated by the Lambda function. Each microservice must have a defined infrastructure pipeline.

4. The pipeline for `microservice-n` is initiated by the Lambda function and starts its isolated CI/CD stages that are based on the source code in CodeCommit.

5. The pipeline for `microservice-1` is initiated by the Lambda function and starts its isolated CI/CD stages that are based on the source code in CodeCommit.

The following diagram shows the deployment of the AWS CloudFormation stacks `MonoRepoStack` and `PipelinesStack` in an account.



1. A user changes code in one of the application's microservices.

2. The user pushes the changes from a local repository to a CodeCommit repository.

3. The push activity initiates the Lambda function that receives all pushes to the CodeCommit repository.

4. The Lambda function reads a parameter in Parameter Store, a capability of AWS Systems Manager, to retrieve the most recent commit ID. The parameter has the naming format: `/MonoRepoTrigger/{repository}/{branch_name}/LastCommit`. If the parameter isn't

found, the Lambda function reads the last commit ID from the CodeCommit repository and saves the returned value in Parameter Store.

5. After identifying the commit ID and the changed files, the Lambda function identifies the pipelines for each microservice directory and initiates the required CodePipeline pipeline.

Tools

- AWS Cloud Development Kit (AWS CDK) is a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation.
- Python is a programming language that lets you work quickly and integrate systems more effectively.

Code

The source code and templates for this pattern are available in the GitHub AWS CodeCommit monorepo multi-pipeline triggers repository.

Best practices

- This sample architecture doesn't include a monitoring solution for the deployed infrastructure. If you want to deploy this solution in a production environment, we recommend that you enable monitoring. For more information, see Monitor your serverless applications with CloudWatch Application Insights in the AWS Serverless Application Model (AWS SAM) documentation.
- When you edit the sample code provided by this pattern, follow the best practices for developing and deploying cloud infrastructure in the AWS CDK documentation.
- When you define your microservice pipelines, review the security best practices in the AWS CodePipeline documentation.
- You can also check your AWS CDK code for best practices by using the cdk-nag utility. This tool uses a set of rules, grouped by packs, to evaluate your code. The available packs are:
  - AWS Solutions Library
  - Health Insurance Portability and Accountability Act (HIPAA) security
  - National Institute of Standards and Technology (NIST) 800-53 rev 4
  - NIST 800-53 rev 5
  - Payment Card Industry Data Security Standard (PCI DSS) 3.2.1

Epics

Set up the environment

Task	Description	Skills required
Create a virtual Python envionment.	In your AWS Cloud9 IDE, create a virtual Python environment and install the required dependencies by running the following command: `make install`	Developer
Bootstrap the AWS account and AWS Region for the AWS CDK.	Bootstrap the required AWS account and Region by running the following command: `make bootstrap account-id=<your-AWS-account-ID> region=<required-region>`	Developer

Add a new pipeline for a microservice

Task	Description	Skills required
Add your sample code to your application directory.	Add the directory that contains your sample application code to the `monorepo-sample` directory in the cloned GitHub AWS CodeCommi	Developer

Task	Description	Skills required
	t monorepo multi-pipeline triggers repository.	
Edit the `monorepo-main.json` file.	Add the directory name of your application's code and the pipeline's name to the `monorepo-main.json` file in the cloned repository .	Developer
Create the pipeline.	In the `Pipelines` directory for the repository, add the pipeline `class` for your application. The directory contains two sample files, `pipeline_hotsite.py` and `pipeline_demo.py` . Each file has three stages: source, build, and deploy. You can copy one of the files and makes changes to it according to your applicati on's requirements.	Developer

Task	Description	Skills required

Edit the `monorepo_config.py` file.	In `service_map`, add the directory name for your application and the class that you created for the pipeline.
	
	For example, the following code shows a pipeline definition in the `Pipelines` directory that uses a file named `pipeline_mysample.py` with a `MySamplePipeline` class:
	
	`...`
	`# Pipeline definition imports`
	`from pipelines.pipeline_demo import DemoPipeline`
	`from pipelines.pipeline_hotsite import HotsitePipeline`
	`from pipelines.pipeline_mysample import MySamplePipeline`
	
	`### Add your pipeline configuration here`
	`service_map: Dict[str, ServicePipeline] = {`
	`    # folder-name -> pipeline-class`
	`    'demo': DemoPipeline(),`
	`    'hotsite': HotsitePipeline(),`
	`    'mysample': MySamplePipeline()`
		Developer

Task	Description	Skills required
	```}```	

Deploy the MonoRepoStack stack

Task	Description	Skills required
Deploy the AWS CloudFormation stack.	Deploy the AWS CloudFormation `MonoRepoStack` stack with default parameter values in the root directory of the cloned repository by running the `make deploy-core` command. You can change the repository's name by running the `make deploy-core monorepo-name=<repo_name>` command. **Note** You can simultaneously deploy both pipelines by using the `make deploy monorepo-name=<repo_name>` command.	Developer
Validate the CodeCommit repository.	Validate that your resources were created by running the `aws codecommit get-repository --reposit`	Developer

Task	Description	Skills required
	`ory-name <repo_name>` command. **Important** Because the AWS CloudFormation stack creates the CodeCommit repository where the monorepo is stored, don't run the `cdk destroy MonoRepoStack` command if you have started to push modifications into it.	
Validate the AWS CloudFormation stack results.	Validate that the AWS CloudFormation `MonoRepoStack` stack is correctly created and configured by running the following command: `aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE --query 'StackSummaries[?StackName == 'MonoRepoStack']'`	Developer

Deploy the PipelinesStack stack

Task	Description	Skills required
Deploy the AWS CloudFormation stack.	The AWS CloudFormation `PipelinesStack` stack must be deployed after you deploy the `MonoRepoStack` stack. The stack increases in size when new microservices are added to the monorepo's code base and is redeployed when a new microservice is onboarded. Deploy the PipelinesStack stack by running the `make deploy-pipelines` command. **Note** You can also deploy simultaneously deploy both pipelines by running the `make deploy monorepo-name=<repo_name>` command. The following sample output shows how the `PipelinesStacks` deployment prints the URLs for the microservices at the end of the implementation:	Developer

Task	Description	Skills required
	`Outputs: PipelinesStack.demourl = .cloudfront.net PipelinesStack.hotsiteurl = .cloudfront.net`	
Validate the AWS CloudFormation stack results.	Validate that the AWS CloudFormation `PipelinesStacks` stack is correctly created and configured by running the following command: `aws cloudformation list-stacks --stack-status-filter CREATE_COMPLETE UPDATE_COMPLETE --query 'StackSummaries[?StackName == 'PipelinesStack']'`	Developer

Clean up resources

Task	Description	Skills required
Delete your AWS CloudFormation stacks.	Run the `make destroy` command.	Developer
Delete the S3 buckets for your pipelines.	1. Sign in to the [AWS Management Console](#) and open the [Amazon Simple Storage Service (Amazon S3) console](#).	Developer

Task	Description	Skills required
	2. Delete the S3 buckets that are associated with your pipelines and use the following name: `pipelinesstack-codepipeline*`	

Troubleshooting

Issue	Solution
I encountered AWS CDK issues.	See [Troubleshooting common AWS CDK issues](#) in the AWS CDK documentation.
I pushed my microservice code, but the microservice pipeline didn't run.	**Setup validation** *Verify branch configuration:* • Make sure that you are pushing your code to the correct branch. This pipeline is configured to run only when changes are made to the `main` branch. Pushes to other branches don't initiate the pipeline unless they are configured specifically. • After you push your code, check if the commit is visible in AWS CodeCommit to make sure that the push was successful and that the connection between your local environment and the repository is intact. Refresh your credentials if there are issues pushing code. *Validate configuration files:*

Issue	Solution
	• Confirm that the `service_map` variable in `monorepo_config.py` accurately reflects the current directory structure of your microservices. This variable plays a crucial role in mapping your code push to the respective pipeline. • Make sure that `monorepo-main.json` is updated to include the new mapping for your microservice. This file is essential for the pipeline to recognize and correctly handle changes to your microservice. **Troubleshooting on the console** *AWS CodePipeline checks:* • On the AWS Management Console, confirm that you're in the AWS Region where your pipeline is hosted. Open the CodePipeline console and check if the pipeline that corresponds to your microservice has been initiated. **Error analysis**: If the pipeline was initiated but failed, review any error messages or logs provided by CodePipeline to understand what went wrong. *AWS Lambda troubleshooting:* • On the AWS Lambda console, open the `monorepo-event-handler` Lambda function. Verify that the function was initiated in response to the code push.

Issue	Solution
	Log analysis: Examine the Lambda function's logs for any issues. The logs can provide detailed insights into what happened when the function ran and help identify whether the function processed the event as expected.

Issue	Solution
I need to redeploy all my microservices.	There are two approaches to force the redeployment of all microservices. Choose the option that fits your requirements. **Approach 1: Delete a parameter in Parameter Store** This method involves deleting a specific parameter within Systems Manager Parameter Store that tracks the last commit ID used for deployment. When you remove this parameter , the system is forced to redeploy all microserv ices upon the next trigger, because it perceives it as a fresh state. Steps: 1. Locate the specific Parameter Store entry that holds the commit ID or a related deployment marker for your monorepo. The parameter name follows the format: `"/MonoRepoTrigger/{repository}/{branch_name}/LastCommit"` 2. Consider backing up the parameter value if it's critical or if you wish to maintain a record of the deployment state before resetting it. 3. Use the AWS Management Console, AWS CLI, or SDKs to delete the identifie d parameter. This action resets the deployment marker. 4. After deletion, the next push to the repository should cause the system to deploy all microservices, because it looks

Issue	Solution
	for the latest commit to consider for deployment. Pros: • Simple and quick to implement with minimal steps. • Doesn't require making arbitrary code changes to initiate deployments. Cons: • Less granular control over the deployment process. • Potentially risky if the Parameter Store is used for managing other critical configura tions. **Approach 2: Push a commit in each monorepo subfolder** This method involves making a minor change and pushing it in each microservice subfolder within the monorepo to initiate their individua l pipelines. Steps: 1. List all the microservices within the monorepo that need redeployment. 2. For each microservice, make a minimal, non-impactful change in its subfolder. This might be updating a `README` file, adding a comment in a configuration file, or any

Issue	Solution
	change that doesn't affect the service's functionality. 3. Commit these changes with a clear message (such as "Initiate redeployment of microserv ices") and push them to the repository. Make sure that you push the changes to the branch that initiates the deployment. 4. Monitor the pipelines for each microserv ice to confirm that they are initiated and complete successfully. Pros: • Provides granular control over which microservices are redeployed. • Safer because it doesn't involve deleting configuration parameters that might be used for other purposes. Cons: • More time-consuming, especially with a large number of microservices. • Requires making unnecessary code changes that could clutter the commit history.

Related resources

- [Continuous integration and delivery (CI/CD) using CDK Pipelines](#) (AWS CDK documentation)
- [aws-cdk/pipelines module](#) (AWS CDK API reference)

Integrate a Bitbucket repository with AWS Amplify using AWS CloudFormation

Alwin Abraham, Amazon Web Services

Summary

AWS Amplify helps you to quickly deploy and test static websites without having to set up the infrastructure that is typically required. You can deploy this pattern's approach if your organization wants to use Bitbucket for source control, whether to migrate existing application code or build a new application. By using AWS CloudFormation to automatically set up Amplify, you provide visibility into the configurations that you use.

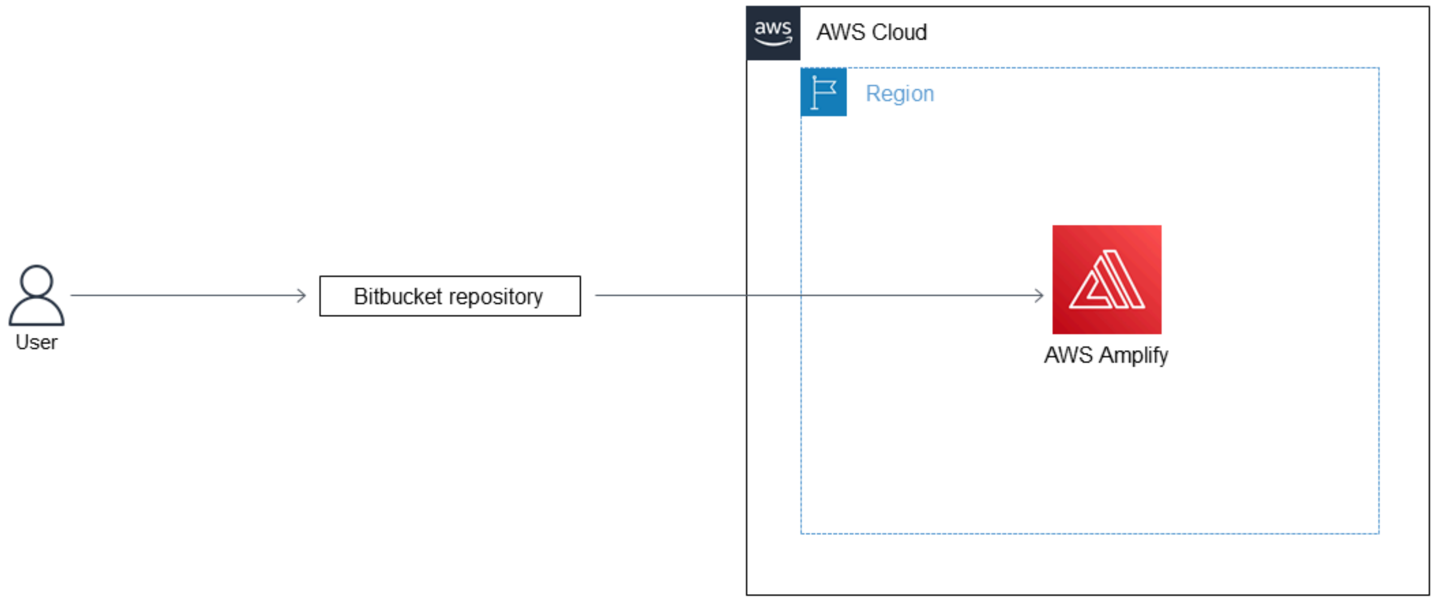
This pattern describes how to create a front-end continuous integration and continuous deployment (CI/CD) pipeline and deployment environment by using AWS CloudFormation to integrate a Bitbucket repository with AWS Amplify. The pattern's approach means that you can build an Amplify front-end pipeline for repeatable deployments.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- An active Bitbucket account with administrator access
- Access to a terminal that uses cURL or the Postman application
- Familiarity with Amplify
- Familiarity with AWS CloudFormation
- Familiarity with YAML-formatted files

Architecture



Technology stack

- Amplify
- AWS CloudFormation
- Bitbucket

Tools

- AWS Amplify – Amplify helps developers to develop and deploy cloud-powered mobile and web apps.
- AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS.
- Bitbucket – Bitbucket is a Git repository management solution designed for professional teams. It gives you a central place to manage Git repositories, collaborate on your source code, and guide you through the development flow.

Code

The `bitbucket-amplify.yml` file (attached) contains the AWS CloudFormation template for this pattern.

Epics

Configure the Bitbucket repository

Task	Description	Skills required
(Optional) Create a Bitbucket repository.	1. Sign in to your Bitbucket account and create a new repository. For more information about this, see Create a Git repository in the Bitbucket documenta tion. 2. Record the workspace's name. **Note** You can also use an existing Bitbucket repository.	DevOps engineer
Open the workspace settings.	1. Open the workspace and choose the **Repository** tab. 2. Choose the repository that you want to integrate with Amplify. 3. Choose the name of the workspace that is above the repository's name. 4. On the sidebar, choose **Settings**.	DevOps engineer
Create an OAuth consumer.	1. In the **Apps and Features** section, choose **OAuth**	DevOps engineer

Task	Description	Skills required
	consumers, and then choose **Add consumer.** 2. Enter a name for your consumer, for example, `Amplify Integration` . 3. Enter a callback URL. Although this field is a required input, it's not used to complete the integrati on so the value could be `http://localhost:3000` 4. Check the box for **This is a private consumer**. 5. Choose the following permissions: • **Project –** `Read` • **Repositories –** `Admin` • **Pull requests –** `Read` • **Webhooks –** `Read and Write` 6. Leave the default choices for all the other fields and choose **Submit**. 7. Record the key and secret that are generated.	

Task	Description	Skills required
Obtain OAuth access token.	1. Open a terminal window and run the following command: `curl -X POST -u "KEY:SECRET" https://bitbucket.org/site/oauth2/access_token -d grant_type=client_credentials` **Important** Replace KEY and SECRET with the key and secret that you recorded earlier. 2. Record the access token without using the quotation marks. The token is only valid for a limited time and the default time is two hours. You must run the AWS CloudFormation template in this timeframe.	DevOps engineer

Create and deploy the AWS CloudFormation stack

Task	Description	Skills required
Download the AWS CloudFormation template.	Download the `bitbucket-amplify.yml` AWS	

Task	Description	Skills required
	CloudFormation template (attached). This template creates the CI/CD pipeline in Amplify, in addition to the Amplify project and branch.	

Task	Description	Skills required
Create and deploy the AWS CloudFormation stack.	1. Sign in to the AWS Management Console in the AWS Region that you want to deploy in and open the AWS CloudFormation console. 2. Choose **Create Stack (with new resources)** and then choose **Upload a Template File**. 3. Upload the `bitbucket-amplify.yml` file. 4. Choose **Next**, enter a stack name, and then enter the following parameters: • **Access token**: Paste the OAuth access token that you created earlier. • **Repository URL**: Add the Bitbucket project repository's URL. The URL is typically in the following format: `https://bitbucket.org/<WORKSPACE_NAME>/<REPO_NAME>` • **Branch name**: This must match the name of a branch in your Bitbucket repository. This branch doesn't need to exist when you run the AWS	DevOps engineer

Task	Description	Skills required
	CloudFormation stack but it is required for deploying code to the environment. • **Project name**: This is the name to associate with the Amplify project. 5. Choose **Next** and then choose **Create Stack**.	

Test the CI/CD pipeline

Task	Description	Skills required
Deploy the code to the branch in your repository.	1. Clone your Bitbucket repository by running the following command: `git clone https://bitbucket.org/<WORKSPACE_NAME>/<REPO_NAME>` 2. Check out the branch name that was used when running the AWS CloudFormation script. To create and check out a new branch, run the `git checkout -b <BRANCH_NAME>` command. To check out an existing branch, run the `git checkout`	App developer

Task	Description	Skills required
	<BRANCH_NAME> command 3. Commit the code into the branch and push it to the remote branch by running the `git commit` and `git push` commands. 4. Amplify then builds and deploys the application. For more information about this, see Basic Git commands in the Bitbucket documentation.	

Related resources

Authentication methods (Atlassian documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function

Richard Milner-Watts and Amit Anjarlekar, Amazon Web Services

Summary

This pattern demonstrates how to asynchronously launch an AWS CodeBuild project across multiple AWS accounts by using AWS Step Functions and an AWS Lambda proxy function. You can use the pattern's sample Step Functions state machine to test the success of your CodeBuild project.

CodeBuild helps you launch operational tasks using the AWS Command Line Interface (AWS CLI) from a fully-managed runtime environment. You can change the behavior of your CodeBuild project at runtime by overriding environment variables. Additionally, you can use CodeBuild to manage workflows. For more information, see Service Catalog Tools on the AWS Workshop website and Schedule jobs in Amazon RDS for PostgreSQL using AWS CodeBuild and Amazon EventBridge on the AWS Database Blog.

Prerequisites and limitations

Prerequisites

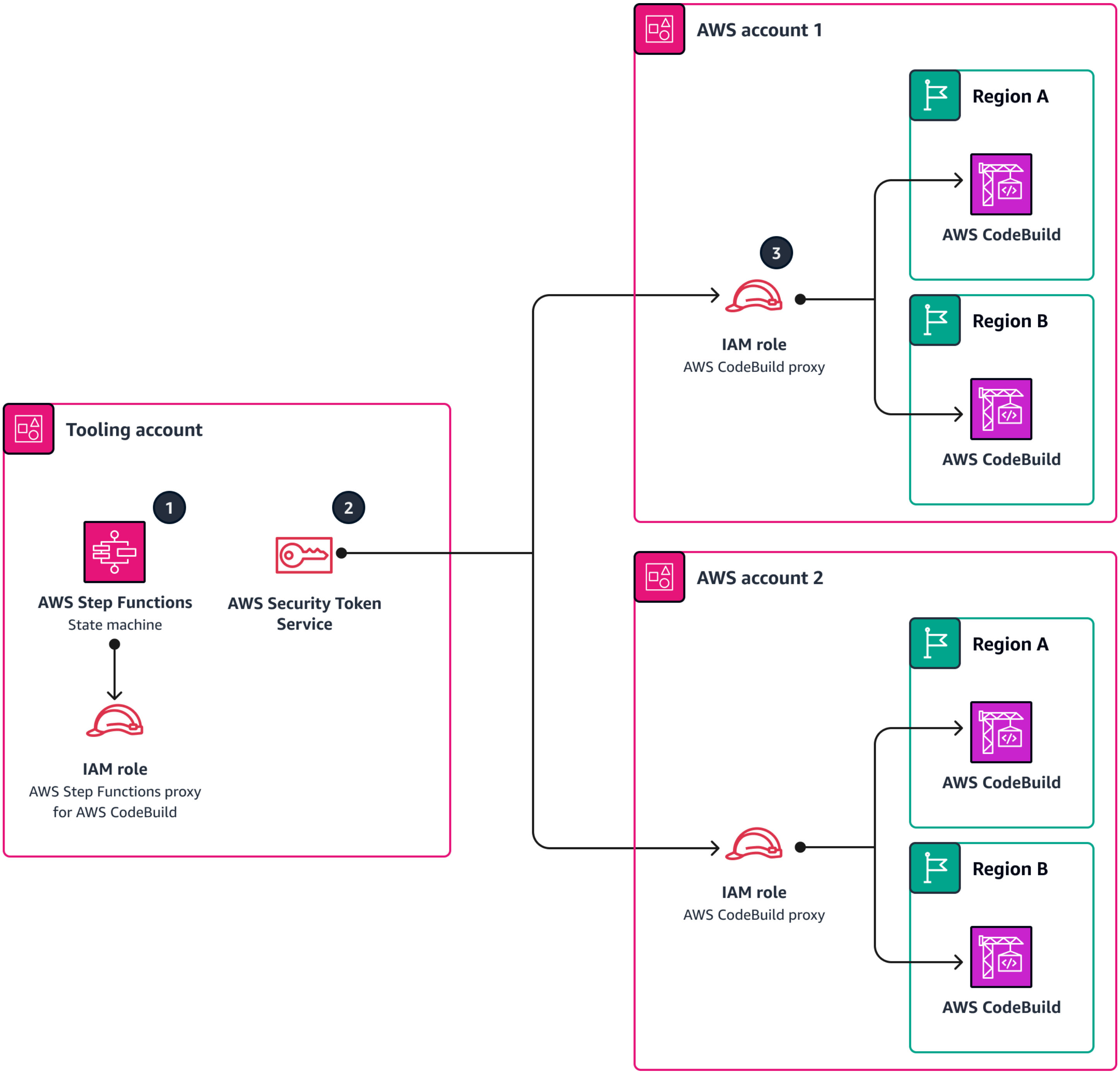
- Two active AWS accounts: a source account for invoking a Lambda proxy function with Step Functions and a target account for building a remote CodeBuild sample project

Limitations

- This pattern cannot be used to copy artifacts between accounts.

Architecture

The following diagram shows the architecture that this pattern builds.

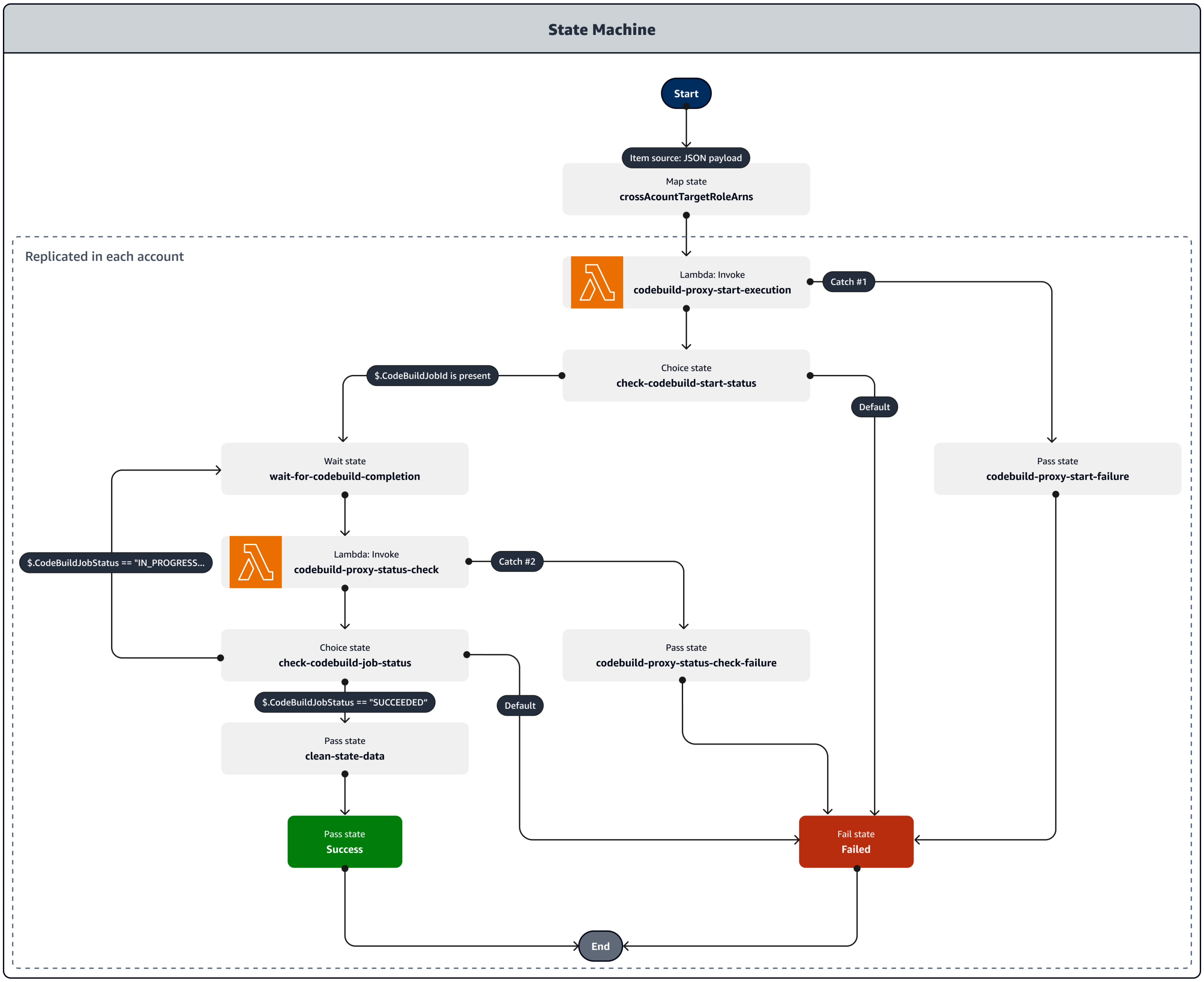


The diagram shows the following workflow:

1. The Step Functions state machine parses the supplied input map and invokes the Lambda proxy function (`codebuild-proxy-lambda`) for each account, Region, and project you defined.
2. The Lambda proxy function uses AWS Security Token Service (AWS STS) to assume an IAM proxy role (`codebuild-proxy-role`), which is associated with an IAM policy (`codebuild-proxy-policy`) in the target account.

3. Using the assumed role, the Lambda function launches the CodeBuild project and returns the CodeBuild job ID. The Step Functions state machine loops and polls the CodeBuild job until receiving a success or failure status.

The state machine logic is shown in the following image.



Technology stack

- AWS CloudFormation
- CodeBuild
- IAM

- Lambda
- Step Functions
- X-Ray

Tools

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CloudFormation Designer provides an integrated JSON and YAML editor that helps you view and edit CloudFormation templates.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- AWS X-Ray helps you collect data about the requests that your application serves, and provides tools that you can use to view, filter, and gain insights into that data to identify issues and opportunities for optimization.

Code

The sample code for this pattern is available in the GitHub Cross Account CodeBuild Proxy repository. This pattern uses the AWS Lambda Powertools for Python library to provide logging and tracing functionality. For more information on this library and its utilities, see Powertools for AWS Lambda (Python).

Best practices

1. Adjust the wait time values in the Step Function state machine to minimize polling requests for job status. Use the expected execution time for the CodeBuild project.

2. Adjust the `MaxConcurrency` property of the map in Step Functions to control how many CodeBuild projects can run in parallel.
3. If required, review the sample code for production readiness. Consider what data might be logged by the solution and whether the default Amazon CloudWatch encryption is sufficient.

Epics

Create the Lambda proxy function and associated IAM role in the source account

Task	Description	Skills required
Record the AWS account IDs.	AWS account IDs are required to set up access across accounts. Record the AWS account ID for your source and target accounts. For more information, see [Finding your AWS account ID](#) in the IAM documentation.	AWS DevOps
Download the AWS CloudFormation templates.	1. Download the `sample_target_codebuild_template.yaml` AWS CloudFormation template from the [GitHub repository](#) for this pattern. 2. Download the `codebuild_lambda_proxy_template.yaml` AWS CloudFormation template from the [GitHub repository](#) for this pattern.	AWS DevOps

Task	Description	Skills required
	Note In the AWS CloudFormation templates, `<SourceAccountId>` is the AWS account ID for the source account, and `<TargetAccountId>` is the AWS account ID for the target account.	

Task	Description	Skills required
Create and deploy the AWS CloudFormation stack.	1. Sign in to the AWS Management Console for your source account, open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/), and then choose **Stacks**. 2. Choose **Create Stack** and then choose **With new resources (standard)**. 3. For **Template source**, choose **Upload a template file**. 4. For **Upload a template** file, choose **file**, and then choose your downloaded `codebuild_lambda_proxy_template.yaml` file. Choose **Next**. 5. For **Stack name**, enter a name for the stack (for example, `codebuild-lambda-proxy`). 6. **Note** Replace the `crossAccountTargetRoleArn` parameter with your `<TargetAccountId>` (for example, `<arn:aws:`	AWS DevOps

Task	Description	Skills required
	`iam::123456789012:role/proxy-lambda-codebuild-role`>). : You aren't required to update the default value for the `targetCodeBuildProject` parameter. 7. Choose **Next**, accept the default stack creation options, and then chose **Next**. 8. Choose the **I acknowledge that AWS CloudFormation might create IAM resources with custom names** check box, and then choose **Create stack**. **Note** You must create the AWS CloudFormation stack for the proxy Lambda function before creating any resources in target accounts. When you create a trust policy in a target account, the	

Task	Description	Skills required
	IAM role is translated from the role name to an internal identifier. This is why the IAM role must already exist.	
Confirm the creation of the proxy function and state machine.	1. Wait for the AWS CloudFormation stack to reach **CREATE_COMPLETE** status. This should take less than one minute. 2. Open the [AWS Lambda console](https://console.aws.amazon.com/lambda/), choose **Functions**, and then find the `lambda-proxy-ProxyLambda-<GUID>` function. 3. Open the [AWS Step Functions console](https://console.aws.amazon.com/states/), choose **state machines**, and then find the `sample-crossaccount-codebuild-state-machine` state machine.	AWS DevOps

Create an IAM role in the target account and launch a sample CodeBuild project

Task	Description	Skills required
Create and deploy the AWS CloudFormation stack.	1. Sign in to the AWS Management Console for your target account, open the [AWS CloudFormation](https://console.aws.amazon.com/cloudformation/)	AWS DevOps

Task	Description	Skills required
	console, and then choose **Stacks**. 2. Choose **Create Stack** and then choose **With new resources (standard)**. 3. For **Template source**, choose **Upload a template file**. 4. For **Upload a template** file, choose **Choose file**, and then choose the sample_target_codebuild_template.yaml file. Choose **Next**. 5. For **Stack name**, enter a name for the stack (for example: sample-codebuild-stack). 6. Replace the crossAccountSourceRoleArn parameter with your <SourceAccountId> (for example, <arn:aws:iam::123456789012:role/codebuild-proxy-lambda-role>). 7. Choose **Next**, accept the default stack creation options, and then choose **Next**. 8. Choose the **I acknowledge that AWS CloudFormation might create IAM**	

Task	Description	Skills required
	resources with custom names check box, and then choose **Create stack**.	
Verify the creation of the sample CodeBuild project.	1. Wait for the AWS CloudFormation stack to reach **CREATE_COMPLETE** status. This should take less than one minute. 2. Open the [AWS CodeBuild console](#) and then find the `sample-codebuild-project` project.	AWS DevOps

Test the cross-account Lambda proxy function

Task	Description	Skills required
Launch the state machine.	1. Sign in to the AWS Management Console for your source account, open the [AWS Step Functions console](#), and then choose **State machines**. 2. Choose the `sample-crossaccount-codebuild-state-machine` state machine and then choose **Start execution**. 3. In the **Input** editor, enter the following JSON, and replace `<TargetAccountID>` with the AWS account ID of the	AWS DevOps

Task	Description	Skills required

	account that contains the CodeBuild project.
	``
	{
	  "crossAccountTargetRoleArns": [
	    {
	      "arn": "arn:aws:iam::<TargetAccountID>:role/proxy-lambda-codebuild-role",
	      "region": "eu-west-1",
	      "codeBuildProject": "sample-codebuild-project",
	      "SampleValue1": "Value1",
	      "SampleValue2": "Value2"
	    }
	  ]
	}
	``
	Note
	The key-value pairs are passed as environment variables from the function in the source account to the CodeBuild project in the target account.
	4. Choose **Start execution**.	

Task	Description	Skills required
	5. On the **Details** tab of the state machine page, check if **Execution Status** is set to **Succeeded**. This confirms that your state machine is running. **Note**: It can take around 30 seconds for the state machine to reach **Succeeded** status. 6. To see the output and input of a step in the state machine, expand that step in the **Execution event history** section. For example, expand the **Lambda - CodeBuild Proxy – Start** step. The output includes details on the overridden environme nt variables, the original payload, and the CodeBuild job ID.	

Task	Description	Skills required
Validate the environment variables.	1. Sign in to the AWS Management Console for your target account. 2. Open the [AWS CodeBuild console](), expand **Build**, and then choose **Build projects**. 3. Choose the `sample-codebuild-project` project, and then choose **View details**. 4. On the **Build history** tab, choose the most recent build of the project, and then choose **View logs**. 5. In the log output, verify that the environment variables printed to STDOUT match the environment variables from the Step Functions sample state machine.	AWS DevOps

Troubleshooting

Issue	Solution
Step Functions execution is taking longer than expected.	Adjust the `MaxConcurrency` property of the map in the Step Function state machine to control how many CodeBuild projects can run in parallel.

Issue	Solution
The execution of the CodeBuild jobs is taking longer than expected.	1. Adjust the wait time values in the Step Functions state machine to minimize polling requests for job status. Use the expected execution time for the CodeBuild project. 2. Consider whether CodeBuild is the appropriate tool to be using. For example, the time required to initialize a CodeBuild job can be significantly longer than AWS Lambda. If high throughput and fast completion times are a requirement, consider migrating the business logic to AWS Lambda and using a fan-out architect ure.

Manage Multi-AZ failover for EMR clusters by using Application Recovery Controller

Aarti Rajput, Ashish Bhatt, Neeti Mishra, and Nidhi Sharma, Amazon Web Services

Summary

This pattern offers an efficient disaster recovery strategy for Amazon EMR workloads to help ensure high availability and data consistency across multiple Availability Zones within a single AWS Region. The design uses Amazon Application Recovery Controller and an Application Load Balancer to manage failover operations and traffic distribution for an Apache Spark-based EMR cluster.

Under standard conditions, the primary Availability Zone hosts an active EMR cluster and application with full read/write functionality. If an Availability Zone fails unexpectedly, traffic is automatically redirected to the secondary Availability Zone, where a new EMR cluster is launched. Both Availability Zones access a shared Amazon Simple Storage Service (Amazon S3) bucket through dedicated gateway endpoints, which ensure consistent data management. This approach minimizes downtime and enables rapid recovery for critical big data workloads during Availability Zone failures. The solution is useful in industries such as finance or retail, where real-time analytics are crucial.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Amazon EMR on Amazon Elastic Compute Cloud (Amazon EC2)
- Access from the master node of the EMR cluster to Amazon S3.
- AWS Multi-AZ infrastructure

Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Product versions

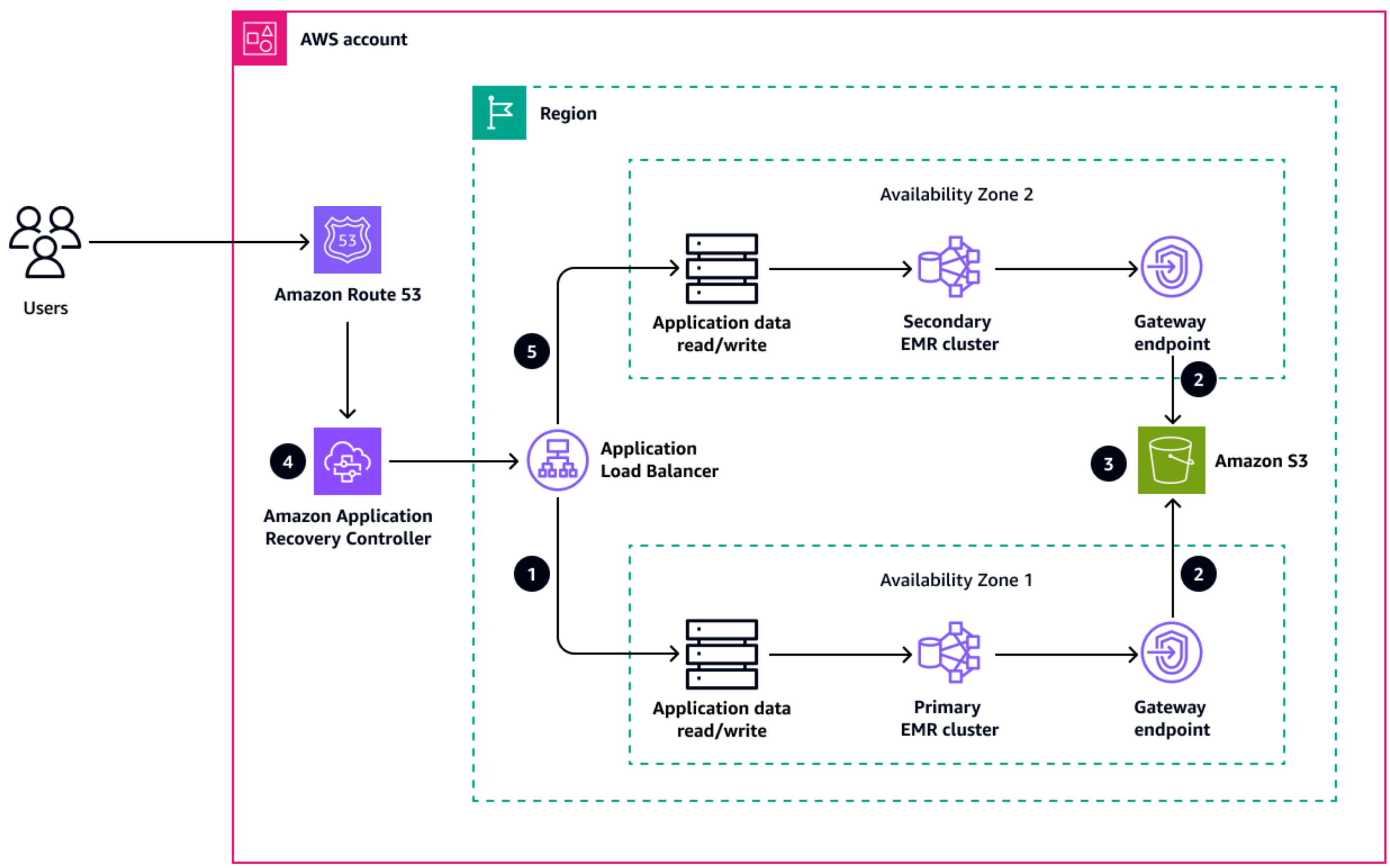
- [Amazon EMR 6.x and later releases]

Architecture

Target technology stack

- Amazon EMR cluster
- Amazon Application Recovery Controller
- Application Load Balancer
- Amazon S3 bucket
- Gateway endpoints for Amazon S3

Target architecture



This architecture provides application resilience by using multiple Availability Zones and implementing an automated recovery mechanism through the Application Recovery Controller.

1. The Application Load Balancer routes traffic to the active Amazon EMR environment, which is typically the primary EMR cluster in the primary Availability Zone.
2. The active EMR cluster processes the application requests and connects to Amazon S3 through its dedicated Amazon S3 gateway endpoint for read and write operations.
3. Amazon S3 serves as a central data repository and is potentially used as a checkpoint or as shared storage between EMR clusters. EMR clusters maintain data consistency when they write directly to Amazon S3 through the `s3://` protocol and the EMR File System (EMRFS).
4. Application Recovery Controller continuously monitors the health of the primary Availability Zone and automatically manages failover operations when necessary.
5. If the Application Recovery Controller detects a failure in the primary EMR cluster, it takes these actions:
    - Initiates the failover process to the secondary EMR cluster in Availability Zone 2.
    - Updates routing configurations to direct traffic to the secondary cluster.

Tools

AWS services

- Amazon Application Recovery Controller helps you manage and coordinate the recovery of your applications across AWS Regions and Availability Zones. This service simplifies the process and improves the reliability of application recovery by reducing the manual steps required by traditional tools and processes.
- Application Load Balancer operates at the application layer, which is the seventh layer of the Open Systems Interconnection (OSI) model. It distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. This increases the availability of your application.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command line shell.
- Amazon EMR is a big data platform that provides data processing, interactive analysis, and machine learning for open source frameworks such as Apache Spark, Apache Hive, and Presto.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon S3 provides a simple web service interface that you can use to store and retrieve any amount of data, at any time, from anywhere. Using this service, you can easily build applications that make use of cloud native storage.

- Gateway endpoints for Amazon S3 are gateways that you specify in your route table to access Amazon S3 from your virtual private cloud (VPC) over the AWS network.

Best practices

- Follow AWS best practices for security, identity, and compliance to ensure a robust and secure architecture.
- Align the architecture with the AWS Well-Architected Framework.
- Use Amazon S3 Access Grants to manage access from your Spark-based EMR cluster to Amazon S3. For details, see the blog post Use Amazon EMR with S3 Access Grants to Scale Spark access to Amazon S3.
- Improve Spark performance with Amazon S3.

Epics

Set up your environment

Task	Description	Skills required
Sign in to the AWS Management Console.	Sign in to the AWS Management Console as an IAM user. For instructions, see the AWS documentation.	AWS DevOps
Configure the AWS CLI.	Install the AWS CLI or update it to the latest version so you can interact with AWS services in the AWS Management Console. For instructions, see the AWS CLI documentation.	AWS DevOps

Deploy a Spark application on your EMR cluster

Task	Description	Skills required
Create an S3 bucket.	1. Create an S3 bucket to store the input dataset, logs, application, and output data. For instructions, see the [Amazon S3 documentation](). 2. Organize the bucket into separate folders for input data (`dataset`), logs (`logs`), Spark application (`spark-app`), and output data (`output`).	AWS DevOps
Create an EMR cluster.	1. Use the following AWS CLI commands to create an EMR cluster (for example, version 6.12 or later) with instances that span two Availability Zones (such as `us-east-1a` and `us-east-1b`) for high availability. The command specifies the `m4.large` instance type as an example. `aws emr create-cluster \ --ec2-attributes AvailabilityZone=<AZ-name-1> \ --release-label emr-6.12.0 \`	AWS DevOps

Task	Description	Skills required

	``
	    --instance-groups
	  InstanceGroupType=
	MASTER,InstanceCou
	nt=1,InstanceType=
	m4.large InstanceG
	roupType=CORE,Inst
	anceCount=2,Instan
	ceType=m4.large
	``
	``
	aws emr create-cl
	uster \
	    --ec2-attributes
	  AvailabilityZone=<
	AZ-name-2> \
	    --release-label
	  emr-6.12.0 \
	    --instance-groups
	  InstanceGroupType=
	MASTER,InstanceCou
	nt=1,InstanceType=
	m4.large InstanceG
	roupType=CORE,Inst
	anceCount=2,Instan
	ceType=m4.large
	``
	For more information, see the create-cluster command and the Amazon EMR documentation.
	2. Provide the key pair, service role, and instance profile with the required permissions, as necessary.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Configure security settings for the EMR cluster.</td><td>1. Identify the security group associated with the EMR cluster's master node by using the AWS CLI describe-cluster command:
<pre><code>aws emr describe-cluster --cluster-id j-XXXXXXXX</code></pre>2. To enhance security, modify the security group settings to permit SSH access (TCP port 22) to the master node, but restrict it to your specific IP address.

For more information, see the Amazon EMR documentation.</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required

Connect to the EMR cluster.	Connect to the master node of the EMR cluster through SSH by using the provided key pair.
	
	Ensure that the key pair file is present in the same directory as your application.
	
	Run the following commands to set the correct permissions for the key pair and to establish the SSH connection:
	
	`chmod 400 <key-pair-name>`
	`ssh -i ./<key-pair-name> hadoop@<master-node-public-dns>`
		AWS DevOps

Task	Description	Skills required

Deploy the Spark application.	After you establish the SSH connection, you will be in the Hadoop console.
	
	1. Create or edit the Spark application file (`main.py`) by using a text editor such as vim:
	
	`vim main.py`
	
	For more informati on about creating and modifying the Spark application, see the [Amazon EMR documenta tion](#).
	2. Submit the Spark applicati on to the EMR cluster, specifying the input data and output data locations in the S3 bucket:
	
	`spark-submit main.py —data_source <input-data-folder-in-s3> —output_uri <output-folder-in-s3>`
	
	Here's an example based on the folders you set up earlier:
		AWS DevOps

Task	Description	Skills required

	`spark-submit main.py —data_source dataset —output_uri output`
	3. Monitor the application's progress by checking the application logs:
	`yarn logs -applicationId <application-id>`	

Monitor the Spark application.	1. Open another terminal window and establish an SSH tunnel to the EMR cluster's resource manager web UI: `ssh -i <key-pair-name> -N -L 8157:<resource-manager-public-dns>:8088 hadoop@<resource-manager-public-dns>` 2. To monitor the application, access the resource manager web UI by navigating to `http://localhost:8157` in your web browser.	AWS DevOps

Shift traffic to another Availability Zone

Task	Description	Skills required
Create an Application Load Balancer.	Set up the target group that routes traffic between Amazon EMR master nodes that are deployed across two Availability Zones within an AWS Region. For instructions, see Create a target group for your Application Load Balancer in the Elastic Load Balancing documentation.	AWS DevOps
Configure zonal shift in Application Recovery Controller.	In this step, you'll use the zonal shift feature in Applicati on Recovery Controller to shift traffic to another Availability Zone. 1. Open the Application Recovery Controller console. 2. Under **Getting started**, choose **Zonal shift**, **Start zonal shift**. 3. Select the Availability Zone that you want to shift traffic away from. 4. Select a supported resource (for example, Application Load Balancer) for the zonal shift from the Resources table.	AWS DevOps

Task	Description	Skills required
	5. For **Set zonal shift expiration**, choose or enter an expiration for the zonal shift. You can set a duration between 1 minute and three days (72 hours). All zonal shifts are temporary. You must set an expiration, but you can update active shifts later to set a new expiration period of up to three days. 6. Enter a comment about this zonal shift. 7. Select the check box to acknowledge that starting a zonal shift will reduce available capacity for your application by shifting traffic away from the Availability Zone. 8. Choose **Start**. To use the AWS CLI, see [Examples of using the AWS CLI with zonal shift] in the Application Recovery Controller documentation.	

Task	Description	Skills required

Verify zonal shift configura tion and progress.	1. Verify the resources that are registered with zonal shift:
	`aws arc-zonal-shift list-managed-resou rces --region <AWS-region-name>`
	For example, the following output confirms that the resources are up and running in both Availability Zones.
	`"appliedWeights": { "use1-az1": 1.0, "use1-az2": 1.0 },`
	2. To visualize the zonal shift, use the following AWS CLI command to start the zonal shift:
	`aws arc-zonal-shift start-zonal-shift \ --resource-identif ier <application-load-balancer-arn> \ --away-from <source-AZ> \ --expires-in 10m --comment "testing" \ --region <AWS-regi on-name>`
		AWS DevOps

Task	Description	Skills required

	where `<source-AZ>` is the identifier of the Availability Zone you want to shift traffic away from, and `<application-load-balancer-arn>` is the Amazon Resource Name (ARN) of your Application Load Balancer.
	3. Verify that the traffic has shifted to another Availability Zone.
	`aws arc-zonal-shift get-managed-resource \ --resource-identifier <application-load-balancer-arn> \ --region <AWS-region-name>`
	You can see the zonal shift confirmed by these weights:
	`"appliedWeights": { "use1-az1": 0.0, "use1-az2": 1.0 },`	

Related resources

- AWS CLI commands:
  - create-cluster
  - describe-cluster
  - arc-zonal-shift
- Configuring Amazon EMR cluster instance types and best practices for Spot instances (Amazon EMR documentation)
- Security best practices in IAM (IAM documentation)
- Use instance profiles (IAM documentation)
- Use zonal shift and zonal autoshift to recovery applications in ARC (Application Recovery Controller documentation)

Manage blue/green deployments of microservices to multiple accounts and Regions by using AWS code services and AWS KMS multi-Region keys

Balaji Vedagiri, Vanitha Dontireddy, Ashish Kumar, Faisal Shahdad, Vivek Thangamuthu, and Anand Krishna Varanasi, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern describes how to deploy a global microservices application from a central AWS account to multiple workload accounts and Regions in accordance with a blue/green deployment strategy. The pattern supports the following:

- Software is developed in a central account, whereas workloads and applications are spread across multiple accounts and AWS Regions.
- A single AWS Key Management System (AWS KMS) multi-Region key is used for encryption and decryption to cover disaster recovery.
- The KMS key is Region-specific and has to be maintained or created in three different Regions for pipeline artifacts. A KMS multi-Region key helps retain the same key ID across Regions.
- The Git workflow branching model is implemented with two branches (development and main) and code is merged by using pull requests (PRs). The AWS Lambda function that is deployed from this stack creates a PR from the development branch to the main branch. The PR merge to the main branch initiates an AWS CodePipeline pipeline, which orchestrates the continuous integration and continuous delivery (CI/CD) flow and deploys the stacks across accounts.

This pattern provides a sample infrastructure as code (IaC) setup through AWS CloudFormation stacks to demonstrate this use case. The blue/green deployment of microservices is implemented by using AWS CodeDeploy.

Prerequisites and limitations

Prerequisites

- Four active AWS accounts:

    - A tools account to manage the code pipeline and maintain the AWS CodeCommit repository.
    - Three workload (test) accounts for deploying the microservices workload.
- This pattern uses the following Regions. If you want to use other Regions, you must make the appropriate modifications to the AWS CodeDeploy and AWS KMS multi-Region stacks.
    - Tools (AWS CodeCommit) account: `ap-south-1`
    - Workload (test) account 1: `ap-south-1`
    - Workload (test) account 2: `eu-central-1`
    - Workload (test) account 3: `us-east-1`
- Three Amazon Simple Storage Service (Amazon S3) buckets for the deployment Regions in each workload account. (These are called `S3BUCKETNAMETESTACCOUNT1`, `S3BUCKETNAMETESTACCOUNT2` and `S3BUCKETNAMETESTACCOUNT3` later in this pattern.)

  For example, you can create these buckets in specific accounts and Regions with unique bucket names as follows (replace *xxxx* with a random number):

```
##In Test Account 1
aws s3 mb s3://ecs-codepipeline-xxxx-ap-south-1 --region ap-south-1
##In Test Account 2
aws s3 mb s3://ecs-codepipeline-xxxx-eu-central-1 --region eu-central-1
##In Test Account 3
aws s3 mb s3://ecs-codepipeline-xxxx-us-east-1 --region us-east-1

#Example
##In Test Account 1
aws s3 mb s3://ecs-codepipeline-18903-ap-south-1 --region ap-south-1
##In Test Account 2
aws s3 mb s3://ecs-codepipeline-18903-eu-central-1 --region eu-central-1
##In Test Account 3
aws s3 mb s3://ecs-codepipeline-18903-us-east-1 --region us-east-1
```

Limitations

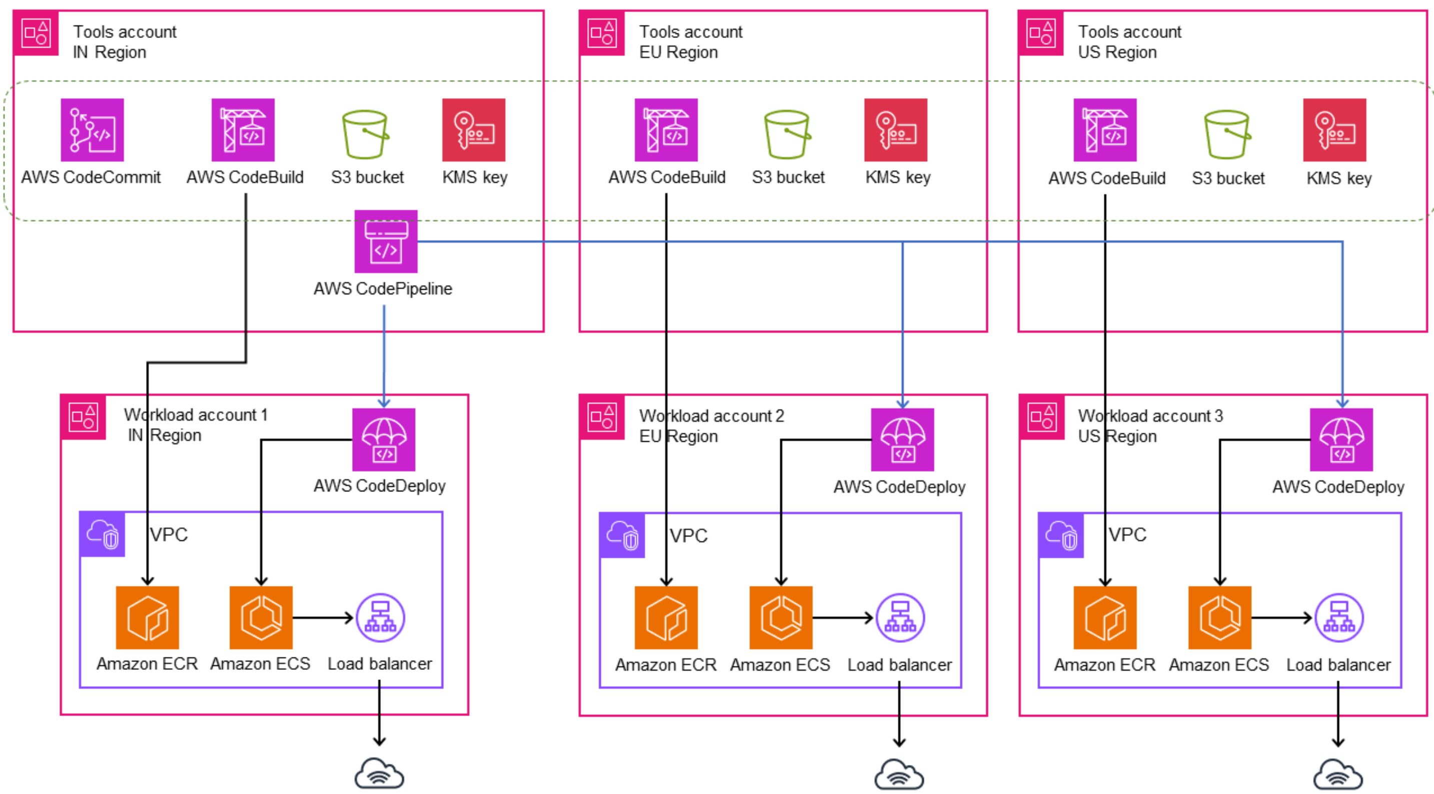
The pattern uses AWS CodeBuild and other configuration files to deploy a sample microservice. If you have a different workload type (for example, serverless), you must update all relevant configurations.

Architecture

Target technology stack

- AWS CloudFormation
- AWS CodeCommit
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

Target architecture



Automation and scale

The setup is automated by using AWS CloudFormation stack templates (IaC). It can be easily scaled for multiple environments and accounts.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS CodeDeploy automates deployments to Amazon Elastic Compute Cloud (Amazon EC2) or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Additional tools

- Git is an open-source, distributed version control system that works with the AWS CodeCommit repository.
- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers. This pattern uses Docker to build and test container images locally.
- cfn-lint and cfn-nag are open-source tools that help you review CloudFormation stacks for any errors and security issues.

Code repository

The code for this pattern is available in the GitHub Global Blue/Green deployments in multiple regions and accounts repository.

Epics

Set up environment variables

Task	Description	Skills required

Export environment variables for CloudFormation stack deployment.	Define environment variables that will be used as input to the CloudFormation stacks later in this pattern.
	
	1. Update the bucket names that you created in the three accounts and Regions as explained earlier in the Prerequisites section:
	
	`export S3BUCKETNAMETESTACCOUNT1=<S3BUCKETACCOUNT1>`
	`export S3BUCKETNAMETESTACCOUNT2=<S3BUCKETACCOUNT2>`
	`export S3BUCKETNAMETESTACCOUNT3=<S3BUCKETACCOUNT3>`
	
	2. Define a random string to create artifact buckets, because bucket names must be unique globally:
	
	`export BUCKETSTARTNAME=ecs-codepipeline-artifacts-19992`
	
	3. Define and export the account IDs and Regions:
		AWS DevOps

Task	Description	Skills required
	``` export TOOLSACCO UNT=<TOOLSACCOUNT> export CODECOMMI TACCOUNT=<CODECOMM ITACCOUNT> export CODECOMMI TREGION=ap-south-1 export CODECOMMI TREPONAME=Poc export TESTACCOU NT1=<TESTACCOUNT1> export TESTACCOU NT2=<TESTACCOUNT2> export TESTACCOU NT3=<TESTACCOUNT3> export TESTACCOU NT1REGION=ap-south -1 export TESTACCOU NT2REGION=eu-centr al-1 export TESTACCOU NT3REGION=us-east-1 export TOOLSACCO UNTREGION=ap-south -1 export ECRREPOSI TORYNAME=web ```	

Package and deploy the CloudFormation stacks for the infrastructure

Task	Description	Skills required
Clone the repository.	Clone the [sample repository](#) into a new repository in your work location: ``` ##In work location ```	AWS DevOps

Task	Description	Skills required
	`git clone https://github.com/aws-samples/ecs-blue-green-global-deployment-with-multiregion-cmk-codepipeline.git`	

Task	Description	Skills required

Package the Cloudformation resources.	In this step, you package the local artifacts that the CloudFormation templates reference to create the infrastructure resources required for services such as Amazon Virtual Private Cloud (Amazon VPC) and Application Load Balancer.
	
	The templates are available in the `Infra` folder of the code repository.
	
	`##In TestAccount1## aws cloudformation package \ --template-file mainInfraStack.yaml \ --s3-bucket $S3BUCKETNAMETESTACCOUNT1 \ --s3-prefix infraStack \ --region $TESTACCOUNT1REGION \ --output-template-file infrastructure_${TESTACCOUNT1}.template`
	
	`##In TestAccount2## aws cloudformation package \ --template-file mainInfraStack.yaml \`
		AWS DevOps

Task	Description	Skills required

	``
	    --s3-bucket
	 $S3BUCKETNAMETESTA
	CCOUNT2 \
	    --s3-prefix
	 infraStack \
	    --region $TESTACCO
	UNT2REGION \
	    --output-template-
	file infrastructure_
	${TESTACCOUNT2}.templ
	ate
	``
	``
	##In TestAccount3##
	aws cloudformation
	 package \
	    --template-file
	 mainInfraStack.yaml \
	    --s3-bucket
	 $S3BUCKETNAMETESTA
	CCOUNT3 \
	    --s3-prefix
	 infraStack \
	    --region $TESTACCO
	UNT3REGION \
	    --output-template-
	file infrastructure_
	${TESTACCOUNT3}.templ
	ate
	``	

Task	Description	Skills required

Validate the package templates.	Validate the package templates:
	`aws cloudformation validate-template \ --template-body file://infrastructure_${TESTACCOUNT1}.template`
	
	`aws cloudformation validate-template \ --template-body file://infrastructure_${TESTACCOUNT2}.template`
	
	`aws cloudformation validate-template \ --template-body file://infrastructure_${TESTACCOUNT3}.template`
		AWS DevOps

Task	Description	Skills required

Deploy the package files into the workload accounts,	1. Update the placeholder values and account names in the `infraParameters.json` script based on your setup.
	2. Deploy the package templates into your three workload accounts.
	<pre>##In TestAccount1##
	aws cloudformation
	 deploy \
	    --template-file
	 infrastructure_${T
	ESTACCOUNT1}.templ
	ate \
	    --stack-name
	 mainInfrastack \
	    --parameter-
	overrides file://in
	fraParameters.json \
	    --region $TESTACCO
	UNT1REGION \
	    --capabilities
	 CAPABILITY_IAM
	 CAPABILITY_NAMED_I
	AM
	
	##In TestAccount2##
	aws cloudformation
	 deploy \
	    --template-file
	 infrastructure_${T
	ESTACCOUNT2}.templ
	ate \
	    --stack-name
	 mainInfrastack \</pre>
		AWS DevOps

Task	Description	Skills required

	`--parameter-overrides file://infraParameters.json \ --region $TESTACCOUNT2REGION \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM`
	
	`##In TestAccount3## aws cloudformation deploy \ --template-file infrastructure_${TESTACCOUNT3}.template \ --stack-name mainInfrastack \ --parameter-overrides file://infraParameters.json \ --region $TESTACCOUNT3REGION \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM`	

Push a sample image and scale Amazon ECS

Task	Description	Skills required
Push a sample image to the Amazon ECR repository.	Push a sample (NGINX) image to the Amazon Elastic Container Registry (Amazon ECR) repository named web (as set in parameters). You	AWS DevOps

Task	Description	Skills required

	can customize the image as required.
	
	To log in and set the credentials for pushing an image to Amazon ECR, follow the instructions in the [Amazon ECR documentation](#).
	
	The commands are:
	
	`docker pull nginx`
	`docker images`
	`docker tag <imageid> aws_account_id.dkr.ecr.region.amazonaws.com/<web>:latest`
	`docker push <aws_account_id>.dkr.ecr.<region>.amazonaws.com/<web>:tag`	

Task	Description	Skills required
Scale Amazon ECS and verify access.	1. Scale Amazon ECS to create two replicas: `aws ecs update-service --cluster QA-Cluster --service Poc-Service --desired-count 2` where `Poc-Service` refers to your sample application. 2. Verify that the services are accessible from the Application Load Balancer by using a fully qualified domain name (FQDN) or DNS from a browser or by using the **curl** command.	AWS DevOps

Set up code services and resources

Task	Description	Skills required
Create a CodeCommit repository in the tools account.	Create a CodeCommit repository in the tools account by using the `codecommit.yaml` template, which is in the `code` folder of the GitHub repository. You must create this repository only in the single Region where you plan to develop the code.	AWS DevOps

Task	Description	Skills required
	```aws cloudformation deploy --stack-name codecommitrepoStack --parameter-overrides CodeCommitReponame=$CODECOMMITREPONAME \ ToolsAccount=$TOOLSACCOUNT --template-file codecommit.yaml --region $TOOLSACCOUNTREGION \ --capabilities CAPABILITY_NAMED_IAM```	

Task	Description	Skills required
Create an S3 bucket for managing artifacts generated by CodePipeline.	Create an S3 bucket for managing artifacts generated by CodePipeline by using the `pre-reqs-bucket.yaml` template, which is in the `code` folder of the GitHub repository. The stacks must be deployed in all three workload (test) and tools accounts and Regions. `aws cloudformation deploy --stack-name pre-reqs-artifacts-bucket --parameter-overrides BucketStartName=$BUCKETSTARTNAME \ TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT \ --template-file pre-reqs_bucket.yaml --region $TESTACCOUNT1REGION --capabilities CAPABILITY_NAMED_IAM`  `aws cloudformation deploy --stack-name pre-reqs-artifacts-bucket --parameter-overrides BucketSta`	AWS DevOps

Task	Description	Skills required
	``` rtName=$BUCKETSTAR TNAME \ TestAccount1=$TE STACCOUNT1 TestAccou nt2=$TESTACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TESTACCO UNT2REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts -bucket --parameter- overrides BucketSta rtName=$BUCKETSTAR TNAME \ TestAccount1=$TE STACCOUNT1 TestAccou nt2=$TESTACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TESTACCO UNT3REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts ```	

Task	Description	Skills required
	```-bucket --parameter-overrides BucketStartName=$BUCKETSTARTNAME \ TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT \ --template-file pre-reqs_bucket.yaml --region $TOOLSACCOUNTREGION --capabilities CAPABILITY_NAMED_IAM```	

Task	Description	Skills required
Set up a multi-Region KMS key.	1. Create a multi-Region KMS key with primary and replica keys that CodePipeline will use. In our example, `ToolsAccount1region - ap-south-1` will be the primary Region. `aws cloudformation deploy --stack-name ecs-codepipeline-pre-reqs-KMS \ --template-file pre-reqs_KMS.yaml --parameter-overrides \ TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT --region $TOOLSACCOUNTREGION` 2. Set the CMKARN variables to pass to CodeBuild projects. The values are available in the output of the ecs-codepipeline-pre-reqs-KMS template stack (the key ID will be same in all Regions and starts with `mrk-`). Or, you can get the CMKARN values from the	AWS DevOps

Task	Description	Skills required

	tools account. Export them in all account sessions:
	`export CMKARN1=arn:aws:kms:ap-south-1:<TOOLSACCOUNTID>:key/mrk-xxx`
	`export CMKARN2=arn:aws:kms:eu-central-1:<TOOLSACCOUNTID>:key/mrk-xxx`
	`export CMKARN3=arn:aws:kms:us-east-1:<TOOLSACCOUNTID>:key/mrk-xxx`
	`export CMARNTOOLS=arn:aws:kms:ap-south-1:<TOOLSACCOUNTID>:key/mrk-xxx`	

Task	Description	Skills required

Set up the CodeBuild project in the tools account.	1. Use the `codebuild _IAM.yaml` template from the code folder of the GitHub repository to set up AWS Identity and Access Management (IAM) for AWS CodeBuild in a single Region in the tools account:
	```#In ToolsAccount aws cloudformation deploy --stack-name ecs-codebuild-iam \ --template-file codebuild_IAM.yaml --region $TOOLSACCOUNTREGION \ --capabilities CAPABILITY_NAMED_IAM```
	2. Use the `codebuild .yaml` template to set up CodeBuild for your build project. Deploy this template in all three Regions as follows:
	```aws cloudformation deploy --stack-name ecscodebuildstack --parameter-overrides ToolsAccount=$TOOLSACCOUNT \ CodeCommitRepoName=$CODECOMMITREPONAME ECRRepositoryName=```
		AWS DevOps

Task	Description	Skills required
	``` $ECRREPOSITORYNAME APPACCOUNTID=$TEST ACCOUNT1 \ TestAccount3=$TE STACCOUNT3 CodeCommi tRegion=$CODECOMMI TREGION CMKARN=$C MKARN1 \ --template-file codebuild.yaml --region $TESTACCO UNT1REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name ecscodebuildstack -- parameter-overrides ToolsAccount=$TOOL SACCOUNT \ CodeCommitRepoName= $CODECOMMITREPONAME ECRRepositoryName= $ECRREPOSITORYNAME APPACCOUNTID=$TEST ACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tRegion=$CODECOMMI TREGION CMKARN=$C MKARN2 \ --template-file codebuild.yaml --region $TESTACCO UNT2REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name ecscodebuildstack -- ```	

Task	Description	Skills required
	```parameter-overrides ToolsAccount=$TOOLSACCOUNT \ CodeCommitRepoName=$CODECOMMITREPONAME ECRRepositoryName=$ECRREPOSITORYNAME APPACCOUNTID=$TESTACCOUNT3 \ CodeCommitRegion=$CODECOMMITREGION CMKARN=$CMKARN3 \ --template-file codebuild.yaml --region $TESTACCOUNT3REGION --capabilities CAPABILITY_NAMED_IAM```	

Task	Description	Skills required

Set up CodeDeploy in workload accounts.	Use the `codedeploy.yaml` template in the `code` folder of the GitHub repository to set up CodeDeploy in all three workload accounts. The output of `mainInfraStack` includes the Amazon Resource Names (ARNs) of the Amazon ECS cluster and Application Load Balancer listener.
	
	Note
	The values from the infrastructure stacks are exported already, so they are imported by the CodeDeploy stack templates.
	
	`##WorkloadAccount1##`
	`aws cloudformation deploy --stack-name ecscodedeploystack \`
	`--parameter-overrides ToolsAccount=$TOOLSACCOUNT mainInfrastackname=mainInfrastack \`
	`--template-file codedeploy.yaml --region $TESTACCOUNT1REGION --capabilities CAPABILITY_NAMED_IAM`
		AWS DevOps

Task	Description	Skills required
| | ```
##WorkloadAccount2##
aws cloudformation
 deploy --stack-name
 ecscodedeploystack \
--parameter-overrides
 ToolsAccount=$TOOL
SACCOUNT mainInfra
stackname=mainInfr
astack \
--template-file
 codedeploy.yaml
 --region $TESTACCO
UNT2REGION --capabil
ities CAPABILIT
Y_NAMED_IAM

##WorkloadAccount3##
aws cloudformation
 deploy --stack-name
 ecscodedeploystack \
--parameter-overrides
 ToolsAccount=$TOOL
SACCOUNT mainInfra
stackname=mainInfr
astack \
--template-file
 codedeploy.yaml
 --region $TESTACCO
UNT3REGION --capabil
ities CAPABILIT
Y_NAMED_IAM
``` | |

Set up CodePipeline in the tools account

Task	Description	Skills required
Create a code pipeline in the tools account.	In the tools account, run the command:	AWS DevOps
```

Task	Description	Skills required

	<pre>aws cloudformation
	 deploy --stack-name
	 ecscodepipelinestack
	 --parameter-overrides
	  \
	TestAccount1=$TE
	STACCOUNT1 TestAccou
	nt1Region=$TESTACC
	OUNT1REGION \
	TestAccount2=$TE
	STACCOUNT2 TestAccou
	nt2Region=$TESTACC
	OUNT2REGION \
	TestAccount3=$TE
	STACCOUNT3 TestAccou
	nt3Region=$TESTACC
	OUNT3REGION \
	CMKARNTools=$CMK
	TROOLSARN CMKARN1=
	$CMKARN1 CMKARN2=$
	CMKARN2 CMKARN3=$
	CMKARN3 \
	CodeCommitRepoName=
	$CODECOMMITREPONAME
	 BucketStartName=$B
	UCKETSTARTNAME \
	--template-file
	 codepipeline.yaml --
	capabilities CAPABILIT
	Y_NAMED_IAM</pre>	

Task	Description	Skills required
Provide access for CodePipeline and CodeBuild roles in the AWS KMS key policy and S3 bucket policy.	1. Provide access for CodePipeline and CodeBuild roles in the AWS KMS key policy: `aws cloudformation deploy --stack-name ecs-codepipeline-pre-reqs-KMS \ --template-file pre-reqs_KMS.yaml --parameter-overrides \ CodeBuildCondition=true TestAccount1=$TESTACCOUNT1 TestAccount2=$TESTACCOUNT2 \ TestAccount3=$TESTACCOUNT3 CodeCommitAccount=$CODECOMMITACCOUNT ToolsAccount=$TOOLSACCOUNT --region $TOOLSACCOUNTREGION` 2. Update the S3 bucket policy to allow access for CodePipeline and CodeDeploy roles: `aws cloudformation deploy --stack-name pre-reqs-artifacts-bucket --parameter-overrides BucketStartName=$BUCKETSTARTNAME \ PutS3BucketPolicy=true TestAccou`	AWS DevOps

Task	Description	Skills required

	<pre>nt1=$TESTACCOUNT1
	 TestAccount2=$TEST
	ACCOUNT2 \
	TestAccount3=$TE
	STACCOUNT3 CodeCommi
	tAccount=$CODECOMM
	ITACCOUNT ToolsAcco
	unt=$TOOLSACCOUNT \
	--template-file pre-
	reqs_bucket.yaml
	 --region $TESTACCO
	UNT1REGION --capabil
	ities CAPABILIT
	Y_NAMED_IAM
	
	aws cloudformation
	 deploy --stack-name
	 pre-reqs-artifacts
	-bucket --parameter-
	overrides BucketSta
	rtName=$BUCKETSTAR
	TNAME \
	PutS3BucketPolic
	y=true TestAccou
	nt1=$TESTACCOUNT1
	 TestAccount2=$TEST
	ACCOUNT2 \
	TestAccount3=$TE
	STACCOUNT3 CodeCommi
	tAccount=$CODECOMM
	ITACCOUNT ToolsAcco
	unt=$TOOLSACCOUNT \
	--template-file pre-
	reqs_bucket.yaml
	 --region $TESTACCO
	UNT2REGION --capabil
	ities CAPABILIT
	Y_NAMED_IAM
	
	aws cloudformation
	 deploy --stack-name
	 pre-reqs-artifacts</pre>	

Task	Description	Skills required
	``` -bucket --parameter- overrides BucketSta rtName=$BUCKETSTAR TNAME \ PutS3BucketPolic y=true TestAccou nt1=$TESTACCOUNT1 TestAccount2=$TEST ACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TESTACCO UNT3REGION --capabil ities CAPABILIT Y_NAMED_IAM aws cloudformation deploy --stack-name pre-reqs-artifacts -bucket --parameter- overrides BucketSta rtName=$BUCKETSTAR TNAME \ PutS3BucketPolic y=true TestAccou nt1=$TESTACCOUNT1 TestAccount2=$TEST ACCOUNT2 \ TestAccount3=$TE STACCOUNT3 CodeCommi tAccount=$CODECOMM ITACCOUNT ToolsAcco unt=$TOOLSACCOUNT \ --template-file pre- reqs_bucket.yaml --region $TOOLSACC OUNTREGION --capabil ```	

Task	Description	Skills required
	`ities CAPABILITY_NAMED_IAM`	

Call and test the pipeline

Task	Description	Skills required
Push changes to the CodeCommit repository.	1. Clone the CodeCommit repository that was created in the `codecommitrepoStack` by using the `git clone` command, as described in the [AWS CodeCommit documentation](). 2. Update the input artifacts with the required details: • JSON file: Update `AccountID` in the file in three places of this file. Rename the three files to include the account IDs. • YAML files: Update the task definition ARN and version. Rename the three files to include the account IDs. 3. Modify the `index.html` file to make a few minor changes to the home page. 4. Copy the following files to the repository and commit:	

Task	Description	Skills required
	``` index.html Dockerfile buildspec.yaml appspec_<accountid>.yaml (3 files - one per account) taskdef<accountid>.json (3 files - one per account) ``` 5. Start or restart the pipeline and verify the results. 6. Access the service from the Application Load Balancer using by using an FQDN or DNS, and verify that the updates have been deployed.	

Clean up

Task	Description	Skills required
Clean up all the deployed resources.	1. Scale down Amazon ECS to zero instances: ``` aws ecs update-service --cluster QA-Cluster --service Poc-Service --desired-count 0 ``` 2. Delete the CloudFormation stacks in each account and Region:	

Task	Description	Skills required
	```##In Tools Account## aws cloudformation delete-stack --stack-name ecscodepipelinestack --region $TOOLSACCOUNTREGION aws cloudformation delete-stack --stack-name ecscodebuildstack --region $TESTACCOUNT1REGION aws cloudformation delete-stack --stack-name ecscodebuildstack --region $TESTACCOUNT2REGION aws cloudformation delete-stack --stack-name ecscodebuildstack --region $TESTACCOUNT3REGION aws cloudformation delete-stack --stack-name ecs-codepipeline-pre-reqs-KMS --region $TOOLSACCOUNTREGION aws cloudformation delete-stack --stack-name codecommitrepoStack --region $TOOLSACCOUNTREGION aws cloudformation delete-stack --stack-name pre-reqs-artifacts-bucket --region $TESTACCOUNT1REGION aws cloudformation delete-stack --stack-name pre-reqs-```	

Task	Description	Skills required
	``` artifacts-bucket --region $TESTACCO UNT2REGION aws cloudformation delete-stack -- stack-name pre-reqs- artifacts-bucket --region $TESTACCO UNT3REGION aws cloudformation delete-stack -- stack-name pre-reqs- artifacts-bucket --region $TOOLSACC OUNTREGION aws cloudformation delete-stack -- stack-name ecs-codeb uild-iam --region $TOOLSACCOUNTREGION ##NOTE: Artifact buckets will not get deleted if there are artifacts so it has to be emptied manually before deleting.## ``` ``` ##In Workload / Test Accounts## ##Account:1## aws cloudformation delete-stack -- stack-name ecscodede ploystack --region $TESTACCOUNT1REGION aws cloudformation delete-stack -- stack-name mainInfra ```	

Task	Description	Skills required
	``` stack --region $TESTACCOUNT1REGION ##Account:2## aws cloudformation delete-stack -- stack-name ecscodede ploystack --region $TESTACCOUNT2REGION aws cloudformation delete-stack -- stack-name mainInfra stack --region $TESTACCOUNT2REGION ##Account:3## aws cloudformation delete-stack -- stack-name ecscodede ploystack --region $TESTACCOUNT3REGION aws cloudformation delete-stack -- stack-name mainInfra stack --region $TESTACCOUNT3REGION ##NOTE: Amazon ECR (web) will not get deleted if the registry still includes images. It can be manually cleaned up if not required. ```	

Troubleshooting

Issue	Solution
Changes that you committed to the repository aren't getting deployed.	• Check the CodeBuild logs for errors in the Docker build action. For more information, see the CodeBuild documentation. • Check the CodeDeploy deployment for any Amazon ECS deployment issues.

Related resources

- Pushing a Docker image (Amazon ECR documentation)
- Connect to an AWS CodeCommit repository (AWS CodeCommit documentation)
- Troubleshooting AWS CodeBuild (AWS CodeBuild documentation)

Monitor Amazon ECR repositories for wildcard permissions using AWS CloudFormation and AWS Config

Vikrant Telkar, Wassim Benhallam, and Sajid Momin, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that supports private repositories with resource-based permissions using AWS Identity and Access Management (IAM).

IAM supports the "*" wildcard in both the resource and action attributes, which makes it easier to automatically choose multiple matching items. In your testing environment, you can allow all authenticated AWS users to access an Amazon ECR repository by using the `ecr:*` wildcard permission in a principal element for your repository policy statement. The `ecr:*` wildcard permission can be useful when developing and testing in development accounts that can't access your production data.

However, you must make sure that the `ecr:*` wildcard permission is not used in your production environments because it can cause serious security vulnerabilities. This pattern's approach helps you to identify Amazon ECR repositories that contain the `ecr:*` wildcard permission in repository policy statements. The pattern provides steps and an AWS CloudFormation template to create a custom rule in AWS Config. An AWS Lambda function then monitors your Amazon ECR repository policy statements for `ecr:*` wildcard permissions. If it finds non-compliant repository policy statements, Lambda notifies AWS Config to send an event to Amazon EventBridge and EventBridge then initiates an Amazon Simple Notification Service (Amazon SNS) topic. The SNS topic notifies you by email about the non-compliant repository policy statements.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
- An existing Amazon ECR repository with an attached policy statement, installed and configured in your testing environment. For more information about this, see Creating a private repository and Setting a repository policy statement in the Amazon ECR documentation.

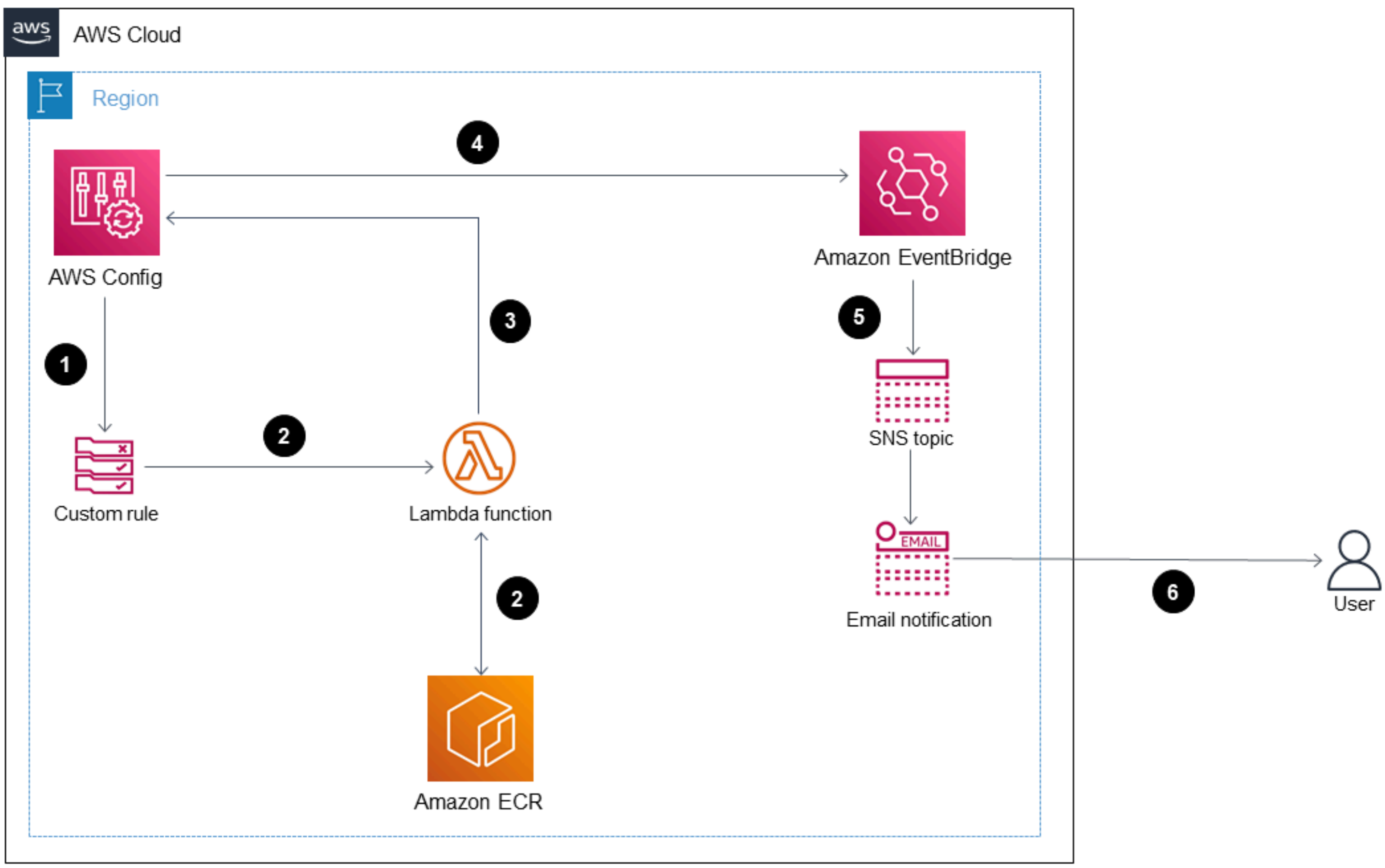
- AWS Config, configured in your preferred AWS Region. For more information about this, see [Getting started with AWS Config](#) in the AWS Config documentation.
- The `aws-config-cloudformation.template` file (attached), downloaded to your local machine.

Limitations

- This pattern's solution is Regional and your resources must be created in the same Region.

Architecture

The following diagram shows how AWS Config evaluates Amazon ECR repository policy statements.



The diagram shows the following workflow:

1. AWS Config initiates a custom rule.
2. The custom rule invokes a Lambda function to evaluate the compliance of the Amazon ECR repository policy statements. The Lambda function then identifies non-compliant repository policy statements.
3. The Lambda function sends the non-compliance status to AWS Config.
4. AWS Config sends an event to EventBridge.
5. EventBridge publishes the non-compliance notifications to an SNS topic.
6. Amazon SNS sends an email alert to you or an authorized user.

Automation and scale

This pattern's solution can monitor any number of Amazon ECR repository policy statements, but all resources that you want to evaluate must be created in the same Region.

Tools

- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- AWS Config – AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable. Amazon ECR supports private repositories with resource-based permissions using IAM.
- Amazon EventBridge – Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your applications, software as a service (SaaS) applications, and AWS services to targets such as AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other accounts.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales

automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- [Amazon SNS](https://docs.aws.amazon.com/sns/latest/dg/welcome.html) – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

The code for this pattern is available in the `aws-config-cloudformation.template` file (attached).

Epics

Create the AWS CloudFormation stack

Task	Description	Skills required
Create the AWS CloudFormation stack.	Create an AWS CloudFormation stack by running the following command in AWS CLI: `$ aws cloudformation create-stack --stack-name=AWSConfigECR \ --template-body file://aws-config-cloudformation.template \ --parameters ParameterKey=<email>,ParameterValue=<myemail@example.com> \ --capabilities CAPABILITY_NAMED_IAM`	AWS DevOps

Test the AWS Config custom rule

Task	Description	Skills required
Test the AWS Config custom rule.	1. Sign in to the AWS Management Console, open the AWS Config console, and then choose **Resources**. 2. On the **Resource inventory** page, you can filter by resource category, resource type, and compliance status. 3. An Amazon ECR repository that contains `ecr:*` is `NON-COMPLIANT?` and an Amazon ECR repository that doesn't contain `ecr:*` is `COMPLIANT` . 4. The email address subscribed to the SNS topic receives notifications if an Amazon ECR repository contains non-compliant policy statements.	AWS DevOps

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Optimize multi-account serverless deployments by using the AWS CDK and GitHub Actions workflows

Sarat Chandra Pothula and VAMSI KRISHNA SUNKAVALLI, Amazon Web Services

Summary

Organizations deploying serverless infrastructure across multiple AWS accounts and environments often encounter challenges like code duplication, manual processes, and inconsistent practices. This pattern's solution shows how to use the AWS Cloud Development Kit (AWS CDK) in Go and GitHub Actions reusable workflows to streamline multi-account serverless infrastructure management. This solution demonstrates how you can define cloud resources as code, implement standardized continuous integration/continuous deployment (CI/CD) processes, and create modular, reusable components.

By using these tools, organizations can efficiently manage cross-account resources, implement consistent deployment pipelines, and simplify complex serverless architectures. The approach also enhances security and compliance by enforcing standardized practices for use with AWS accounts, ultimately improving productivity and reducing errors in serverless application development and deployment.

Prerequisites and limitations

Prerequisites

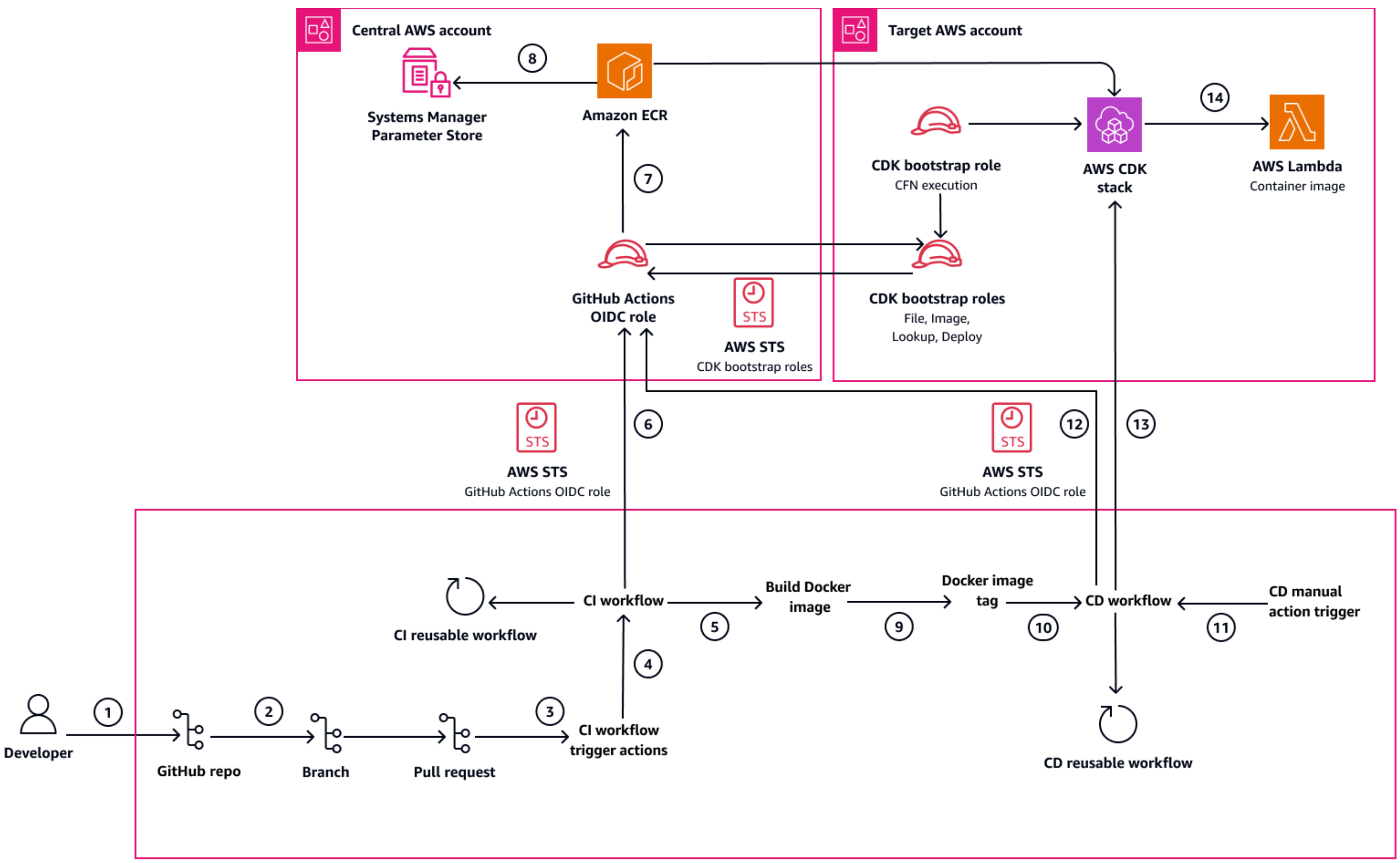
- An active AWS account.
- AWS Identity and Access Management (IAM) roles and permissions are in place for the deployment process. This includes permissions to access Amazon Elastic Container Registry (Amazon ECR) repositories, create AWS Lambda functions, and any other required resources across the target AWS accounts.
- AWS Command Line Interface (AWS CLI) version 2.9.11 or later, installed and configured.
- AWS Cloud Development Kit (AWS CDK) version 2.114.1 or later, installed and bootstrapped.
- Go 1.22 or later, installed.
- Docker 24.0.6 or later, installed.

Limitations

- **Language compatibility –** Go is a popular language for serverless applications. However, in addition to Go, the AWS CDK supports other programming languages, including C#, Java, Python, and TypeScript. If your organization has existing code bases or expertise in other languages, you might need to adapt or learn Go to fully use the solution described in the pattern.
- **Learning curve –** Adopting the AWS CDK, Go (if it's new to the organization), and GitHub reusable workflows might involve a learning curve for developers and DevOps teams. Training and documentation might be required to ensure smooth adoption and effective use of these technologies.

Architecture

The following diagram shows the workflow and architecture components for this pattern.



This solution performs the following steps:

1. The developer clones the repository, creates a new branch, and makes changes to the application code in their local environment.

2. The developer commits these changes and pushes the new branch to the GitHub repository.
3. The developer creates a pull request in the GitHub repository, proposing to merge their feature or new feature branch into the main branch.
4. This pull request triggers the continuous integration (CI) GitHub Actions workflow. The CI and the continuous deployment (CD) workflows in this pattern use reusable workflows, which are predefined, modular templates that can be shared and executed across different projects or repositories. Reusable workflows promote standardization and efficiency in the CI/CD processes.
5. The CI workflow sets up the necessary environment, generates a Docker tag for the image, and builds the Docker image using the application code.
6. The CI workflow authenticates with AWS by using the central AWS account GitHub OIDC role. For CI workflows, the central AWS account GitHub OIDC role uses AWS Security Token Service (AWS STS) to obtain temporary credentials. These credentials allow the role to build and push Docker images to the Amazon ECR repository of the central AWS account.
7. The CI workflow pushes the built Docker image to Amazon ECR.
8. The CI workflow stores the image tag to the Systems Manager Parameter Store.
9. After the CI workflow completes successfully, the Docker image tag is output.
10. When triggering the CD workflow, the developer manually inputs the image tag of the Docker image that they want to deploy. This image tag corresponds to the tag that was generated and pushed to Amazon ECR during the CI workflow.
11. The developer manually triggers the CD workflow, which uses the CD reusable workflow.
12. The CD workflow authenticates with AWS using the central AWS account GitHub OIDC role. For the CD workflow, AWS STS is first used to assume the central AWS account GitHub OIDC role. Then, this role assumes the CDK bootstrap roles for target account deployments.
13. The CD workflow uses the AWS CDK to synthesize AWS CloudFormation templates.
14. The CD workflow deploys the application to the target AWS account by using CDK deploy, using the manually specified image tag for the Lambda function.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions. CloudFormation is an integral part of the AWS CDK deployment process. The CDK synthesizes CloudFormation templates and then uses CloudFormation to create or update the resources in the AWS environment.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that's tightly integrated with GitHub repositories. You can use GitHub Actions to automate your build, test, and deployment pipeline.
- Go is an open source programming language that Google supports.

Code repository

The code for this pattern is available in the GitHub aws-cdk-golang-serverless-cicd-github-actions repository.

Best practices

- **Modular design** – Organize your AWS CDK code into modular and reusable constructs or stacks, promoting code reuse and maintainability across multiple accounts and projects.
- **Separation of concerns** – Separate the infrastructure code from the application code, allowing for independent deployment and management of each component.

- **Versioning and immutability** – Treat your infrastructure as code (IaC), and use Git for version control. Embrace immutable infrastructure principles by creating new resources instead of modifying existing ones.
- **Testing and validation** – Implement comprehensive testing strategies, including unit tests, integration tests, and end-to-end tests, to help support the correctness and reliability of your AWS CDK code and deployments.
- **Security and compliance** – Follow AWS security best practices, such as least-privilege access, secure communication, and data encryption. Implement compliance checks and auditing mechanisms to ensure adherence to organizational policies and regulatory requirements. Implement security best practices for container images, such as scanning for vulnerabilities, enforcing image signing, and adhering to compliance requirements for your organization.
- **Monitoring and logging** – Set up monitoring and logging mechanisms to track the health and performance of your serverless applications and infrastructure. Use AWS services like Amazon CloudWatch, AWS CloudTrail, and AWS X-Ray for monitoring and auditing purposes.
- **Automation and CI/CD** – Use GitHub reusable workflows and other CI/CD tools to automate the build, testing, and deployment processes, which can help support consistent and repeatable deployments across multiple accounts.
- **Environment management** – Maintain separate environments (for example, development, staging, and production). Implement strategies for promoting changes between environments, ensuring proper testing and validation before production deployments.
- **Documentation and collaboration** – Document your infrastructure code, deployment processes, and best practices to facilitate knowledge sharing and collaboration within your team.
- **Cost optimization** – Implement cost monitoring and optimization strategies, such as rightsizing resources, making use of auto-scaling, and taking advantage of AWS cost optimization services such as AWS Budgets and AWS Cost Explorer.
- **Disaster recovery and backup** – Plan for disaster recovery scenarios by implementing backup and restore mechanisms for your serverless applications and infrastructure resources.
- **Continuous improvement** – Review regularly and update your practices, tools, and processes to align with the latest best practices, security recommendations, and technological advancements in the serverless ecosystem.
- **Improve security posture** – Use AWS PrivateLink to improve the security posture of your virtual private cloud (VPC) by configuring interface VPC endpoints for Amazon ECR, AWS Lambda, and AWS Systems Manager Parameter Store.

Epics

Set up the environments

Task	Description	Skills required
Create an Amazon ECR repository in the central AWS account.	To share container images across multiple AWS accounts, you must configure cross-account access for Amazon ECR. First, create an Amazon ECR repository in the central AWS account. To create an Amazon ECR repository, run the following command: `aws ecr create-repository --repository-name sample-repo` In a later task, grant pull access to the other AWS accounts that need to use the container image.	AWS DevOps

Add cross-account permissions to the Amazon ECR repository.	To add cross-account permissions to the Amazon ECR repository in the central AWS account, run the following code:
	
	`{`
	`  "Version": "2008-10-17",`
	`  "Statement": [`
	`    {`
		AWS DevOps

Task	Description	Skills required

	``
	      "Sid": "LambdaEC
	RImageRetrievalPol
	icy",
	      "Effect": "Allow",
	      "Principal": {
	        "Service":
	 "lambda.amazonaws.
	com"
	      },
	      "Action": [
	        "ecr:Batc
	hGetImage",
	        "ecr:GetD
	ownloadUrlForLayer",
	      ],
	      "Condition": {
	        "StringLike": {
	          "aws:sour
	ceArn": "arn:aws:
	lambda:<Target_Reg
	ion>:<Target_Accou
	nt_ID>:function:*"
	        }
	      }
	    },
	    {
	      "Sid": "new
	 statement",
	      "Effect": "Allow",
	      "Principal": {
	        "AWS": "arn:aws:
	iam::<Target_Accou
	nt_ID>:root"
	        },
	      "Action": [
	        "ecr:Batc
	hGetImage",
	        "ecr:GetD
	ownloadUrlForLayer",
	      ],
	    }
	  ]
	``	

Task	Description	Skills required
	`}`	
Configure a role for GitHub OIDC role in the central AWS account.	1. Configure AWS to trust GitHub's OIDC as a federated identity, which includes adding the GitHub OIDC provider to AWS and configuring the role and trust policy in IAM. To do so, follow the instructions in [Configuring OpenID Connect in Amazon Web Services](#) in the GitHub documentation. 2. After the role is created, add necessary permissio ns to the role. For example, add permissio ns for Amazon ECR and AWS Systems Manager Parameter Store. For more information, see [Configuri ng a role for GitHub OIDC identity provider](#) in the IAM documentation.	AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Bootstrap the AWS environment in the target AWS accounts.</td><td>Set up a CDK environment in a specific AWS account and AWS Region that enables cross-account deployments from a central account and applies least-privilege principles to the CloudFormation execution role.

To bootstrap an AWS environment, run the following command:

<pre><code>cdk bootstrap aws://<Target_Account_ID>/<Target_Region> --trust <Central_Account_ID> --cloudformation-execution-policies arn:aws:iam::aws:policy/<Least_Privilege_Policy></code></pre></td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
Grant central AWS account OIDC role access to the target AWS account bootstrap roles.	The CDK bootstrap creates the following IAM roles that are designed to be assumed by the central AWS account during various stages of the CDK deployment process: • File publishing role • Image publishing role • Lookup role • Deployment role Each role has specific permissions tailored to its purpose, following the least-privilege principle. The `Target_Account_ID` and `Target_Region` in each role name help to indicate that these roles are unique across different AWS accounts and Regions. This approach supports clear identification and management in multi-account, multi-Region setups. `Target Account CDK Bootstrap Roles arn:aws:iam::<Target_Account_ID>:role/cdk-deploy-role-<Target_Account_ID>-<Target_Region> arn:aws:iam::<Target_Account_ID>:role/cdk-`	AWS DevOps

Task	Description	Skills required
	``` file-publishing-role- <Target_Account_ID>-< Target_Region> arn:aws:iam::<Target_ Account_ID>:role/cdk- image-publishing-role- <Target_Account_ID>- <Target_Region> arn:aws:iam::<Target _Account_ID>:role/ cdk-lookup-role-<T arget_Account_ID>- <Target_Region> ``` • Update the permissions policy for the OIDC role in the central account to grant it the ability to assume roles in the target account. This configuration enables the deployment of CDK stacks across different AWS accounts. By allowing the central account's OIDC role to adopt the necessary permissions from the target account, you create a secure bridge for cross-account CDK deployments. This approach maintains proper access control while facilitating seamless multi-account infrastructure management.	

Task	Description	Skills required

	To update the permissions policy for the OIDC role in the central AWS account, use the following code:
	``
	{
	    "Version": "2012-10-17",
	    "Statement": [
	        {
	            "Effect": "Allow",
	            "Action": "sts:AssumeRole",
	            "Resource": [
	                "arn:aws:iam::<Target_Account_ID>:role/cdk-deploy-role-<Target_Account_ID>-<Target_Region>",
	                "arn:aws:iam::<Target_Account_ID>:role/cdk-file-publishing-role-<Target_Account_ID>-<Target_Region>",
	                "arn:aws:iam::<Target_Account_ID>:role/cdk-image-publishing-role-<Target_Account_ID>-<Target_Region>",
	                "arn:aws:iam::<Target_Account_ID>:role/cdk-lookup-role-<Target_Account_ID>-<Target_Region>"
	            ]
	``	

Task	Description	Skills required

	``
	        }
	    ]
	}
	``	

Build the Docker image

Task	Description	Skills required
Clone the project repository.	To clone this pattern's [GitHub repository](https://github.com/aws-samples/aws-cdk-golang-serverless-cicd-github-actions), run the following command: `git clone https://github.com/aws-samples/aws-cdk-golang-serverless-cicd-github-actions.git`	AWS DevOps
Go to the Dockerfile path.	To navigate to the Dockerfile path, run the following command: `cd lambda`	AWS DevOps
Authenticate Docker with Amazon ECR.	Amazon ECR requires secure access to your private container repositories. By signing in this way, you're allowing Docker on your local machine or CI/CD environment to interact with Amazon ECR securely.	AWS DevOps

Task	Description	Skills required
	To authenticate Docker with Amazon ECR, run the following command: `aws ecr get-login -password --region <AWS_REGION> | docker login --username AWS --password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com` Revise the placeholders `AWS_REGION` and `AWS_Account_ID` with your information.	
Build the Docker image.	To build the Docker image, run the following command: `docker build --platform linux/arm64 -t sample-app .`	AWS DevOps

Task	Description	Skills required

Tag and push the Docker Image.	To tag and push the Docker image to the Amazon ECR repository, run the following commands:
	`docker tag sample-app:latest <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/<ECR_REPOSITORY>:<DOCKER_TAG>`
	`docker push <AWS_ACCOUNT_ID>.dkr.ecr.<AWS_REGION>.amazonaws.com/<ECR_REPOSITORY>:<DOCKER_TAG>`
	Revise the placeholders `AWS_Account_ID` , `AWS_REGION` , `ECR_REPOSITORY` , and `DOCKER_TAG` with your information.
		AWS DevOps

Deploy the AWS CDK app

Task	Description	Skills required
Synthesize the CDK stack with environment-specific variables.	To generate the CloudFormation template for your infrastructure as defined in your CDK code, run the following command:	AWS DevOps

Task	Description	Skills required
	```ENV=<environment> IMAGETAG=<image_tag> ECR_ARN=<ecr_repo_arn> cdk synth``` Revise the following placeholders with your information: • `environment` – Replaced with a specific environment name such as `dev`, `staging`, or `prod`. • `image_tag` – Replace with a specific tag for a Docker image such as `v1.0.0` or `latest`. • `ecr_repo_arn` – Replace with the Amazon Resource Name (ARN) of an Amazon ECR repository.	

Task	Description	Skills required
Deploy the CDK stack.	To deploy the CDK stack to your AWS account, run the following command. The `--require-approval never` flag means that the CDK will automatically approve and execute *all* changes. This includes changes that the CDK would normally flag as needing manual review (such as IAM policy changes or removal of resources). Make sure that your CDK code and CI/CD pipeline are well-tested and secure before using the `--require-approval never` flag in production environments. `ENV=<environment> IMAGETAG=<image_tag> ECR_ARN=<ecr_repo_arn> cdk deploy --require-approval never`	AWS DevOps

Automate CI/CD using GitHub Actions workflows

Task	Description	Skills required
Create a feature branch, and add your changes.	Use the cloned repository that you created earlier, create a feature branch, and then add your changes to	AWS DevOps

Task	Description	Skills required

	the application code. Use the following commands:
	`git checkout -b <feature_branch>`
	`git add .`
	`git commit -m "add your changes"`
	`git push origin <feature_branch>`
	
	Following are examples of changes:
	• Changes to the Lambda function logic
	• Adding new features or functionality to Lambda code
	• Fixing bugs or optimizing the existing code within the Lambda function
	
	GitHub Actions will use the reusable workflows and trigger the CI/CD pipelines.	

Merge your changes.	Create a pull request, and merge your changes to main.	AWS DevOps

Troubleshooting

Issue	Solution
`AccessDenied` errors when deploying resources across AWS accounts, for example, `AccessDenied: User not authorized to perform: "sts:AssumeRole"`.	To help resolve this issue, do the following to verify cross-account permissions: • Make sure that the necessary IAM roles and policies are in place for cross-account deployments. • Check that the `assume` role permissions are correctly configured.
Compatibility issues because of version mismatches, for example, `undefined: awscdkStack` error with an outdated CDK version.	To help resolve this issue, do the following to verify that you're using the required versions of the AWS CDK and Go: • Make sure you're using compatible versions of the AWS CDK and Go. • Check for any known issues or breaking changes in recent versions.
CI/CD pipeline failures, for example, `Error: No such file or directory` because of incorrect YAML configuration or `Permission denied` for protected branches.	To help resolve issues with the GitHub Actions configuration, verify that the reusable workflows are properly referenced and configured.

Related resources

AWS resources

- AWS Best Practices for Security, Identity, & Compliance
- AWS CDK Workshop
- AWS Cloud Development Kit Library
- Create a Lambda function using a container image
- Identity and Access Management for Amazon Elastic Container Registry

- Working with the AWS CDK in Go

Other resources

- Configuring OpenID Connect in Amazon Web Services (GitHub documentation)
- Golang Documentation
- Quickstart for GitHub Actions (GitHub documentation)
- Reusing workflows (GitHub documentation)

Provision AWS Service Catalog products based on AWS CloudFormation templates by using GitHub Actions

Ashish Bhatt and Ruchika Modi, Amazon Web Services

Summary

This pattern provides organizations with a streamlined approach using AWS Service Catalog products and portfolios to provision standardized and compliant AWS services across teams. AWS CloudFormation helps to combine essential components in Service Catalog products and portfolios for provisioning base network infrastructure on AWS Cloud. This pattern also promotes DevOps practices by integrating infrastructure as code (IaC) into automated development workflows by using GitHub Actions.

AWS Service Catalog enables organizations to create and manage approved IT services on AWS, offering benefits such as standardization, centralized control, self-service provisioning, and cost management. By automating the deployment of Service Catalog portfolios and products through GitHub Actions, companies can do the following:

- Achieve consistent and repeatable deployments.
- Use version control for IaC.
- Integrate cloud resource management with existing development workflows.

This combination streamlines cloud operations, enforces compliance, and accelerates the delivery of approved services while reducing manual errors and improving overall efficiency.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Access to GitHub repository
- Basic understanding of AWS CloudFormation and AWS Service Catalog
- An Amazon Simple Storage Service (Amazon S3) bucket to host CloudFormation templates
- An AWS Identity and Access Management (IAM) role named `github-actions` that is used for connectivity between GitHub and AWS

Limitations

- This pattern's reusable code has been tested only with GitHub Actions.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

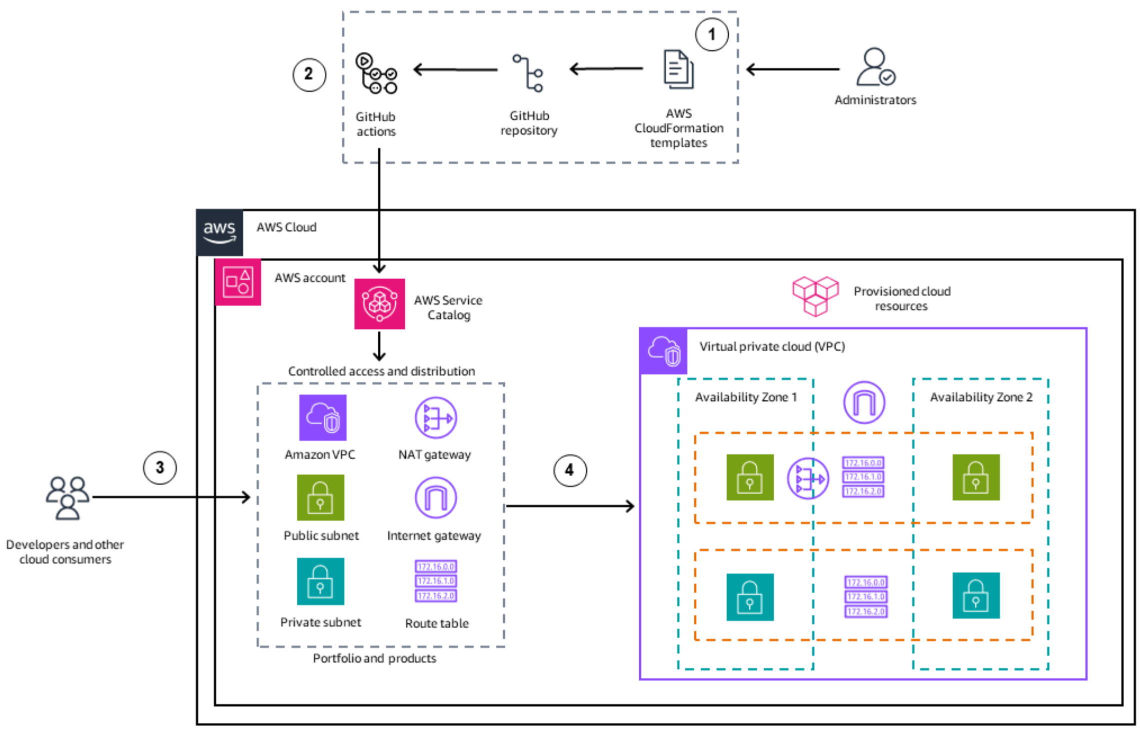
Product versions

This pattern's solution was created by using the following GitHub Marketplace actions and their respective versions:

- `actions/checkout@v4`
- `aws-actions/configure-aws-credentials@v2`
- `aws-actions/aws-cloudformation-github-deploy@v1.2.0`

Architecture

The following diagram shows the architecture for this solution.



1. Administrators or platform engineers push standardized CloudFormation templates to a GitHub repository, where the templates are maintained. The GitHub repo also contains workflows that automate the provisioning of AWS Service Catalog using GitHub Actions.
2. GitHub Actions triggers a workflow that connects to the AWS Cloud using an OpenID Connect (OIDC) provider to provision Service Catalog.
3. Service Catalog contains the portfolio and products that developers can directly use to provision standardized AWS resources. This pattern bundles AWS resources such as virtual private clouds (VPCs), subnets, NAT and internet gateways, and route tables.
4. After the developer creates a Service Catalog product, Service Catalog converts it into pre-configured and standardized AWS resources. As a result, developers save time because they don't need to provision individual resources and configure them manually.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions. It's an infrastructure as code (IaC) service that can be easily used as one of the product types with AWS Service Catalog.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Service Catalog helps you centrally manages catalog of IT services that are approved for AWS. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Others

- GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that's tightly integrated with GitHub repositories. You can use GitHub Actions to automate your build, test, and deployment pipeline.

Code repository

The code for this pattern is available in the GitHub service-catalog-with-github-actions repository. The repo contains the following files of interest:

- `github/workflows`:
  - `e2e-test.yaml` – This file calls `workflow.yaml`, which is the reusable workflow. This workflow is triggered as soon as there is a commit and push on a branch.
  - `workflow.yaml` – This file contains the reusable workflow for this solution and is configured with `workflow_call` as its trigger. As a reusable workflow, `workflow.yaml` can be called from any other workflow.
- `templates`:
  - `servicecatalog-portfolio.yaml` – This CloudFormation template includes resources that provision the Service Catalog portfolio and Service Catalog product. The template contains a set of parameters that are used while provisioning the Service Catalog portfolio and products. One parameter accepts an Amazon S3 file URL where the template `vpc.yaml` is uploaded. Although this pattern includes the `vpc.yaml` file to provision AWS resources, you can also use the parameter S3 file URL for configuration.
  - `vpc.yaml` – This CloudFormation template contains AWS resources to be added in the Service Catalog product. AWS resources include VPCs, subnets, internet gateways, NAT gateways, and route tables. The `vpc.yaml` template is an example of how you can use any CloudFormation template with a Service Catalog product and portfolio template.

Best practices

- See Security Best Practices for AWS Service Catalog in the AWS Service Catalog documentation.
- See Security hardening for GitHub Actions in the GitHub documentation.

Epics

Set up local workstation

Task	Description	Skills required
Set up Git on your local workstation.	To install and configure Git on your local workstation, use the Getting Started – Installin	App developer

Task	Description	Skills required
	g Git instructions in the Git documentation.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Clone the GitHub project repo.</td><td>To clone the GitHub project repo, do the following:

1. Open the GitHub repo for this pattern
2. Choose Code to see cloning options, and copy the URL provided in the HTTPS tab.
3. Create a folder for your project on your workstati on.
4. Open a terminal, and navigate to this folder
5. To clone the GitHub repo, run the following command using the URL that you copied in step 2:
<pre><code>git clone https://github.com/aws-samples/service-catalog-with-github-actions.git</code></pre>6. When cloning is complete, to change to the cloned repo in your project folder, run the following command:
<pre><code>cd <folder-name>/service-catalog-with-github-actions</code></pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	7. Open the project in an integrated development environment (IDE) of your choice.	

Set up the OIDC provider

Task	Description	Skills required
Configure an OIDC provider.	Create an OpenID Connect (OIDC) provider that allows the GitHub Actions workflows to access resources in AWS, without needing to store the AWS credentials as long-lived GitHub secrets. For instructions, see [Configuring OpenID Connect in Amazon Web Services](#) in the GitHub documentation. After an OIDC provider is configured, the trust policy of the IAM role `github-actions` , mentioned earlier in the [Prerequisites](#), will be updated.	AWS administrator, AWS DevOps, General AWS

Trigger GitHub Actions pipeline to deploy Service Catalog portfolio and products

Task	Description	Skills required
Update `e2e-test.yaml` .	The `e2e-test.yaml` file triggers the reusable	DevOps engineer

Task	Description	Skills required

	workflow at `workflow.yaml` . Update and validate the values for the following input parameters in `e2e-test.yaml` :
	
	• `aws_account_id` – Specify the correct AWS account.
	• `aws_region` – Specify the correct AWS Region.
	• `s3BucketName` – Specify the Amazon S3 bucket to hold the CloudFormation templates.
	• The workflow file requires two IAM roles as input:
	  • `LaunchConstraintRole` - The IAM role that AWS Service Catalog assumes when an end user launches, updates, or terminates a product.
	  • `PrincipalArn` - The Amazon Resource Name (ARN) of the principal (IAM user, role, or group) that will be associated to the Service Catalog Portfolio. If the `PrincipalType` is `IAM`, the supported value is a fully defined [IAM Amazon Resource Name](#)	

Task	Description	Skills required
	(ARN). If the `PrincipalType` is `IAM_PATTERN`, the supported value is an IAM ARN without an `AccountID` in the following format: `arn:partition:iam:::resource-type/resource-id`	

Validate deployment

Task	Description	Skills required
Validate the Service Catalog resources.	To validate the Service Catalog resources, do the following: 1. Sign in to the AWS Management Console of the AWS account, and verify that the AWS Region is correct. 2. Navigate to **AWS Service Catalog** and validate that a portfolio is present under **Administration**, **Portfolios**. 3. Choose the portfolio, and validate the information on the **Products**, **Constraints**, and **Access** tabs.	AWS DevOps

Clean up resources

Task	Description	Skills required
Delete the CloudFormation stack.	To delete the CloudFormation stack, do the following: 1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation. 2. On the navigation bar at the top of the screen, choose the AWS Region where the stack is located. 3. On the **Stacks** page, choose the stack that you want to delete. The stack must be currently running. 4. In the stack details pane, choose **Delete**. 5. Select **Delete stack** when prompted. For more information, see [Delete a stack from the CloudFormation console](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-delete-stack.html) in the CloudFormation documentation	DevOps engineer, AWS administrator

Troubleshooting

Issue	Solution

`e2e-test`
 	
`Can't find 'action.yml', 'action.yaml' or 'Dockerfile' under '*/home/runner/work/service-catalog-with-github-actions/service-catalog-with-github-actions`
 	
`Did you forget to run actions/checkout before running your local action?`
	To make sure that you have the correct repository settings enabled, do the following:
	
	1. Navigate to the Github repository, **Settings** tab.
	2. Choose **Actions**, **General** from the menu on the left side
	3. Go to the **Access** section, and select the option **Accessible from repositories in the 'XXX' organization**.

Related resources

AWS documentation

- Overview of Service Catalog

Other resources

- About events that trigger workflows (GitHub documentation)
- Reusing workflows (GitHub documentation)

Additional information

To see screenshots related to the Epics, go to the **Images** folder in this pattern's GitHub repo. The following screenshots are available:

- AWS Service Catalog portfolio, Administration section
- AWS Service Catalog product, Administration section
- AWS Service Catalog product, User/Provisioning section

Provision least-privilege IAM roles by deploying a role vending machine solution

Benjamin Morris, Nima Fotouhi, Aman Kaur Gandhi, and Chad Moon, Amazon Web Services

Summary

Over-scoped AWS Identity and Access Management (IAM) role permissions for pipelines can introduce unnecessary risk to an organization. Developers sometimes grant broad permissions during development but neglect to scope down permissions after troubleshooting their code. This causes a problem where powerful roles are present without a business need and may have never been reviewed by a security engineer.

This pattern offers a solution to this problem: the role vending machine (RVM). Using a secure and centralized deployment model, the RVM demonstrates how to provision least-privilege IAM roles for individual GitHub repositories' pipelines with minimal effort from developers. Because the RVM is a central solution, you can configure your security teams as required reviewers to approve changes. This approach allows security to reject over-permissioned pipeline role requests.

The RVM takes Terraform code as input and generates pipeline-ready IAM roles as output. The required inputs are the AWS account ID, GitHub repository name, and permissions policy. The RVM uses these inputs to create the role's trust policy and permissions policy. The resulting trust policy allows the specified GitHub repository to assume the role and use it for pipeline operations.

The RVM uses an IAM role (configured during bootstrap). This role has permissions to assume a role-provisioning-role in each account in the organization. The role is configured through either AWS Control Tower Account Factory for Terraform (AFT) or AWS CloudFormation StackSets. The role-provisioning-roles are the roles that actually create the pipeline roles for developers.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A GitHub organization that is used to deploy infrastructure as code (IaC) through GitHub Actions. (GitHub Enterprise/Premium/Ultimate are *not* required.)
- A multi-account AWS environment. This environment does not need to be part of AWS Organizations.

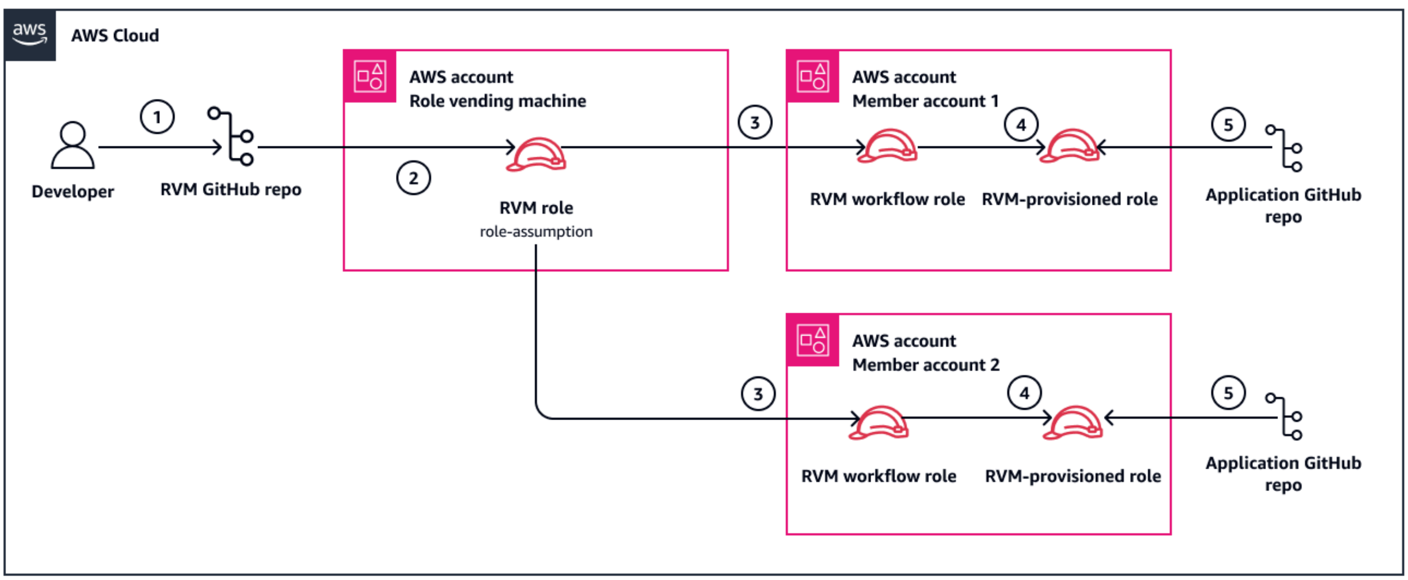
- A mechanism for deploying an IAM role in all AWS accounts (for example, AFT or CloudFormation StackSets).
- Terraform version 1.3 or latter installed and configured.
- Terraform AWS Provider version 4 or later installed and configured.

Limitations

- This pattern's code is specific to GitHub Actions and Terraform. However, the pattern's general concepts can be reused in other continuous integration and delivery (CI/CD) frameworks.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

The following diagram illustrates the workflow for this pattern.



The workflow for the typical usage of the role vending machine consists of the following steps:

1. A developer pushes code that contains Terraform code for a newly requested IAM role to the RVM GitHub repository. This action triggers the RVM GitHub Actions pipeline.
2. The pipeline uses an OpenID Connect (OIDC) trust policy to assume the RVM role-assumption role.

3. As the RVM pipeline runs, it assumes the RVM workflow role in the account in which it's provisioning the developer's new IAM role. (The RVM workflow role was provisioned by using AFT or CloudFormation StackSets.)
4. The RVM creates the developer's IAM role with appropriate permissions and trust, so that the role can be assumed by other application pipelines.
5. App developers can configure their app pipelines to assume this RVM-provisioned role.

The created role includes the permissions requested by the developer and a `ReadOnlyAccess` policy. The role is assumable only by pipelines that run against the `main` branch of the developer's specified repository. This approach helps to ensure that branch protection and reviews can be required to use the role.

Automation and scale

Least-privilege permissions require attention to detail for each role being provisioned. This model reduces the complexity required to create these roles, allowing developers to create the roles they need without much additional learning or effort.

Tools

AWS services

- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Organizations](#) is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.

Other tools

- [Git](#) is an open source, distributed version control system. It includes the ability to create an [organization account](#).
- [GitHub Actions](#) is a continuous integration and continuous delivery (CI/CD) platform that's tightly integrated with GitHub repositories. You can use GitHub Actions to automate your build, test, and deployment pipeline.
- [Terraform](#) is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub role-vending-machine repository.

Best practices

- **Make the right way easy and the wrong way hard –** Make it easy to do the right thing. If developers are struggling with the RVM provisioning process, they might attempt to create roles through other means, which undermines the central nature of RVM. Make sure that your security team provides clear guidance about how to use the RVM securely and effectively.

  You should also make it hard for developers to do the wrong thing. Use service control policies (SCPs) or permission boundaries to restrict what roles can create other roles. This approach can help limit role creation to just RVM and other trusted sources.
- **Provide good examples –** Inevitably, some developers will adapt existing roles in the RVM repository as informal templates for granting permissions to their new roles. If you have least-permissions examples that they can copy from, that can reduce the risk of developers requesting broad, wildcard-heavy permissions. If you start with highly permissioned roles with lots of wildcards, that problem can multiply as time goes on.
- **Use naming conventions and conditions –** Even if a developer doesn't know all the resource names that their application will create, they should still limit role permissions by using a naming convention. For example, if they're creating Amazon S3 buckets, their resource key's value might look like `arn:aws:s3:::myorg-myapp-dev-*` so that their role doesn't have permissions beyond buckets matching that name. Enforcing naming convention through an IAM policy has the additional benefit of improving compliance with the naming convention. This improvement occurs because non-matching resources will not be permitted to be created.
- **Require pull request (PR) reviews –** The value of the RVM solution is that it creates a central location where new pipeline roles can be reviewed. However, this design is only useful if there are guardrails that help ensure secure, high-quality code is committed to the RVM. Protect the branches that are used to deploy code (for example, `main`) from direct pushes and require approvals for any merge requests that target them.
- **Configure read-only roles –** By default, the RVM provisions a `readonly` version of each requested role. This role can be used in CI/CD pipelines that don't write data, such as a `terraform plan` pipeline workflow. This approach helps prevent unwanted changes if a read-only workflow misbehaves.

By default, the AWS managed `ReadOnlyAccess` policy is attached to both the read-only roles and read-write roles. This policy reduces the need for iteration when determining required permissions, but it might be overly permissive for some organizations. If you want, you can remove the policy from the Terraform code.

- **Grant minimum permissions** – Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Prepare environment

Task	Description	Skills required
Copy the sample repository to your GitHub organization.	Clone this pattern's repository or fork this repository to your GitHub organization so that you can adapt it for your needs. • If you choose to clone this repository, you can use the following command: `git clone https://github.com/aws-samples/role-vending-machine` • If you choose to fork the repository to your GitHub organization, you can use the following command: `gh repo fork aws-samples/role-vending-machine`	DevOps engineer

Task	Description	Skills required
	`--org YOUR_ORGA NIZATION_NAME`	
Determine the AWS account for the RVM.	Determine which infrastru cture deployment AWS account to use for the RVM. Don’t use the management or root account.	Cloud architect

Task	Description	Skills required
(Optional) Allow the organization's pipelines to create PRs.	**Note** This step is only necessary if you want to allow the `generate_providers_and_account_vars` workflow to create PRs. To allow your organization’s pipelines to create PRs, use the following steps: 1. Go to `https://github.com/organizations/YOUR_ORG/settings/actions` . 2. Select **Allow GitHub Actions to create and approve pull requests**. For more information, see [Managing GitHub Actions settings for a repository](#) in the GitHub documentation.	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Grant read-only permissions to the RVM account.</td><td>

Create a delegation policy in your management account that grants your RVM account read-only permissions. This allows your RVM GitHub workflows to dynamically pull a list of your AWS organization's accounts when the `generate_providers_and_account_vars.py` script runs.

Use the following code and replace `<YOUR RVM Account ID>` with the AWS account ID that you selected in Step 2:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<YOUR RVM Account ID>:root"
      },
      "Action": [
        "organizations:ListAccounts",
        "organizations:DescribeOrganization",
```

</td><td>Cloud administrator</td></tr>
</table>

Task	Description	Skills required
	`"organizations:DescribeOrganizationalUnit", "organizations:ListRoots", "organizations:ListAWSServiceAccessForOrganization", "organizations:ListDelegatedAdministrators" ], "Resource": "*" } ] }`	
Update default values from the sample repo.	To configure the RVM to operate in your specific environment and AWS Region, do the following: 1. Update `scripts/generate_providers_and_account_vars.py` and other files (such as the `bootstrap` folder) with the appropriate Region that you operate in. 2. Update the `.github/workflows/.env` file with the AWS account ID, Region, and any other variables that you want to specify.	DevOps engineer

Initialize infrastructure

Task	Description	Skills required
Bootstrap the RVM repo.	This step is necessary to create the OIDC trust and IAM roles used by the RVM pipeline itself, so that it can start operating and vending other roles. In the context of your RVM account, manually run a `terraform apply` command from the `scripts/bootstrap` directory. Provide any required values based on variable documentation.	DevOps engineer

Configure operations

Task	Description	Skills required
Deploy the `github-workflow-rvm` and `github-workflow-rvm-readonly` roles to all accounts.	Choose a deployment method that aligns with your organization's practices, such as AFT or StackSets. Use that method to deploy the two IAM roles in the `scripts/assumed_role/main.tf` file (default names `github-workflow-rvm` and `github-workflow-rvm-readonly`) to each account where you want the RVM to	AWS administrator

Task	Description	Skills required
	be able to create pipeline roles. These IAM roles have trust policies that allows the RVM account's role-assumption role (or its `readonly` equivalent) to assume it. The roles also have IAM permission policies that allow them to read and write (unless using the `readonly` role) roles that match `github-workflow-role-*` .	

Task	Description	Skills required
Run the `generate_providers_and_account_vars` workflow.	To configure your RVM so that it's ready to create pipeline roles, do the following: 1. Run the `generate_providers_and_account_vars` workflow by using [workflow_dispatch](https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows#workflow_dispatch). 2. Merge the PR that the workflow creates. After the workflow completes , the RVM is ready to: • Accept requests for new pipeline roles. • Create either read-only or read-write roles as requested. • Deploy roles to the specified AWS accounts.	DevOps engineer

Troubleshooting

Issue	Solution
I created a role by using the RVM, but GitHub isn't able to assume it.	Verify that the name of the GitHub repository matches the name provided to the `github_workflow_roles` module. Roles are scoped so that only one repository can assume them. Similarly, verify that the branch used in the GitHub pipeline matches the name of

Issue	Solution
	the branch provided to the `github_workflow_roles` module. Typically, RVM-created roles with write permissions can only be used by workflows scoped to the `main` branch (that is, deployments sourced from `main`).
My read-only role is failing to run its pipeline because it lacks permissions to read a specific resource.	Although the `ReadOnlyAccess` policy provides broad read-only permissions, the policy doesn't have some read actions (for example, certain AWS Security Hub actions). You can add specific action permissions by using the `inline_policy_readonly` parameter of the `github-workflow-roles` module.

Related resources

- Best practices for using AWS CloudFormation StackSets
- Organizing Your AWS Environment Using Multiple Accounts
- Overview of AWS Control Tower Account Factory for Terraform (AFT)
- Policy best practices

Additional information

Using GitHub environments

GitHub environments are an alternative approach to branch-based restrictions for role access. If you prefer to use a GitHub environment, following is an example of the syntax for an additional condition in the IAM trust policy. This syntax specifies that the role can be used only when the GitHub action is running in the `Production` environment.

```
"StringLike": {
```

```
    "token.actions.githubusercontent.com:sub": "repo:octo-org/octo-
repo:environment:Production"
}
```

The example syntax uses the following placeholder values:

- `octo-org` is the GitHub organization name.
- `octo-repo` is the repository name.
- `Production` is the specific GitHub environment name.

Publish Amazon CloudWatch metrics to a CSV file

Abdullahi Olaoye, Amazon Web Services

Summary

This pattern uses a Python script to retrieve Amazon CloudWatch metrics and to convert the metrics information into a comma-separated values (CSV) file for improved readability. The script takes the AWS service whose metrics should be retrieved as a required argument. You can specify the AWS Region and AWS credential profile as optional arguments. If you don't specify those arguments, the script uses the default Region and profile that are configured for the workstation where the script is run. After the script runs, it generates and stores a CSV file in the same directory.

See the *Attachments* section for the script and associated files provided with this pattern.

Prerequisites and limitations

Prerequisites

- Python 3.x
- AWS Command Line Interface (AWS CLI)

Limitations

The script currently supports the following AWS services:

- AWS Lambda
- Amazon Elastic Compute Cloud (Amazon EC2)
  - By default, the script doesn't collect Amazon Elastic Block Store (Amazon EBS) volume metrics. To collect Amazon EBS metrics, you must modify the attached `metrics.yaml` file.
- Amazon Relational Database Service (Amazon RDS)
  - However, the script doesn't support Amazon Aurora.
- Application Load Balancer
- Network Load Balancer
- Amazon API Gateway

Tools

- [Amazon CloudWatch](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html) is a monitoring service built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers. CloudWatch provides data and actionable insights to help you monitor your applications, respond to systemwide performance changes, optimize resource utilization, and get a unified view of operational health. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, and provides a unified view of AWS resources, applications, and services that run on AWS and on-premises servers.

Epics

Install and configure the prerequisites

Task	Description	Skills required
Install the prerequisites.	Run the following command: `$ pip3 install -r requirements.txt`	Developer
Configure the AWS CLI.	Run the following command: `$ aws configure`	Developer

Configure the Python script

Task	Description	Skills required
Open the script.	To change the default configuration of the script, open `metrics.yaml` .	Developer
Set the period for the script.	This is the time period to fetch. The default period is 5 minutes (300 seconds). You can change the time	Developer

Task	Description	Skills required
	period, but note the following limitations: • If the hours value that you specify is between 3 hours and 15 days ago, use a multiple of 60 seconds (1 minute) for the period. • If the hours value that you specify is between 15 hours and 63 days ago, use a multiple of 300 seconds (5 minutes) for the period. • If the hours value that you specify is greater than 63 days ago, use a multiple of 3,600 seconds (1 hour) for the period. Otherwise, the API operation won't return any data points.	
Set the hours for the script.	This value specifies how many hours of metrics you want to fetch. The default is 1 hour. To retrieve multiple days of metrics, provide the value in hours. For example, for 2 days, specify 48.	Developer

Task	Description	Skills required
Change statistics values for the script.	(Optional) The global statistics value is `Average`, which is used when fetching metrics that do not have a specific statistics value assigned. The script supports the statistics values `Maximum`, `SampleCount`, and `Sum`.	Developer

Run the Python script

Task	Description	Skills required
Run the script.	Use the following command: `$ python3 cwreport.py <service>` To see a list of service values and the optional `region` and `profile` parameters, run the following command: `$ python3 cwreport.py -h` For more information about the optional parameters, see the *Additional information* section.	Developer

Related resources

- Configuring the AWS CLI

- Using Amazon CloudWatch metrics
- Amazon CloudWatch documentation
- EC2 CloudWatch Metrics
- AWS Lambda Metrics
- Amazon RDS Metrics
- Application Load Balancer Metrics
- Network Load Balancer Metrics
- Amazon API Gateway Metrics

Additional information

Script usage

```
$ python3 cwreport.py -h
```

Example syntax

```
python3 cwreport.py <service> <--region=Optional Region> <--profile=Optional credential
 profile>
```

Parameters

- **service (required)** – The service you want to run the script against. The script currently supports these services: AWS Lambda, Amazon EC2, Amazon RDS, Application Load Balancer, Network Load Balancer, and API Gateway.
- **region (optional)** – The AWS Region to fetch metrics from. The default Region is `ap-southeast-1`.
- **profile (optional)** – The AWS CLI named profile to use. If this parameter isn't specified, the default configured credential profile is used.

Examples

- To use the default Region `ap-southeast-1` and default configured credentials to fetch Amazon EC2 metrics: `$ python3 cwreport.py ec2`

- To specify a Region and fetch API Gateway metrics: `$ python3 cwreport.py apigateway --region us-east-1`
- To specify an AWS profile and fetch Amazon EC2 metrics: `$ python3 cwreport.py ec2 --profile testprofile`
- To specify both Region and profile to fetch Amazon EC2 metrics: `$ python3 cwreport.py ec2 --region us-east-1 --profile testprofile`

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Remove Amazon EC2 entries across AWS accounts from AWS Managed Microsoft AD by using AWS Lambda automation

Dr. Rahul Sharad Gaikwad and Tamilselvan P, Amazon Web Services

Summary

Active Directory (AD) is a Microsoft scripting tool that manages domain information and user interactions with network services. It's widely used among managed services providers (MSPs) to manage employee credentials and access permissions. Because AD attackers can use inactive accounts to try and hack into an organization, it's important to find inactive accounts and disable them on a routine maintenance schedule. With AWS Directory Service for Microsoft Active Directory, you can run Microsoft Active Directory as a managed service. This pattern can help you to configure AWS Lambda automation to quickly find and remove inactive accounts.

If the following scenarios apply to your organization, this pattern can assist you:

- **Centralized AD management –** If your organization has multiple AWS accounts, each with its own AD deployment, it can be challenging to manage user accounts and access permissions consistently across all accounts. With an across-accounts AD cleanup solution, you can disable or remove inactive accounts from all AD instances in a centralized manner.
- **AD restructuring or migration –** If your organization plans to restructure or migrate its AD deployment, an across-accounts AD cleanup solution can help you prepare the environment. The solution can help you remove unnecessary or inactive accounts, simplify the migration process, and reduce potential conflicts or issues.

When you use this pattern, you can get the following benefits:

- Improve database and server performance, and fix vulnerabilities in your security from inactive accounts.
- If your AD server is hosted in the cloud, removing inactive accounts can also reduce storage costs while improving performance. Your monthly bills might decrease because bandwidth charges and compute resources can both drop.
- Keep potential attackers at bay with a clean Active Directory.

Prerequisites and limitations

Prerequisites

- An active parent AWS account and one or multiple child accounts. In this pattern, a *parent account* is where Active Directory is created. *Child accounts* host Windows servers and are joined through the parent account Active Directory.
- Git installed and configured on a local workstation.
- Terraform installed and configured on a local workstation.
- AWS Managed Microsoft AD directory configured in the parent account and shared to all child accounts. For more details, see Tutorial: Sharing your AWS Managed Microsoft AD directory for seamless EC2 domain-join in the *AWS Directory Service Administration Guide*.
- A virtual private cloud (VPC) peering connection or AWS Transit Gateway connection available between the VPC of AWS Directory Service (parent account) and the VPC of the Amazon Elastic Compute Cloud (Amazon EC2) instances (child accounts). For more details, see Configure a VPC peering connection between the directory owner and the directory consumer account in the *AWS Directory Service Administration Guide*.
- A Windows machine configured with the `EC2WindowsUserdata` script on all the parent and child accounts. The script file is available in the root of this pattern's code repository.
- A cross-account AWS Identity and Access Management (IAM) role available on each child account that's configured with a trust policy to allow the use of an AWS Lambda function from the parent account. For more information, see Sending and receiving events between AWS accounts in Amazon EventBridge in the *Amazon EventBridge User Guide*.
- The following secrets values available in AWS Systems Manager Parameter Store of the parent account:
  - `domainJoinUser` – Username of the directory service
  - `domainJoinPassword` – Password of the directory service

  For more information about secrets, see Create an AWS Secrets Manager secret in the *AWS Secrets Manager User Guide*.

Limitations

- Creating a resource in a child account isn't automated with Terraform. You must create the following resources manually by using the AWS Management Console:

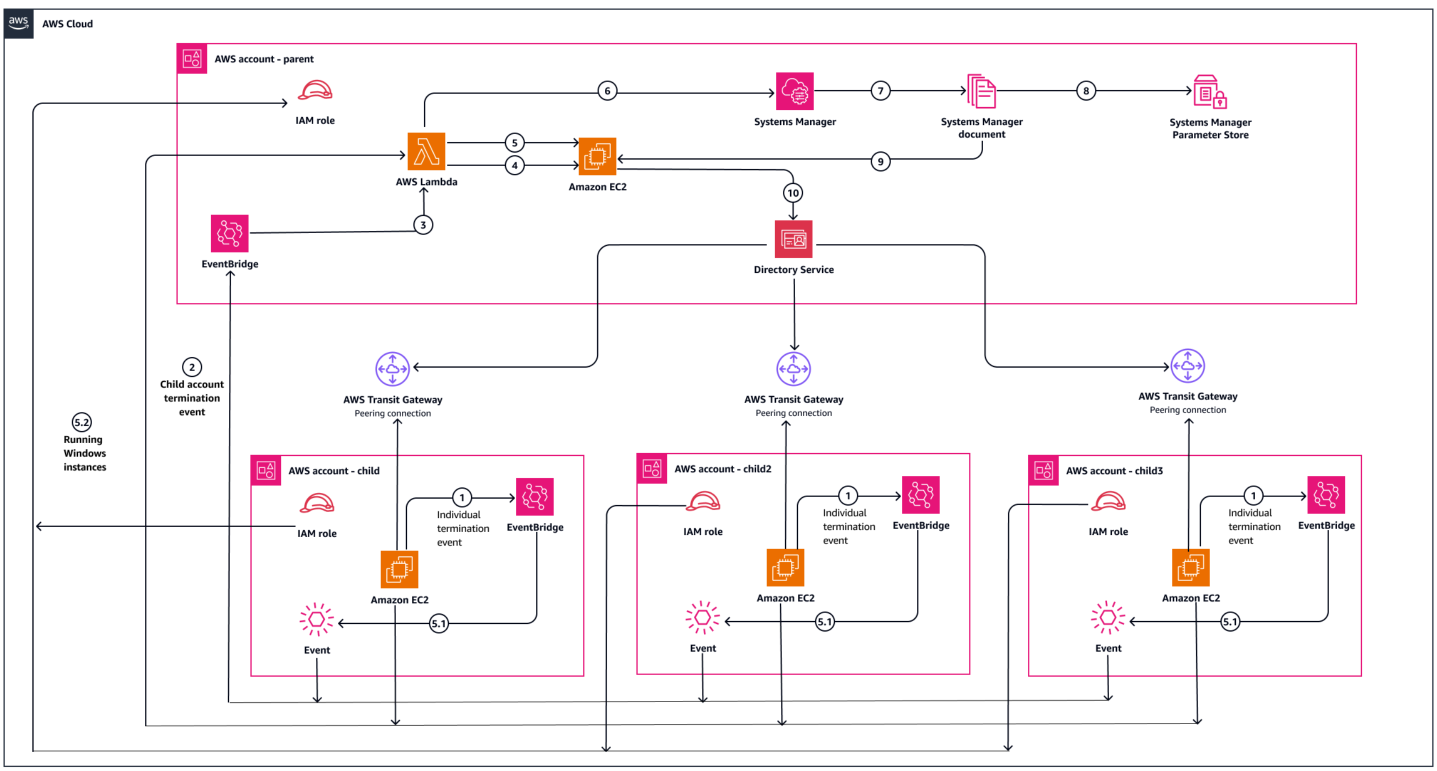
  - Amazon EventBridge rule to send the Amazon EC2 termination events to the parent account
  - Amazon EC2 cross-account role creation in the child account with trust policy
  - VPC peering or Transit Gateway connection
- Some AWS services aren't available in all AWS Regions. For Region availability, see [AWS services by Region](). For specific endpoints, see [Service endpoints and quotas](), and choose the link for the service.

Product versions

- [Terraform version 1.1.9 or later]()
- [Terraform AWS Provider version 3.0 or higher]()

Architecture

The following diagram displays the high-level architecture of the solution.



The architecture diagram illustrates the following process:

1. In child accounts, the EventBridge rule collects all the Amazon EC2 termination events. The rule sends those events to EventBridge which is present in the parent account.

2. From the parent account, EventBridge collects all the events and contains the rule to trigger the Lambda function `ADcleanup-Lambda`.
3. The parent account receives any termination events from the parent or child account and triggers the Lambda function.
4. The Lambda function makes the call to Amazon EC2 Auto Scaling groups using the Python boto module and gets the random instance ID. The instance ID is used to execute Systems Manager commands.
5. The Lambda function makes another call to Amazon EC2 using the boto module. The Lambda function gets the private IP addresses of the running Windows servers and stores the addresses in a temporary variable. In steps 5.1 and 5.2, the running Windows EC2 instances are collected from child accounts.
6. The Lambda function makes another call to Systems Manager to get the computer information that is connected to AWS Directory Service.
7. An AWS Systems Manager document helps to execute the PowerShell command on Amazon EC2 Windows servers to get the private IP addresses of the computers which are connected to AD. (The Systems Manager document uses the instance ID that was obtained in step 4.)
8. The AD domain username and passwords are stored in the AWS Systems Manager Parameter Store. AWS Lambda and Systems Manager make a call to Parameter Store and get the username and password values to use to connect to AD.
9. Using the Systems Manager document, the PowerShell script is executed on the Amazon EC2 Windows server using the instance id obtained earlier in step 4.
10. Amazon EC2 connects to AWS Directory Service by using PowerShell commands and remove the computers that are not in use or inactive.

Tools

AWS services

- AWS Directory Service provides multiple ways to use Microsoft Active Directory (AD) with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Relational Database Service (Amazon RDS) for SQL Server, and Amazon FSx for Windows File Server.
- AWS Directory Service for Microsoft Active Directory enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.

- Amazon EventBridge is a serverless event bus service that that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them. With IAM, you can specify who or what can access services and resources in AWS, centrally manage fine-grained permissions, and analyze access to refine permissions across AWS.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.
- AWS Systems Manager documents define the actions that Systems Manager performs on your managed instances. Systems Manager includes more than 100 pre-configured documents that you can use by specifying parameters at runtime.
- AWS Systems Manager Parameter Store is a capability of AWS Systems Manager and provides secure, hierarchical storage for configuration data management and secrets management.

Other tools

- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources.
- PowerShell is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub aws-lambda-ad-cleanup-terraform-samples repository.

Best practices

- **Automatically join domains.** When you launch a Windows instance that's to be part of an AWS Directory Service domain, join the domain during the instance creation process instead of manually adding the instance later. To automatically join a domain, select the correct directory from the **Domain join directory** dropdown list when launching a new instance. For more details, see [Seamlessly join an Amazon EC2 Windows instance to your AWS Managed Microsoft AD Active Directory]() in the *AWS Directory Service Administration Guide*.

- **Delete unused accounts.** It's common to find accounts in AD that have never been used. Like disabled or inactive accounts that remain in the system, neglected unused accounts can slow down your AD system or make your organization vulnerable to data breaches.
- **Automate Active Directory cleanups.** To help mitigate security risks and prevent obsolete accounts from impacting AD performance, conduct AD cleanups should at regular intervals. You can accomplish most AD management and cleanup tasks by writing scripts. Example tasks include removing disabled and inactive accounts, deleting empty and inactive groups, and locating expired user accounts and passwords.

Epics

Set up child accounts

Task	Description	Skills required
Create a cross-account role in the child account.	To create a cross-account role in a child account, do the following: 1. For each child account, create a role named `ec2crossaccountrole` by using the managed policy named `AmazonEC2ReadOnlyAccess`. (For more information, see [Create a role using custom trust policies]() in the IAM documentation.) 2. In the **Custom trust policy** section, add the following code:	DevOps engineer

Task	Description	Skills required

	``{
	    "Version":
	 "2012-10-17",
	    "Statement": [
	        {
	            "Effect":
	 "Allow",
	            "Principa
	l": {
	
	 "Service": "ec2.amaz
	onaws.com"
	            },
	            "Action":
	 "sts:AssumeRole"
	        },
	        {
	            "Effect":
	 "Allow",
	            "Principa
	l": {
	                "AWS":
	 "arn:aws:iam::${Pa
	rentaccountid}:rol
	e/ADcleanuprole"
	            },
	            "Action":
	 "sts:AssumeRole"
	        }
	    ]
	}```	

Task	Description	Skills required

Create an event rule in the child account.	To create an EventBridge rule for each child account, do the following:
	
	1. Sign in to the child AWS account, and then open the Amazon EventBridge console at https://console.aws.amazon.com/events/.
	2. In the navigation pane, choose **Rules**.
	3. Choose **Create rule**.
	4. Enter a **Name** and, optionally, a description for the rule.
	5. For **Event bus**, select **AWS default event bus**.
	6. For **Rule type**, choose **Rule with an event pattern**.
	7. Choose **Next**.
	8. For **Event pattern**, paste the following code:
	
	`{`
	`  "source": ["aws.ec2"],`
	`  "detail-type": ["EC2 Instance State-change Notification"],`
	`  "detail": {`
	`    "state": ["terminated"]`
	`  }`
		DevOps engineer

Task	Description	Skills required
	`}` 9. Choose **Next**. 10. For **Target type**, choose **Event bus in a different account or Region.** For **Event bus as target**, enter the event bus Amazon Resource Name (ARN) for the parent account. 11. For **Execution role**, choose **Create a new role for this specific resource.** 12. Choose **Next** to review the details for the new rule, and then choose **Create**. For more details, see [Creating rules that react to events in Amazon EventBridge] in the *Amazon EventBridge User Guide*.	

Task	Description	Skills required
Create an EC2 instance and join to AD.	To create an EC2 instance for Windows, do the following: 1. Use the `EC2WindowsUserdata` script that's available in this pattern's [code repository](#). 2. In the user data script, revise the following code to use your `Directory service addresses` values from the parent account: `set-DnsClientServerAddress -InterfaceIndex 6 -ServerAddresses $(Directory service addresses)`	DevOps engineer

Set up the local workstation

Task	Description	Skills required
Create a project folder and add the files.	To clone the repository and create a project folder, do the following: 1. Open this pattern's [GitHub repository](#). 2. Choose the **Code** button to see the options to clone in the **Clone** dropdown.	DevOps engineer

Task	Description	Skills required

	3. On the **HTTPS** tab, copy the URL provided in **Clone using the web URL**.
	4. Create a folder on your machine, and name it with your project name.
	5. Open a terminal in your local machine, and navigate to this folder.
	6. To clone the git repository, use the following command.
	`git clone <repository-URL>.git`
	7. After the repository has been cloned, use the following command to go to the cloned directory.
	`cd <directory name>/terraform-aws-lambda-ad-cleanup/multiple-account-cleanup`
	8. In the cloned repository, open this project in an integrated development environment (IDE) of your choice.	

Task	Description	Skills required
Build the `adcleanup.zip` file.	To compress the `lambda_function.py` file, run the following command: `zip -r adcleanup.zip lambda_function.py`	DevOps engineer

Provision the target architecture using the Terraform configuration

Task	Description	Skills required
Provide values for the Terraform variables.	For the child account, provide values for the following `arn` variables as string types in the `terraform.tfvars` file: • `lambda_env_cross_role_arn` • `child_account_cross_role_arn`	DevOps engineer
Initialize the Terraform configuration.	To initialize your working directory that contains the Terraform files, run the following command: `terraform init`	DevOps engineer
Preview changes.	You can preview the changes that Terraform will make to the infrastructure before your infrastructure is deployed. To validate that Terraform will	DevOps engineer

Task	Description	Skills required
	make the changes as required, run the following command: `terraform plan -- var-file=examples/ terraform.tfvars`	
Execute the proposed actions.	To verify that the results from the `terraform plan` command are as expected, do the following: 1. Run the following command: `terraform apply` 2. Sign in to the AWS Management Console, and verify that the resources are present.	DevOps engineer

Verify the deployment

Task	Description	Skills required
Execute and test the Lambda function.	To verify that the deploymen t occurred successfully, do the following: 1. Sign in to the AWS Management Console, and open the Lambda console. Open the **Functions** page, and select the function name that begins with **ADcleanup-Lambda-***.	DevOps engineer

Task	Description	Skills required
	2. On the function overview page, choose **Test** on the **Code** tab in the **Code source** section. 3. To save the test event, provide a name for the event and choose **Save**. To test the event, choose **Test** again. The execution results show the output of the function.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>View results of EventBridge rule execution from parent account.</td><td>To view the results of the EventBridge rule that’s based on Amazon EC2 terminati on events from the parent account, do the following:
Terminate EC2 instances from the parent account.
Open the Lambda console of the parent account. Open the Functions page, and select the function name that begins with ADcleanup-Lambda-*.
Choose the Monitor tab, and choose View CloudWatch logs.
In the CloudWatch console, the Log groups page shows the results of the Lambda function.</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
View results of EventBridge rule execution from the child account.	To view the results of the EventBridge rule that's based on Amazon EC2 termination events from the child account, do the following: 1. Terminate EC2 instances from the child account. 2. Open the Lambda console of the parent account. Open the **Functions** page, and select the function name that begins with **ADcleanup-Lambda-***. 3. Choose the **Monitor** tab, and choose **View CloudWatch logs**. In the CloudWatch console, the **Log groups** page shows the results of the Lambda function.	DevOps engineer

Clean up infrastructure after use

Task	Description	Skills required
Clean up the infrastructure.	To clean up the infrastructure that you created, use the following command: `terraform destroy`	DevOps engineer

Task	Description	Skills required
	To confirm the `destroy` command, type `yes`.	
Verify after cleanup.	Verify that the resources are successfully removed.	DevOps engineer

Troubleshooting

Issue	Solution
Connection issue between AWS Directory Service (parent account) and Amazon EC2 instance (child account) – You are unable to join the child account's computers to AD even though VPC peering is available.	Add routing in the VPCs. For instructions, see Configure a VPC peering connection between the directory owner and the directory consumer account in the AWS Directory Service documentation.

Related resources

AWS documentation

- Amazon EventBridge and AWS Identity and Access Management
- Configure instance permissions required for Systems Manager
- Identity and access management for AWS Directory Service
- Identity-based IAM policies for Lambda
- Manually join an Amazon EC2 Windows instance to your AWS Managed Microsoft AD Active Directory
- Remove Amazon EC2 entries in the same AWS account from AWS Managed Microsoft AD by using AWS Lambda automation

Other resources

- AWS Provider (Terraform documentation)
- Backend Configuration (Terraform documentation)

- Install Terraform (Terraform documentation)
- Python boto module (Python Package Index repository)
- Terraform binary download (Terraform documentation)

Remove Amazon EC2 entries in the same AWS account from AWS Managed Microsoft AD by using AWS Lambda automation

Dr. Rahul Sharad Gaikwad and Tamilselvan P, Amazon Web Services

Summary

Active Directory (AD) is a Microsoft scripting tool that manages domain information and user interactions with network services. It's widely used among managed services providers (MSPs) to manage employee credentials and access permissions. Because AD attackers can use inactive accounts to try and hack into an organization, it's important to find inactive accounts and disable them on a routine maintenance schedule. With AWS Directory Service for Microsoft Active Directory, you can run Microsoft Active Directory as a managed service.

This pattern can help you to configure AWS Lambda automation to quickly find and remove inactive accounts. When you use this pattern, you can get the following benefits:

- Improve database and server performance, and fix vulnerabilities in your security from inactive accounts.
- If your AD server is hosted in the cloud, removing inactive accounts can also reduce storage costs while improving performance. Your monthly bills might decrease because bandwidth charges and compute resources can both drop.
- Keep potential attackers at bay with a clean Active Directory.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Git installed and configured on a local workstation.
- Terraform installed and configured on a local workstation.
- Windows computer with Active Directory modules (`ActiveDirectory`).
- A directory in AWS Managed Microsoft AD and credentials stored in a parameter in AWS Systems Manager Parameter Store.
- AWS Identity and Access Management (IAM) role with permissions to the AWS services listed in Tools. For more information about IAM, see Related resources.

Limitations

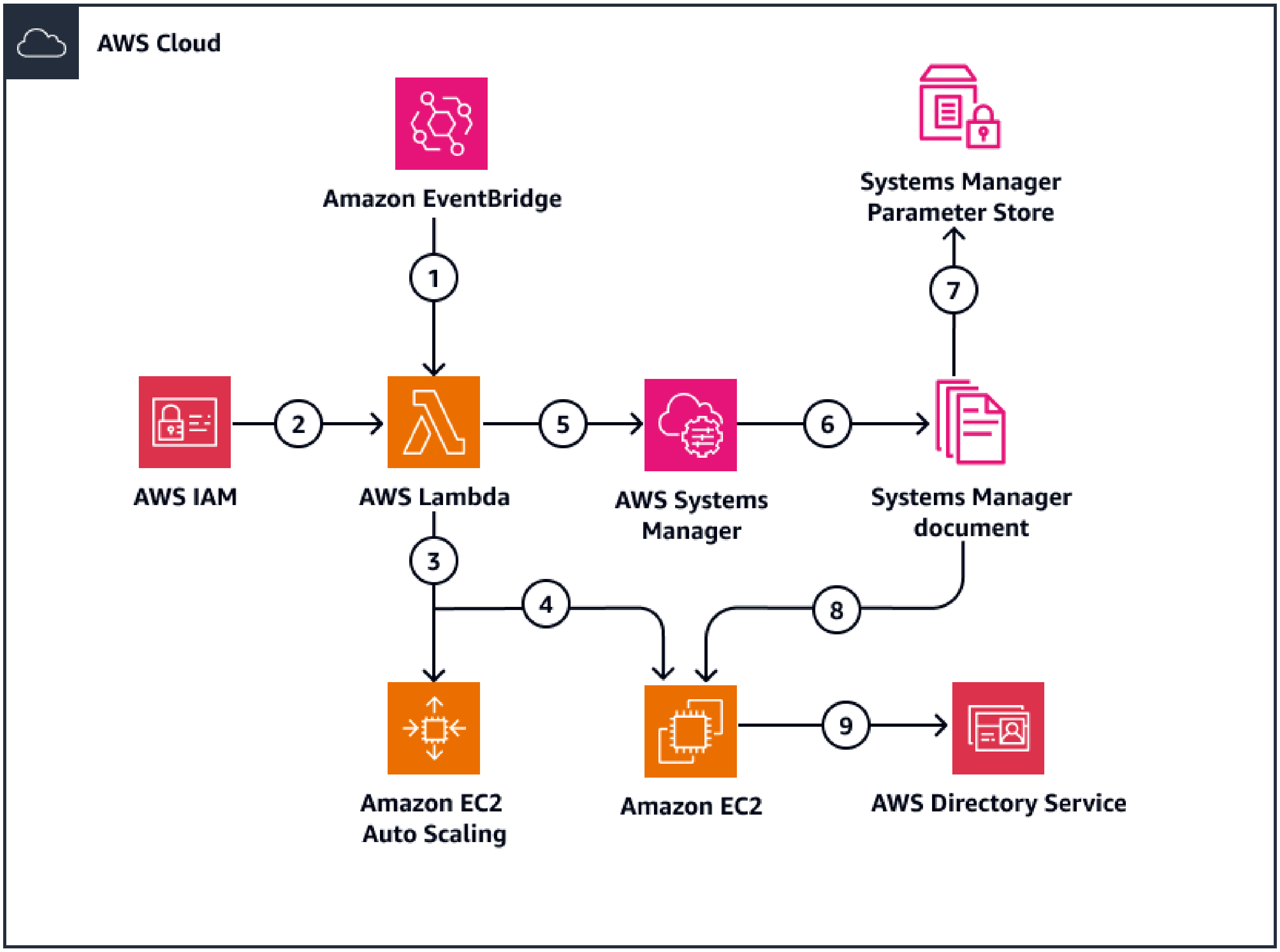
- This pattern doesn't support cross-account setup.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- Terraform version 1.1.9 or later
- Terraform AWS Provider version 3.0 or higher

Architecture

The following diagram shows the workflow and architecture components for this pattern.



The diagram shows the following workflow:

1. Amazon EventBridge triggers the AWS Lambda function based on a cron expression. (For this pattern, the cron expression schedule is once per day.)
2. The required IAM role and policy are created and attached to AWS Lambda through Terraform.
3. The AWS Lambda function is executed and calls to Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling Groups by using the Python boto module. The Lambda function gets the random instance id. The instance id is used to execute AWS Systems Manager commands.
4. AWS Lambda makes another call to Amazon EC2 using the boto module and gets the private IP addresses of the running Windows servers and stores the addresses in a temporary variable.
5. AWS Lambda makes another call to Systems Manager to get the computer information that is connected to AWS Directory Service.

6. An AWS Systems Manager document helps to execute the PowerShell script on Amazon EC2 Windows servers to get the private IP addresses of the computers which are connected with AD.
7. The AD domain username and passwords are stored in the AWS Systems Manager Parameter Store. AWS Lambda and Systems Manager make a call to Parameter Store and get the username and password values to use to connect AD.
8. Using the Systems Manager document, the PowerShell script is executed on the Amazon EC2 Windows server using the instance id obtained earlier in step 3.
9. Amazon EC2 connects AWS Directory Service by using PowerShell commands and removes the computers which are not in use or inactive.

Tools

AWS services

- AWS Directory Service provides multiple ways to use Microsoft Active Directory (AD) with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Relational Database Service (Amazon RDS) for SQL Server, and Amazon FSx for Windows File Server.
- AWS Directory Service for Microsoft Active Directory enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon EventBridge is a serverless event bus service that that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWSresources by controlling who is authenticated and authorized to use them. With IAM, you can specify who or what can access services and resources in AWS, centrally manage fine-grained permissions, and analyze access to refine permissions across AWS.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.

- [AWS Systems Manager documents](#) define the actions that Systems Manager performs on your managed instances. Systems Manager includes more than 100 pre-configured documents that you can use by specifying parameters at runtime.
- [AWS Systems Manager Parameter Store](#) is a capability of AWS Systems Manager and provides secure, hierarchical storage for configuration data management and secrets management.

Other tools

- [HashiCorp Terraform](#) is an open source infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources.
- [PowerShell](#) is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS.
- [Python](#) is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub [Custom AD Cleanup Automation solution](#) repository.

Best practices

- **Automatically join domains.** When you launch a Windows instance that's to be part of an AWS Directory Service domain, join the domain during the instance creation process instead of manually adding the instance later. To automatically join a domain, select the correct directory from the **Domain join directory** dropdown list when launching a new instance. For more details, see [Seamlessly join an Amazon EC2 Windows instance to your AWS Managed Microsoft AD Active Directory](#) in the *AWS Directory Service Administration Guide*.
- **Delete unused accounts.** It's common to find accounts in AD that have never been used. Like disabled or inactive accounts that remain in the system, neglected unused accounts can slow down your AD system or make your organization vulnerable to data breaches.
- **Automate Active Directory cleanups.** To help mitigate security risks and prevent obsolete accounts from impacting AD performance, conduct AD cleanups should at regular intervals. You can accomplish most AD management and cleanup tasks by writing scripts. Example tasks include removing disabled and inactive accounts, deleting empty and inactive groups, and locating expired user accounts and passwords.

Epics

Set up your environment

Task	Description	Skills required

Create a project folder, and add the files.	To clone the repository and create a project folder, do the following:
	
	1. Open this pattern’s [GitHub repository](#).
	2. Choose the **Code** button to see the options to clone in the **Clone** dropdown.
	3. On the **HTTPS** tab, copy the URL provided in **Clone using the web URL**.
	4. Create a folder on your machine, and name it with your project name.
	5. Open a terminal in your local machine, and navigate to this folder.
	6. To clone the git repository, use the following command.
	`git clone <repository-URL>.git`
	7. After the repository has been cloned, use the following command to go to the cloned directory.
	`cd <directory name>`
		DevOps engineer

Task	Description	Skills required
	8. In the cloned repository, open this project in an integrated development environment (IDE) of your choice.	

Provision the target architecture by using the Terraform configuration

Task	Description	Skills required
Initialize the Terraform configuration.	To initialize your working directory that contains the Terraform files, run the following command. `terraform init`	DevOps engineer
Preview changes.	You can preview the changes that Terraform will make to the infrastructure before your infrastructure is deployed. To validate that Terraform will make the changes as required, run the following command. `terraform plan`	DevOps engineer
Execute the proposed actions.	To verify that the results from the `terraform plan` command are as expected, do the following: 1. Run the following command. `terraform apply`	DevOps engineer

Task	Description	Skills required
	2. Sign in to the AWS Management Console, and verify that the resources are present.	
Clean up the infrastructure.	To clean up the infrastructure that you created, use the following command. `terraform destroy` To confirm the destroy command, type `yes`.	DevOps engineer

Verify the deployment

Task	Description	Skills required
Execute and test the Lambda function.	To verify that the deploymen t occurred successfully, do the following: 1. Sign in to the AWS Management Console and open the console. Open the **Functions** page, and select the function name that begins with **ADcleanup-Lambda-***. 2. On the function overview page, choose **Test** on the **Code** tab in the **Code source** section. 3. To save the test event, provide a name for the	DevOps engineer

Task	Description	Skills required
	event and choose **Save**. Then to test the event, choose **Test** again. The execution results show the output of the function.	
View the results of the Lambda function.	In this pattern, an EventBrid ge rule executes the Lambda function once per day. To view the results of the Lambda function, do the following: 1. Sign in to the AWS Management Console and open the AWS Lambda console. Open the **Functions** page and select the function name that begins with **ADcleanup-Lambda-***. 2. Choose the **Monitor** tab and choose **View CloudWatch logs**. In the CloudWatch console, the **Log groups** page shows the results of the Lambda function.	DevOps engineer

Clean up infrastructure after use

Task	Description	Skills required
Clean up infrastructure.	To clean up the infrastructure that you created, use the following command. `terraform destroy` To confirm the destroy command, type `yes`.	DevOps engineer
Verify after cleanup.	Verify that the resources are successfully removed.	DevOps engineer

Troubleshooting

Issue	Solution
If you try to remove the AD computer, you get an "Access Denied" message. The AD computer can't be removed because, by default, the action tries to remove two private IP addresses which are connected as a part of the AD services.	To avoid this error, use the following Python operation to ignore the first two computers when you list the differences between an AD computer output and the output of your machine running Windows. `Difference = Difference[2:]`
When Lambda executes a PowerShell script on a Windows server, it expects Active Directory modules to be available by default. If the modules are not available, a Lambda function creates an error that states "Get-AdComputer is not installed on instance".	To avoid this error, install the required modules by using the user data of the EC2 instances. Use the EC2WindowsUserdata script that's in this pattern's GitHub repository.

Related resources

AWS documentation

- Amazon EventBridge and AWS Identity and Access Management
- Configure instance permissions required for Systems Manager
- Identity and access management for AWS Directory Service
- Manually join an Amazon EC2 Windows instance to your AWS Managed Microsoft AD Active Directory
- Working with identity-based IAM policies in AWS Lambda

Other resources

- AWS Provider (Terraform documentation)
- Backend Configuration (Terraform documentation)
- Install Terraform (Terraform documentation)
- Python boto module (Python Package Index repository)
- Terraform binary download (Terraform documentation)

Run unit tests for Python ETL jobs in AWS Glue using the pytest framework

Praveen Kumar Jeyarajan and Vaidy Sankaran, Amazon Web Services

Summary

You can run unit tests for Python extract, transform, and load (ETL) jobs for AWS Glue in a local development environment, but replicating those tests in a DevOps pipeline can be difficult and time consuming. Unit testing can be especially challenging when you're modernizing mainframe ETL process on AWS technology stacks. This pattern shows you how to simplify unit testing, while keeping existing functionality intact, avoiding disruptions to key application functionality when you release new features, and maintaining high-quality software. You can use the steps and code samples in this pattern to run unit tests for Python ETL jobs in AWS Glue by using the pytest framework in AWS CodePipeline. You can also use this pattern to test and deploy multiple AWS Glue jobs.

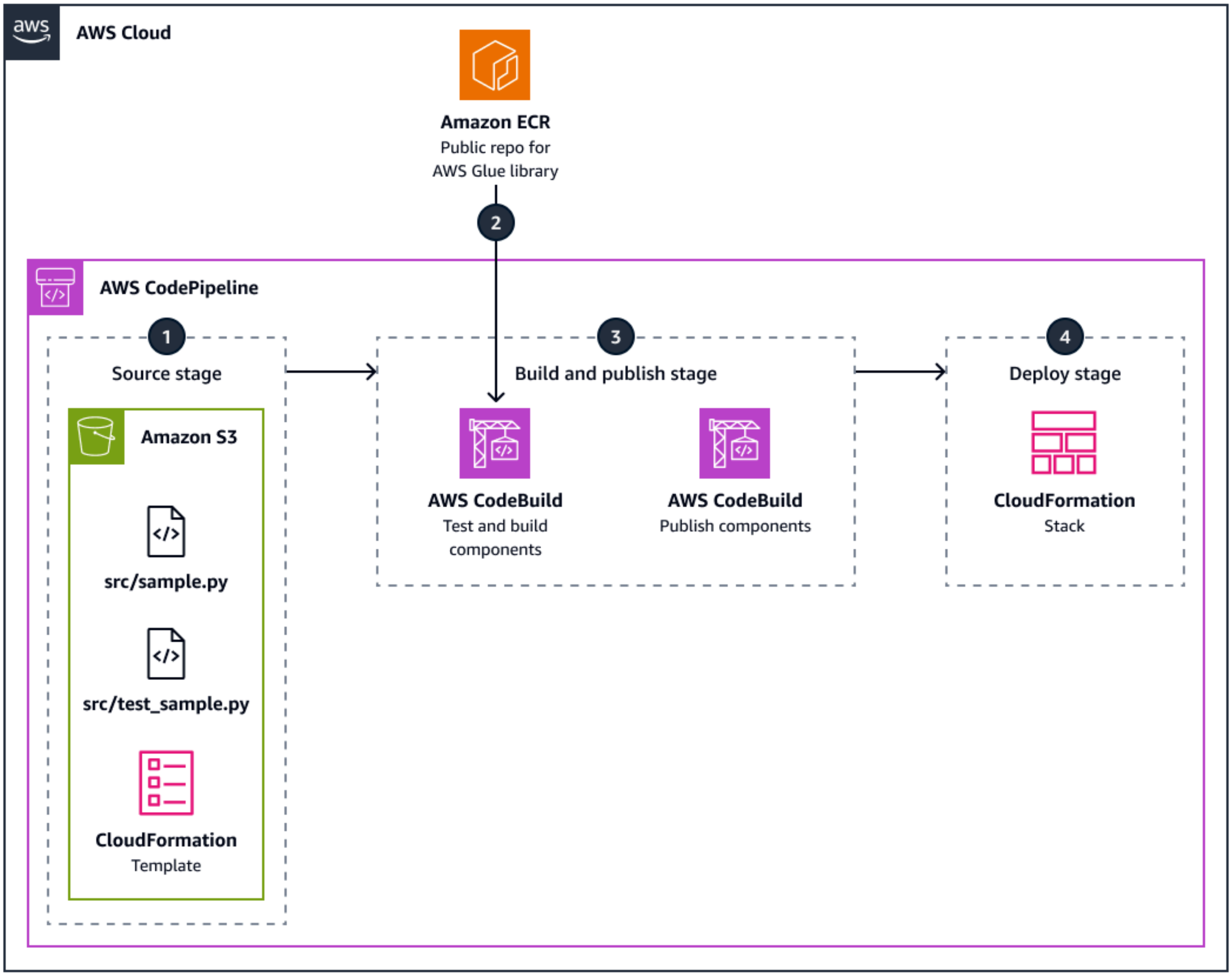
Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Elastic Container Registry (Amazon ECR) image URI for your AWS Glue library, downloaded from the Amazon ECR Public Gallery
- Bash terminal (on any operating system) with a profile for the target AWS account and AWS Region
- Python 3.10 or later
- Pytest
- Moto Python library for testing AWS services

Architecture

The following diagram describes how to incorporate unit testing for AWS Glue ETL processes that are based on Python into a typical enterprise-scale AWS DevOps pipeline.



The diagram shows the following workflow:

1. In the source stage, AWS CodePipeline uses a versioned Amazon Simple Storage Service (Amazon S3) bucket to store and manage source code assets. These assets include a sample Python ETL job (`sample.py`), a unit test file (`test_sample.py`), and an AWS CloudFormation template. Then, CodePipeline transfers the most recent code from the main branch to the AWS CodeBuild project for further processing.
2. In the build and publish stage, the most recent code from the previous source stage is unit tested with the help of an AWS Glue public Amazon ECR image. Then, the test report is published to CodeBuild report groups. The container image in the public Amazon ECR repository for AWS Glue libraries includes all the binaries required to run and unit test [PySpark-based](#) ETL tasks in AWS Glue locally. The public container repository has three image tags, one for

each version supported by AWS Glue. For demonstration purposes, this pattern uses the `glue_libs_4.0.0_image_01` image tag. To use this container image as a runtime image in CodeBuild, copy the image URI that corresponds to the image tag that you intend to use, and then update the `pipeline.yml` file in the GitHub repository for the `TestBuild` resource.

3. In the deploy stage, the CodeBuild project is launched and it publishes the code to an Amazon S3 bucket if all the tests pass.
4. The user deploys the AWS Glue task by using the CloudFormation template in the `deploy` folder.

Tools

AWS services

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- AWS Glue is a fully managed ETL service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance.

Other tools

- Python is a high-level, interpreted general purpose programming language.
- Moto is a Python library for testing AWS services.
- Pytest is a framework for writing small unit tests that scale to support complex functional testing for applications and libraries.
- Python ETL library for AWS Glue is a repository for Python libraries that are used in the local development of PySpark batch jobs for AWS Glue.

Code repository

The code for this pattern is available in the GitHub aws-glue-jobs-unit-testing repository. The repository includes the following resources:

- A sample Python-based AWS Glue job in the `src` folder
- Associated unit test cases (built using the pytest framework) in the `tests` folder
- A CloudFormation template (written in YAML) in the `deploy` folder

Best practices

Security for CodePipeline resources

It’s a best practice to use encryption and authentication for the source repositories that connect to your pipelines in CodePipeline. For more information, see Security best practices in the CodePipeline documentation.

Monitoring and logging for CodePipeline resources

It’s a best practice to use AWS logging features to determine what actions users take in your account and what resources they use. The log files show the following:

- Time and date of actions
- Source IP address of actions
- Which actions failed due to inadequate permissions

Logging features are available in AWS CloudTrail and Amazon CloudWatch Events. You can use CloudTrail to log AWS API calls and related events made by or on behalf of your AWS account. For more information, see Logging CodePipeline API calls with AWS CloudTrail in the CodePipeline documentation.

You can use CloudWatch Events to monitor your AWS Cloud resources and applications running on AWS. You can also create alerts in CloudWatch Events. For more information, see Monitoring CodePipeline events in the CodePipeline documentation.

Epics

Deploy the source code

Task	Description	Skills required

Prepare the code archive for deployment.	1. Download `code.zip` from the GitHub [aws-glue-jobs-unit-testing](https://github.com/aws-samples/aws-glue-jobs-unit-testing) repository, or create the .zip file yourself by using a command-line tool. For example, you can create the .zip file on Linux or Mac by running the following commands in the terminal:
	`git clone https://github.com/aws-samples/aws-glue-jobs-unit-testing.git`
	`cd aws-glue-jobs-unit-testing`
	`git checkout master`
	`zip -r code.zip src/ tests/ deploy/`
	2. Sign in to the AWS Management Console and choose the AWS Region of your choice.
	3. Create an Amazon S3 bucket, and then upload the .zip package and `code.zip` file (downloaded earlier) to the Amazon S3 bucket that you created.
		DevOps engineer

Task	Description	Skills required
Create the CloudFormation stack.	1. Sign in to the AWS Management Console and then open the [CloudFormation console](#). 2. Choose **Create stack**, and then choose **With existing resources (import resources)**. 3. In the **Specify template** section of the **Create stack** page, choose **Upload a template file**, and then choose the **pipeline.yml** template (downloaded from the GitHub repository). Then, choose **Next**. 4. For **Stack name**, enter **glue-unit-testing-pipeline**, or choose a stack name of your choice. 5. For **ApplicationStackName**, use the prepopulated **glue-codepipeline-app** name. This is the name of the CloudFormation stack that's created by the pipeline. 6. For **BucketName**, use the prepopulated **aws-glue-artifacts-us-east-1** bucket name. This is the name of the Amazon S3 bucket that contains the .zip file and	AWS DevOps, DevOps engineer

Task	Description	Skills required
	is used by the pipeline to store code artifacts. 7. For **CodeZipFile**, use the prepopulated **code.zip** value. This is the key name of the sample code Amazon S3 object. The object should be a .zip file. 8. For **TestReportGroupName**, use the prepopulated **glue-unittest-report** name. This is the name of the CodeBuild test report group that's created to store the unit test reports. 9. Choose **Next**, and then choose **Next** again on the **Configure stack options** page. 10. On the **Review** page, under **Capabilities**, choose the **I acknowledge that CloudFormation might create IAM resources with custom names** option. 11. Choose **Submit**. After the stack creation is complete, you can see the created resources on the **Resources** tab. The stack creation takes approximately 5-7 minutes.	

Task	Description	Skills required
	The stack creates a CodePipeline view using Amazon S3 as the source. In the steps above, the pipeline is **aws-glue-unit-test-pipeline**.	

Run the unit tests

Task	Description	Skills required
Run the unit tests in the pipeline.	1. To test the deployed pipeline, sign in to the AWS Management Console, and then open the [CodePipeline console](). 2. Select the pipeline created by the CloudFormation stack, and then choose **Release change**. The pipeline starts running (using the most recent code in the Amazon S3 bucket). 3. After the **Test_and_Build** phase is finished, choose the **Details** tab, and then examine the logs. 4. Choose the **Reports** tab, and then choose the test report from **Report history** to view the unit test results.	AWS DevOps, DevOps engineer

Task	Description	Skills required
	5. After the deployment stage is complete, run and monitor the deployed AWS Glue job on the AWS Glue console. For more information, see [Monitoring AWS Glue]() in the AWS Glue documentation.	

Clean up all AWS resources

Task	Description	Skills required
Clean up the resources in your environment.	To avoid additional infrastructure costs, make sure that you delete the stack after experimenting with the examples provided in this pattern. 1. Open the [CloudFormation console](), and then select the stack that you created. 2. Choose **Delete**. This deletes all the resources that your stack created, including AWS Identity and Access Management (IAM) roles, IAM policies, and CodeBuild projects.	AWS DevOps, DevOps engineer

Troubleshooting

Issue	Solution
The CodePipeline service role cannot access the Amazon S3 bucket.	• For the policy attached to your CodePipeline service role, add `s3:ListBucket` to the list of actions in your policy. For instructions on viewing the service role policy, see View the pipeline ARN and service role ARN (console). Edit the policy statement for your service role as detailed in Add permissions to the CodePipeline service role. • For the resource-based policy that is attached to the Amazon S3 artifact bucket for your pipeline, also called the *artifact bucket policy*, add a statement that allows the CodePipeline service role to use the `s3:ListBucket` permission.
CodePipeline returns an error that the Amazon S3 bucket is not versioned.	CodePipeline requires that the source Amazon S3 bucket be versioned. Enable versioning on your Amazon S3 bucket. For instructions, see Enabling versioning on buckets.

Related resources

- AWS Glue
- Developing and testing AWS Glue jobs locally
- AWS CloudFormation for AWS Glue

Additional information

Additionally, you can deploy the AWS CloudFormation templates by using the AWS Command Line Interface (AWS CLI). For more information, see Quickly deploying templates with transforms in the CloudFormation documentation.

Set up a Helm v3 chart repository in Amazon S3

Abhishek Sharma, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern helps you to manage Helm v3 charts efficiently by integrating the Helm v3 repository into Amazon Simple Storage Service (Amazon S3) on the Amazon Web Services (AWS) Cloud. To use this pattern, you must be familiar with Kubernetes and with Helm, which is a Kubernetes package manager. Using Helm repositories to store charts and control chart versions can improve mean time to restore (MTTR) during outages.

This pattern uses AWS CodeCommit for Helm repository creation, and it uses an S3 bucket as a Helm chart repository, so that the charts can be centrally managed and accessed by developers across the organization.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Python version 2.7.12 or later
- pip
- A virtual private cloud (VPC) with subnets and an Amazon Elastic Compute Cloud (Amazon EC2) instance
- Git installed on the EC2 instance
- AWS Identity and Access Management (IAM) access to create the S3 bucket
- IAM (programmatic or role) access to Amazon S3 from the client machine
- AWS CodeCommit repository
- AWS Command Line Interface (AWS CLI)

Product versions

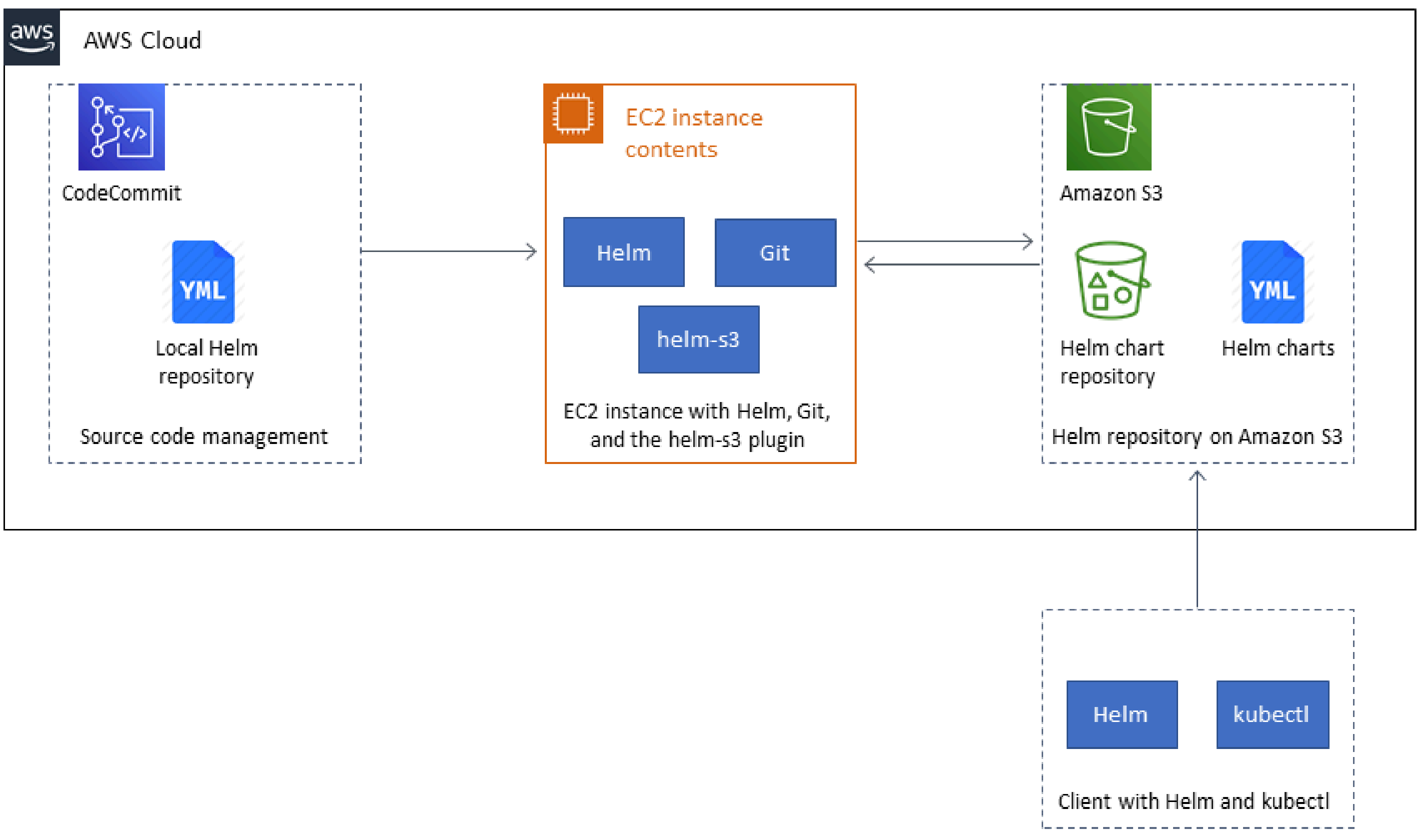
- Helm v3
- Python version 2.7.12 or later

Architecture

Target technology stack

- Amazon S3
- AWS CodeCommit
- Helm
- Kubectl
- Python and pip
- Git
- helm-s3 plugin

Target architecture



Automation and scale

- You can incorporate Helm into your existing continuous integration/continuous delivery (CI/CD) automation tool to automate the packaging and version control of Helm charts (out of scope for this pattern).
- GitVersion or Jenkins build numbers can be used to automate version control of the charts.

Tools

- Helm – Helm is a package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- helm-s3 plugin – The helm-s3 plugin supports interaction with Amazon S3. It can be used with either Helm v2 or Helm v3.

Epics

Install and validate Helm v3

Task	Description	Skills required
Install the Helm v3 client.	To download and install the Helm client on your local system, run the following command: `sudo curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash`	Cloud Administrator, DevOps Engineer
Validate the Helm installation.	To validate the Helm client, run the following command: `helm version --short`	Cloud Administrator, DevOps Engineer

Initialize an S3 bucket as a Helm repository

Task	Description	Skills required
Create an S3 bucket for Helm charts.	Create a unique S3 bucket. In the bucket, create a folder called `stable/myapp` . The example in this pattern uses `s3://my-helm-charts/stable/myapp` as the target chart repository.	Cloud Administrator, DevOps Engineer
Install the helm-s3 plugin for Amazon S3.	To install the helm-s3 plugin on your client machine, run the following command: `helm plugin install https://github.com/hypnoglow/helm-s3.git`	Cloud Administrator, DevOps Engineer
Initialize the Amazon S3 Helm repository.	To initialize the target folder as a Helm repository, use the following command: `helm s3 init s3://my-helm-charts/stable/myapp` The command creates an `index.yaml` file in the target to track all the chart information that is stored at that location.	Cloud Administrator, DevOps Engineer
Verify the newly created Helm repository.	To verify that the `index.yaml` file was created, run the following command: `aws s3 ls s3://my-helm-charts/stable/myapp/`	Cloud Administrator, DevOps Engineer

Task	Description	Skills required
Add the Amazon S3 repository to Helm on the client machine.	To add the target repository alias to the Helm client machine, use the following command: `helm repo add stable-myapp s3://my-helm-charts/stable/myapp/`	Cloud Administrator, DevOps Engineer

Package and publish charts in the Amazon S3 Helm repository

Task	Description	Skills required
Clone your Helm charts.	If no local Helm charts are present on in your CodeCommit repository, clone them from your GitHub repo by running the following command: `git clone <url_of_your_helm_source_code>.git`	Cloud Administrator, DevOps Engineer
Package the local Helm chart.	To package the chart that you created or cloned, use the following command: `helm package ./my-app` As an example, this pattern uses the `my-app` chart. The command packages all the contents of the `my-app` chart folder into an archive file, which is named using the version number that is	Cloud Administrator, DevOps Engineer

Task	Description	Skills required
	mentioned in the `Chart.yaml` file.	
Store the local package in the Amazon S3 Helm repository.	To upload the local package to the Helm repository in Amazon S3, run the following command: `helm s3 push ./my-app-0.1.0.tgz stable-myapp` In the command, `my-app` is your chart folder name, `0.1.0` is the chart version mentioned in `Chart.yaml`, and `stable-myapp` is the target repository alias.	Cloud Administrator, DevOps Engineer
Search for the Helm chart.	To confirm that the chart appears both locally and in the Amazon S3 Helm repository, run the following command: `helm search repo stable-myapp`	Cloud Administrator, DevOps Engineer

Upgrade your Helm repository

Task	Description	Skills required
Modify and package the chart.	In `values.yaml`, set the `replicaCount` value to 1, and then package the chart, this time changing the version in `Chart.yaml` to `0.1.1`. Version control is ideally	Cloud Administrator, DevOps Engineer

Task	Description	Skills required
	achieved through automation by using tools like GitVersion or Jenkins build numbers in a CI/CD pipeline. Automating the version number is out of scope for this pattern. To package the chart, run the following command: `helm package ./my-app/`	
Push the new version to the Helm repository in Amazon S3.	To push the new package, version of 0.1.1, to the *my-helm-charts* Helm repository in Amazon S3, run the following command: `helm s3 push ./my-app-0.1.1.tgz stable-myapp`	Cloud Administrator, DevOps Engineer

Verify the updated Helm chart.	To confirm that the updated chart appears both locally and in the Amazon S3 Helm repository, run the following commands.
	
	`helm repo update`
	
	`helm search repo stable-myapp`
		Cloud Administrator, DevOps Engineer

Search for and install a chart from the Amazon S3 Helm repository

Task	Description	Skills required
Search for all versions of the my-app chart.	To view all the available versions of a chart, run the following command with the `--versions` flag: `helm search repo my-app --versions` Without the flag, Helm by default displays the latest uploaded version of a chart.	DevOps Engineer
Install a chart from the Amazon S3 Helm repository.	Automated installation is out of scope for this pattern, but you can manually install. The search results from the previous task show the multiple versions of the `my-app` chart. To install the new version (0.1.1) from the Amazon S3 Helm repository, use the following command: `helm upgrade --install my-app-release stable-myapp/my-app --version 0.1.1 --namespace dev`	DevOps Engineer

Roll back to a previous version by using Helm

Task	Description	Skills required
Review the details for a specific revision.	Automated rollback is out of scope for this pattern,	DevOps Engineer

Task	Description	Skills required
	but you can roll back to an earlier version manually. Before you switch or roll back to a working version, and for an additional layer of validation before installin g a revision, view which values were passed to each of the revisions by using the following command: `helm get values --revisio n=2 my-app-release`	
Roll back to a previous version.	Automated rollback is out of scope for this pattern. To manually roll back to a previous revision, use the following command: `helm rollback my-app-re lease 1` This example is rolling back to revision number 1.	DevOps Engineer

Related resources

- HELM documentation
- helm-s3 plugin (MIT License)
- Amazon S3

Set up a CI/CD pipeline by using AWS CodePipeline and AWS CDK

Konstantin Zarudaev, Yasha Dabas, Lars Kinder, and Cizer Pereira, Amazon Web Services

Home

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more Automating your software build and release process with continuous integration and continuous delivery (CI/CD) supports repeatable builds and rapid delivery of new features to your users. You can quickly and easily test each code change, and you can catch and fix bugs before releasing your software. By running each change through your staging and release process, you can verify the quality of your application or infrastructure code. CI/CD embodies a culture, a set of operating principles, and a collection of practices that help application development teams to deliver code changes more frequently and reliably. The implementation is also known as the *CI/CD pipeline*.

This pattern defines a reusable continuous integration and continuous delivery (CI/CD) pipeline on Amazon Web Services (AWS) with an AWS CodeCommit repository. The AWS CodePipeline pipeline is written using AWS Cloud Development Kit (AWS CDK) v2.

Using CodePipeline, you can model the different stages of your software release process through the AWS Management Console interface, the AWS Command Line Interface (AWS CLI), AWS CloudFormation, or the AWS SDKs. This pattern demonstrates the implementation of CodePipeline and its components using AWS CDK. In addition to construct libraries, AWS CDK includes a toolkit (the CLI command `cdk`), which is the primary tool for interacting with your AWS CDK app. Among other functions, the toolkit provides the ability to convert one or more stacks to CloudFormation templates and deploy them to an AWS account.

The pipeline includes tests to validate the security of your third-party libraries, and it helps ensure expedited, automated release in the specified environments. You can increase the overall security of your applications by putting them through a validation process.

The intent of this pattern is to accelerate your use of CI/CD pipelines to deploy your code while ensuring the resources you deploy adhere to DevOps best practices. After you implement the example code, you will have an AWS CodePipeline with linting, testing, a security check, deployment, and post-deployment processes. This pattern also includes steps for Makefile. Using a Makefile, developers can reproduce CI/CD steps locally and increase the velocity of the development process.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A basic understanding in the following:
  - AWS CDK
  - AWS CloudFormation
  - AWS CodePipeline
  - TypeScript

Limitations

This pattern uses AWS CDK for TypeScript only. It doesn't cover other languages supported by AWS CDK.

Product versions

Use the latest versions of the following tools:

- AWS Command Line Interface (AWS CLI)
- cfn_nag
- git-remote-codecommit
- Node.js

Architecture

Target technology stack

- AWS CDK
- AWS CloudFormation
- AWS CodeCommit
- AWS CodePipeline

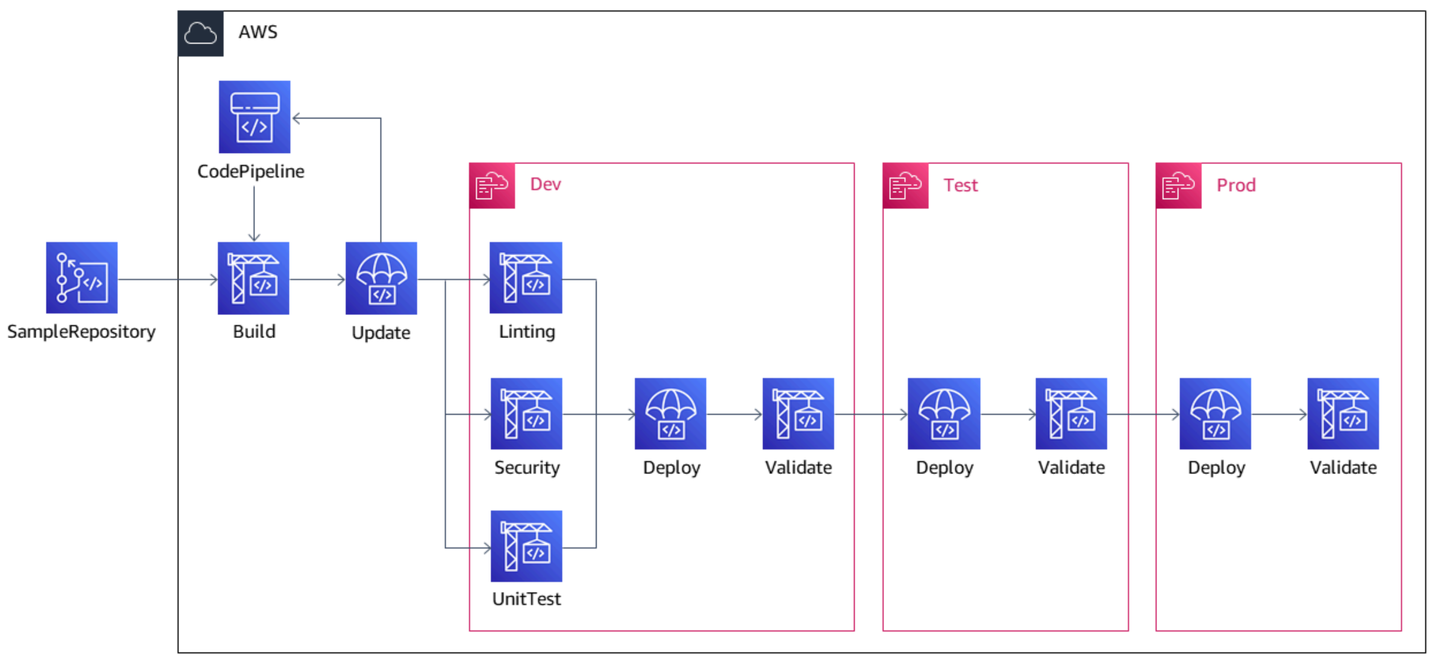
Target architecture

The pipeline is triggered by a change in the AWS CodeCommit repository (`SampleRepository`). In the beginning, CodePipeline builds artifacts, updates itself, and starts the deployment process. The resulting pipeline deploys a solution to three independent environments:

- Dev – Three-step code check in the active development environment
- Test – Integration and regression test environment
- Prod – Production environment

The three steps included in the Dev stage are linting, security, and unit tests. These steps run in parallel to speed up the process. To ensure that the pipeline provides only working artifacts, it will be stop running whenever a step in the process fails. After a Dev stage deployment, the pipeline runs validation tests to verify the results. In the case of success, the pipeline will then deploy the artifacts to the Test environment, which contains post-deployment validation. The final step is to deploy the artifacts to the Prod environment.

The following diagram shows the workflow from the CodeCommit repository to the build and update processes performed by CodePipeline, the three Dev environment steps, and subsequent deployment and validation in each of the three environments.



Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions. In this pattern CloudFormation templates can be used to create a CodeCommit repository and a CodePipeline CI/CD pipeline.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS CodePipeline is a CI/CD service that helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.

Other tools

- cfn_nag is an open-source tool that looks for patterns in CloudFormation templates to identify potential security issues.
- git-remote-codecommit is a utility for pushing and pulling code from CodeCommit repositories by extending Git.
- Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.

Code

The code for this pattern is available in the GitHub AWS CodePipeline with CI/CD practices repository.

Best practices

Review resources, such as AWS Identity and Access Management (IAM) policies, to confirm that they align with your organizational best practices.

Epics

Install tools

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install tools on macOS or Linux.</td><td>If you are using MacOS or Linux, you can install the tools by running the following command in your preferred terminal or using Homebrew for Linux.<pre><code>brew install
brew install git-remote-codecommit
brew install ruby brew-gem
brew-gem install cfn-nag</code></pre></td><td>DevOps engineer</td></tr>
<tr><td>Set up AWS CLI.</td><td>To set up AWS CLI, use the instructions for your operating system:
Windows: Setup steps for HTTPS connections to AWS CodeCommit repositories on Windows with the AWS CLI credential helper
Linux, macOS, Unix: Setup steps for HTTPS connections to AWS CodeCommit repositories on Linux, macOS, or Unix with the AWS CLI credential helper</td><td>DevOps engineer</td></tr>
</table>

Set up the initial deployment

Task	Description	Skills required

Download or clone the code.	To get the code that is used by this pattern, do one of the following:
	
	• Download the latest source code from releases in the GitHub repo, and unzip the downloaded file into a folder.
	• Clone the project by running the following command.
	
	`git clone --depth 1 https://github.com/aws-samples/aws-codepipeline-cicd.git`
	
	Remove the `.git` directory from the cloned repository.
	
	`cd ./aws-codepipeline-cicd`
	`rm -rf ./.git`
	
	Later, you will use a newly created AWS CodeCommit repository as a remote origin.
		DevOps engineer

Connect to the AWS account.	You can connect by using a temporary security token or landing zone authentication. To confirm that you are using	DevOps engineer

Task	Description	Skills required
	the correct account and AWS Region, run the following commands. ``` AWS_REGION="eu-west-1" ACCOUNT_NUMBER=$(aws sts get-caller-identity --query Account --output text) echo "${ACCOUNT_NUMBER}" ```	
Bootstrap the environment.	To bootstrap an AWS CDK environment, run the following commands. ``` npm install npm run cdk bootstrap "aws://${ACCOUNT_NUMBER}/${AWS_REGION}" ``` After you successfully bootstrap the environment, the following output should be displayed. ``` # Bootstrapping environment aws://{account}/{region}... # Environment aws://{account}/{region} bootstrapped ``` For more information about AWS CDK bootstrapping, see the [AWS CDK documentation]().	DevOps engineer

Task	Description	Skills required

Synthesize a template.	To synthesize an AWS CDK app, use the `cdk synth` command.
	`npm run cdk synth`
	You should see the following output.
	`Successfully synthesized to <path-to-directory>/aws-codepipeline-cicd/cdk.out`
	`Supply a stack id (CodePipeline, Dev-MainStack) to display its template.`
		DevOps engineer

Task	Description	Skills required

Deploy the CodePipeline stack.	Now that you bootstrapped and synthesized the CloudFormation template, you can deploy it. The deployment will create the CodePipeline pipeline and a CodeCommit repository, which will be the source and trigger of the pipeline.
	
	`npm run cdk -- deploy CodePipeline --require-approval never`
	
	After you run the command, you should see a successful deployment of the CodePipeline stack and output information. The `CodePipeline.RepositoryName` gives you the name of the CodeCommit repository in the AWS account.
	
	`CodePipeline: deploying ...`
	`CodePipeline: creating CloudFormation changeset...`
	`#  CodePipeline Outputs:`
	`CodePipeline.RepositoryName = SampleRepository`
	`Stack ARN:`
	`arn:aws:cloudformation:REGION:ACCOUNT-ID`
		DevOps engineer

Task	Description	Skills required
	`:stack/CodePipeline/ STACK-ID`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Set up the remote CodeCommit repository and branch.</td><td>After a successful deployment, CodePipeline will initiate the first run of the pipeline, which you can find in the AWS CodePipeline console. Because AWS CDK and CodeCommit don’t initiate a default branch, this initial pipeline run will fail and return the following error message.
<pre><code>The action failed because no branch named main was found in the selected AWS CodeCommit repository SampleRepository. Make sure you are using the correct branch name, and then try again. Error: null</code></pre>
To fix this error, set up a remote origin as <code>SampleRepository</code> , and create the required <code>main</code> branch.
<pre><code>RepoName=$(aws cloudformation describe-stacks --stack-name CodePipeline --query "Stacks[0].Outputs[?OutputKey=='RepositoryName'].OutputValue" --output text)
echo "${RepoName}"
#</code></pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	``` git init git branch -m master main git remote add origin codecommit://${Rep oName} git add . git commit -m "Initial commit" git push -u origin main ```	

Test the deployed CodePipeline pipeline

Task	Description	Skills required
Commit a change to activate the pipeline.	After a successful initial deployment, you should have a complete CI/CD pipeline with a `main` branch for `SampleRepository` as a source branch. As soon as you commit changes to the `main` branch, the pipeline will initiate and run the following sequence of actions: 1. Get your code from the CodeCommit repository. 2. Build your code. 3. Update the pipeline itself (`UpdatePipeline`). 4. Run three parallel jobs for linting, security and unit test checks.	DevOps engineer

Task	Description	Skills required
	5. In the case of success, the pipeline will deploy the `Main` stack from `./lib/main-stack.ts` to the Dev environment. 6. Run a post-deployment check for deployed resources. You can follow all CodePipeline steps and results in the CodePipeline console. 7. In the case of success, the pipeline will repeat deployment and validation for the Test and Prod environments.	

Test locally by using a Makefile

Task	Description	Skills required
Run the development process by using a Makefile.	You can run the whole pipeline locally by using the `make` command, or you can run an individual step (for example, `make linting`). To test using `make`, perform the following actions: • Implement the local pipeline: `make` • Run only unit testing: `make unittest`	App developer, DevOps engineer

Task	Description	Skills required
	• Deploy to the current account: `make deploy` • Clean up the environment: `make clean`	

Clean up resources

Task	Description	Skills required
Delete AWS CDK app resources.	To clean up your AWS CDK app, run the following command. `cdk destroy --all` Be aware that the Amazon Simple Storage Service (Amazon S3) buckets that are created during bootstrapping aren't automatically deleted. They need a retention policy that allows deletion, or you need to delete them manually in your AWS account.	DevOps engineer

Troubleshooting

Issue	Solution
The template isn’t working as expected.	If something goes wrong and template is not working, make sure that you have the following: • The proper versions of the tools.

Issue	Solution
	• Access to the target AWS account (network connectivity). • Enough permissions to the target AWS account.

Related resources

- Get started with common tasks in IAM Identity Center
- AWS CodePipeline documentation
- AWS CDK

Set up centralized logging at enterprise scale by using Terraform

Aarti Rajput, Yashwant Patel, and Nishtha Yadav, Amazon Web Services

Summary

Centralized logging is vital for an organization's cloud infrastructure, because it provides visibility into its operations, security, and compliance. As your organization scales its AWS environment across multiple accounts, a structured log management strategy becomes fundamental for running security operations, meeting audit requirements, and achieving operational excellence.

This pattern provides a scalable, secure framework for centralizing logs from multiple AWS accounts and services, to enable enterprise-scale logging management across complex AWS deployments. The solution is automated by using Terraform, which is an infrastructure as code (IaC) tool from HashiCorp that ensures consistent and repeatable deployments, and minimizes manual configuration. By combining Amazon CloudWatch Logs, Amazon Data Firehose, and Amazon Simple Storage Service (Amazon S3), you can implement a robust log aggregation and analysis pipeline that delivers:

- Centralized log management across your organization in AWS Organizations
- Automated log collection with built-in security controls
- Scalable log processing and durable storage
- Simplified compliance reporting and audit trails
- Real-time operational insights and monitoring

The solution collects logs from Amazon Elastic Kubernetes Service (Amazon EKS) containers, AWS Lambda functions, and Amazon Relational Database Service (Amazon RDS) database instances through CloudWatch Logs. It automatically forwards these logs to a dedicated logging account by using CloudWatch subscription filters. Firehose manages the high-throughput log streaming pipeline to Amazon S3 for long-term storage. Amazon Simple Queue Service (Amazon SQS) is configured to receive Amazon S3 event notifications upon object creation. This enables integration with analytics services, including:

- Amazon OpenSearch Service for log search, visualization, and real-time analytics
- Amazon Athena for SQL-based querying
- Amazon EMR for large-scale processing
- Lambda for custom transformation

- Amazon QuickSight for dashboards

All data is encrypted by using AWS Key Management Service (AWS KMS), and the entire infrastructure is deployed by using Terraform for consistent configuration across environments.

This centralized logging approach enables organizations to improve their security posture, maintain compliance requirements, and optimize operational efficiency across their AWS infrastructure.

Prerequisites and limitations

Prerequisites

- A landing zone for your organization that's built by using AWS Control Tower
- Account Factory for Terraform (AFT), deployed and configured with required accounts
- Terraform for provisioning the infrastructure
- AWS Identity and Access Management (IAM) roles and policies for cross-account access

For instructions for setting up AWS Control Tower, AFT, and Application accounts, see the Epics section.

Required accounts

Your organization in AWS Organizations should include these accounts:

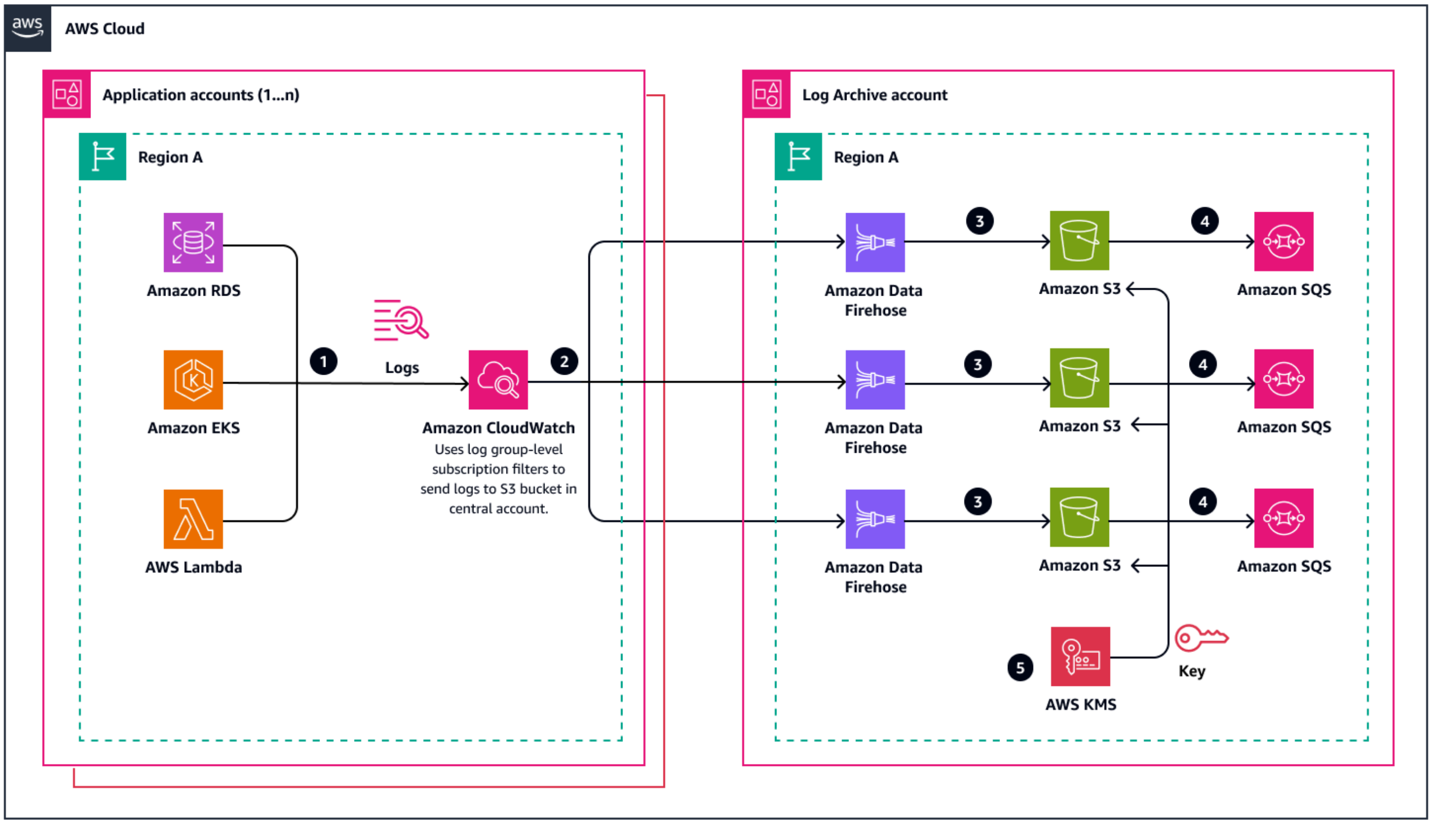
- **Application account** – One or more source accounts where the AWS services (Amazon EKS, Lambda, and Amazon RDS) run and generate logs
- **Log Archive account** – A dedicated account for centralized log storage and management

Product versions

- AWS Control Tower version 3.1 or later
- Terraform version 0.15.0 or later

Architecture

The following diagram illustrates an AWS centralized logging architecture that provides a scalable solution for collecting, processing, and storing logs from multiple Application accounts into a dedicated Log Archive account. This architecture efficiently handles logs from AWS services, including Amazon RDS, Amazon EKS, and Lambda, and routes them through a streamlined process to Regional S3 buckets in the Log Archive account.



The workflow includes five processes:

1. **Log flow process**
   - The log flow process begins in the Application accounts, where AWS services generate various types of logs, such as general, error, audit, slow query logs from Amazon RDS, control plane logs from Amazon EKS, and function execution and error logs from Lambda.
   - CloudWatch serves as the initial collection point. It gathers these logs at the log group level within each application account.
   - In CloudWatch, subscription filters determine which logs should be forwarded to the central account. These filters give you granular control over log forwarding, so you can specify exact log patterns or complete log streams for centralization.
2. **Cross-account log transfer**
   - Logs move to the Log Archive account. CloudWatch subscription filters facilitate the cross-account transfer and preserve Regional context.

- The architecture establishes multiple parallel streams to handle different log sources efficiently, to ensure optimal performance and scalability.

3. **Log processing in the Log Archive account**
   - In the Log Archive account, Firehose processes the incoming log streams.
   - Each Region maintains dedicated Firehose delivery streams that can transform, convert, or enrich logs as needed.
   - These Firehose streams deliver the processed logs to S3 buckets in the Log Archive account, which is located in the same Region as the source Application accounts (Region A in the diagram) to maintain data sovereignty requirements.
4. **Notifications and additional workflows**
   - When logs reach their destination S3 buckets, the architecture implements a notification system by using Amazon SQS.
   - The Regional SQS queues enable asynchronous processing and can trigger additional workflows, analytics, or alerting systems based on the stored logs.
5. **AWS KMS for security**

   The architecture incorporates AWS KMS for security. AWS KMS provides encryption keys for the S3 buckets. This ensures that all stored logs maintain encryption at rest while keeping the encryption Regional to satisfy data residency requirements.

Tools

AWS services

- Amazon CloudWatch is a monitoring and observability service that collects monitoring and operational data in the form of logs, metrics, and events. It provides a unified view of AWS resources, applications, and services that run on AWS and on-premises servers.
- CloudWatch Logs subscription filters are expressions that match a pattern in incoming log events and deliver matching log events to the specified AWS resource for further processing or analysis.
- AWS Control Tower Account Factory For Terraform (AFT) sets up a Terraform pipeline to help you provision and customize accounts in AWS Control Tower. AFT provides Terraform-based account provisioning while allowing you to govern your accounts with AWS Control Tower.
- Amazon Data Firehose delivers real-time streaming data to destinations such as Amazon S3, Amazon Redshift, and Amazon OpenSearch Service. It automatically scales to match the throughput of your data and requires no ongoing administration.

- Amazon Elastic Kubernetes Service (Amazon EKS) is a managed container orchestration service that makes it easy to deploy, manage, and scale containerized applications by using Kubernetes. It automatically manages the availability and scalability of the Kubernetes control plane nodes.
- AWS Key Management Service (AWS KMS) creates and controls encryption keys for encrypting your data. AWS KMS integrates with other AWS services to help you protect the data you store with these services.
- AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. It automatically scales your applications by running code in response to each trigger, and charges only for the compute time that you use.
- Amazon Relational Database Service (Amazon RDS) is a managed relational database service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks.
- Amazon Simple Queue Service (Amazon SQS) is a message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. It eliminates the complexity of managing and operating message-oriented middleware.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that offers scalability, data availability, security, and performance. It can store and retrieve any amount of data from anywhere on the web.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code

The code for this pattern are available in the GitHub Centralized logging repository.

Best practices

- Use multiple AWS accounts in a single organization in AWS Organizations. This practice enables centralized management and standardized logging across accounts.
- Configure S3 buckets with versioning, lifecycle policies, and cross-Region replication. Implement encryption and access logging for security and compliance.
- Implement common logging standards by using JSON format with standard timestamps and fields. Use a consistent prefix structure and correlation IDs for easy tracking and analysis.

- Enable security controls with AWS KMS encryption and least privilege access. Maintain AWS CloudTrail monitoring and regular key rotation for enhanced security.
- Set up CloudWatch metrics and alerts for delivery tracking. Monitor costs and performance with automated notifications.
- Configure Amazon S3 retention policies to meet compliance requirements and enable Amazon S3 server access logging to track all requests made to your S3 buckets. Maintain documentation for S3 bucket policies and lifecycle rules. Conduct periodic reviews of access logs, bucket permissions, and storage configurations to help ensure compliance and security best practices.

Epics

Set up AWS Control Tower, AFT, and Application accounts

Task	Description	Skills required
Set up an AWS Control Tower environment with AFT.	1. Deploy AWS Control Tower by following the instructi ons in the AWS Control Tower documentation. 2. Deploy AFT by following the instructions in the AWS Control Tower documenta tion.	AWS administrator
Enable resource sharing for the organization.	1. Configure AWS Command Line Interface (AWS CLI) with management account credentials, which provide administrative permissio ns to manage AWS Control Tower. 2. Run the following AWS CLI command in any AWS Region:	AWS administrator

Task	Description	Skills required
	```aws ram enable-sharing-with-aws-organization``` This enables resource sharing within your organization in AWS Organizations across all Regions that support AWS Resource Access Manager (AWS RAM).	
Verify or provision Application accounts.	To provision new Application accounts for your use case, create them through AFT. For more information, see [Provision a new account with AFT] in the AWS Control Tower documentation.	AWS administrator

Set up configuration files for Application accounts

Task	Description	Skills required
Copy `Application_account` folder contents into the `aft-account-customizations` repository.	1. Create a folder named `Application_account` in the root path of the `aft-account-customizations` repository. This repository is created automatically when you set up AFT (see the previous epic).	DevOps engineer

Task	Description	Skills required
	2. Navigate to the root directory of the [centralised-logging-at-enterprise-scale-using-terraform](https://github.com/aws-samples/centralised-logging-at-enterprise-scale-using-terraform) repository, copy the contents of the `aft/account` directory, and then paste them into the `Application_account` directory that you created in step 1 in the `aft-account-customizations` repository. 3. From the root directory of the `centralised-logging-at-enterprise-scale-using-terraform` repository, copy the contents of the `Application_account` directory into the `Application_account/terraform` directory in the `aft-account-customizations` repository. 4. In the `aft-account-customizations/Application_account/terraform.tfvars` file, confirm that all parameters are passed as arguments in the correspon	

Task	Description	Skills required
	ding Terraform configura tion files.	

Task	Description	Skills required
Review and edit the input parameters for setting up the Application account.	In this step, you set up the configuration file for creating resources in Application accounts, including CloudWatch log groups, CloudWatch subscription filters, IAM roles and policies, and configuration details for Amazon RDS, Amazon EKS, and Lambda functions. In your `aft-account-customizations` repository, in the `Application_account` folder, configure the input parameters in the `terraform.tfvars` file based on your organization's requirements: • `environment` : The name of the environment (for example, `prod`, `dev`, `staging`) where the resources will be deployed. • `account_name` : The name of the AWS account where the resources will be created. • `log_archive_account_id` : The AWS account ID where logs will be archived. • `admin_role_name` : The name of the administrative	DevOps engineer

Task	Description	Skills required
	role that will be used for managing resources. • `tags`: A map of key-value pairs that represent common tags to be applied to all resources. • `rds_config` : An object that contains configuration details for Amazon RDS instances. • `allowed_cidr_blocks` : A list of CIDR blocks that are allowed to access the resources. • `destination_name`: A variable that is used to create the Amazon Resource Name (ARN) of the CloudWatch destination where logs will be streamed. • `rds_parameters`: An object that contains Amazon RDS parameter group settings. • `vpc_config` : An object that contains VPC configuration details. • `eks_config` : An object that contains configuration details for Amazon EKS clusters.	

Task	Description	Skills required
	• `lambda_config` : An object that contains configuration details for Lambda functions. • `restrictive_cidr_range` : A list of restrictive CIDR ranges for security group rules. • `target_account_id` : The AWS account ID of the target Log Archive account where resources will be deployed.	

Set up configuration files for the Log Archive account

Task	Description	Skills required
Copy `Log_archive_account` folder contents into the `aft-account-customizations` repository.	1. Create a folder named `Log_archive_account` in the root path of the `aft-account-customizations` repository, This repository is created automatically when you set up AFT. 2. Navigate to the root directory of the `centralised-logging-at-enterprise-scale-using-terraform` repository, copy the contents of the `aft/`	DevOps engineer

Task	Description	Skills required
	`account` directory, and then paste them into the `Log_archive_account` directory that you created in the previous step in the `aft-account-customizations` repository. 3. From the root directory of the `centralised-logging-at-enterprise-scale-using-terraform` repository, copy the contents of the `Log_archive_account` directory into the `Log_archive_account/terraform` directory in the `aft-account-customizations` repository. 4. In the `aft-account-customizations/Log_archive_account/terraform.tfvars` file, confirm that all parameters are passed as arguments in the corresponding Terraform configuration files.	

Task	Description	Skills required
Review and edit the input parameters for setting up the Log Archive account.	In this step, you set up the configuration file for creating resources in the Log Archive account, including Firehose delivery streams, S3 buckets, SQS queues, and IAM roles and policies. In the `Log_archive_account` folder of your `aft-account-customizations` repository, configure the input parameters in the `terraform.tfvars` file based on your organization's requirements: • `environment` : The name of the environment (for example, `prod`, `dev`, `staging`) where the resources will be deployed. • `destination_name` : A variable that is used to create the ARN of the CloudWatch destination where logs will be streamed. • `source_account_ids` : A list of AWS account IDs that are allowed to put subscription filters on the log destination. You can input as many account IDs	DevOps engineer

Task	Description	Skills required
	as you want to enable for centralized logging.	

Run Terraform commands to provision resources

Task	Description	Skills required

Option 1 - Deploy the Terraform configuration files from AFT.	In AFT, the AFT pipeline is triggered after you push the code with the configuration changes to the GitHub `aft-account-customizations` repository. AFT automatically detects the changes and initiates the account customization process.
	
	After you make changes to your Terraform (`terraform.tfvars`) files, commit and push your changes to your `aft-account-customizations` repository:
	
	`$ git add *`
	`$ git commit -m "update message"`
	`$ git push origin main`
	
	Note
	If you're using a different branch (such as `dev`), replace `main`
		DevOps engineer

Task	Description	Skills required
	with your branch name.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Option 2 - Deploy the Terraform configuration file manually.</td><td>If you aren't using AFT or you want to deploy the solution manually, you can use the following Terraform commands from the <code>Application_account</code> and <code>Log_archive_account</code> folders:

1. Clone the GitHub repository and configure the input parameters in the <code>terraform.tfvars</code> file.
2. Run the following command:
<pre>$ terraform init</pre>3. Preview changes:
<pre>$ terraform plan</pre>This command evaluates the Terraform configuration to determine the desired state of the resources and compares it with the current state of your infrastructure.
4. Apply changes:
<pre>$ terraform apply</pre>5. Review the planned changes and type yes at</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	the prompt to proceed with the application.	

Validate resources

Task	Description	Skills required
Verify subscription filters.	To verify that the subscription filters forward logs correctly from the Application account log groups to the Log Archive account: 1. In the Application account, open the [CloudWatch console](https://console.aws.amazon.com/cloudwatch/). 2. In the left navigation pane, choose **Log groups**. 3. Select each log group (`/aws/rds`, `/aws/eks`, `/aws/lambda`) and choose the **Subscription filters** tab. You should see active subscription filters that point to the destination ARN, based on the name that you specified in the Terraform configuration file. 4. Choose any subscription filter to verify its configura tion and status.	DevOps engineer

Task	Description	Skills required

Verify Firehose streams.	To verify that the Firehose streams in the Log Archive account process application logs successfully:
	
	1. In the Log Archive account, open the [Firehose console](https://console.aws.amazon.com/firehose/).
	2. In the left navigation pane, choose **Firehose streams**.
	3. Choose any **Firehose streams** and verify the following:
	  • The destination shows the correct S3 bucket.
	  • The **Monitoring** tab shows successful delivery metrics.
	  • The recent delivery timestamp is current.
		DevOps engineer

Task	Description	Skills required
Validate the centralized S3 buckets.	To verify that the centralized S3 buckets receive and organize logs properly: 1. In the Log Archive account, open the [Amazon S3 console](https://console.aws.amazon.com/s3/). 2. Select each central logging bucket. 3. Navigate through the folder structure: **AWSLogs/AccountID/Region/Service**. You should see log files organized by timestamp (**YYYY/MM/DD/HH**). 4. Choose any recent log file and verify its format and data integrity.	DevOps engineer

Task	Description	Skills required
Validate SQS queues.	To verify that the SQS queues receive notifications for new log files: 1. In the Log Archive account, open the [Amazon SQS console](https://console.aws.amazon.com/sqs/). 2. In the left navigation pane, choose **Queues**. 3. Select each configured queue and choose **Send and receive messages**. You should see messages that contain **S3 event notifications** for new log files. 4. Choose any message to verify that it contains the correct S3 object informati on.	DevOps engineer

Clean up resources

Task	Description	Skills required
Option 1 - Decommission the Terraform configuration file from AFT.	When you remove the Terraform configuration files and push the changes, AFT automatically initiates the resource removal process.	DevOps engineer

Task	Description	Skills required

	1. Navigate to the `aft-account-customizations` repository.
	2. Go to the `terraform` directory.
	3. Delete the following files:
	• `modules` directory
	• `iam.tf`
	• `versions.tf`
	• `variables.tf`
	• `outputs.tf`
	• `terraform.tfvars`
	4. Clear the contents of the `main.tf` file.
	5. Push your changes to the repository:
	`# Stage all changes`
	`$ git add *`
	
	`# Commit cleanup changes`
	`$ git commit -m "Remove AFT customizations"`
	
	`# Push to repository`
	`$ git push origin main`
	
	Note
	If you're using a different branch (such as dev),	

Task	Description	Skills required
	replace `main` with your branch name.	

Option 2 – Clean up Terraform resources manually.	If you aren't using AFT or you want to clean up resources manually, use the following Terraform commands from the `Application_account` and `Log_archive_account` folders:
	
	1. Initialize the Terraform configuration:
	`$ terraform init`
	This command initializes Terraform and ensures access to the current state.
	2. Preview cleanup changes:
	`$ terraform destroy`
	This command evaluates which resources will be destroyed and compares the desired state with the current state of your infrastructure.
	3. Run cleanup. When you're prompted, type **yes** to confirm and run the destruction plan.
		DevOps engineer

Troubleshooting

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td>The CloudWatch Logs destination wasn't created or is inactive.</td><td>Validate the following:

In the Log Archive account, verify that the destination policy includes:

The correct source account principal.
The correct action (<code>logs:PutSubscriptionFilter</code>).
A valid destination ARN.

Confirm that the Firehose stream exists and is active.
Verify that the IAM role that's attached to the destination has permissions for Firehose.
</td></tr>
<tr><td>The subscription filter failed or is stuck in pending status.</td><td>Check the following:

In the Application account, verify that the IAM role has:

Permissions to call <code>PutSubscriptionFilter</code>.
A trust relationship with CloudWatch Logs.

Confirm that the destination ARN is correct.
Check CloudWatch Logs for specific error messages.
</td></tr>
<tr><td>The Firehose delivery stream shows no incoming records.</td><td>Verify the following:

Confirm that the Firehose IAM role has:

Permissions to write to Amazon S3.
Access to the AWS KMS key if encryption is enabled.

</td></tr>
</table>

Issue	Solution
	2. Review CloudWatch metrics for: • `IncomingRecords` • `DeliveryToS3.Records` 3. Verify buffer settings and delivery configura tions.

Related resources

- Terraform infrastructure setup (Terraform documentation)
- Deploy AWS Control Tower Account Factory for Terraform (AFT) (AWS Control Tower documentation)
- IAM tutorial: Delegate access across AWS accounts using IAM roles (IAMdocumentation)

Set up end-to-end encryption for applications on Amazon EKS using cert-manager and Let's Encrypt

Mahendra Revanasiddappa and Vasanth Jeyaraj, Amazon Web Services

Summary

Implementing end-to-end encryption can be complex and you need to manage certificates for each asset in your microservices architecture. Although you can terminate the Transport Layer Security (TLS) connection at the edge of the Amazon Web Services (AWS) network with a Network Load Balancer or Amazon API Gateway, some organizations require end-to-end encryption.

This pattern uses NGINX Ingress Controller for ingress. This is because when you create a Kubernetes ingress, the ingress resource uses a Network Load Balancer. The Network Load Balancer doesn't permit uploads of client certificates. Therefore, you can't achieve mutual TLS with Kubernetes ingress.

This pattern is intended for organizations that require mutual authentication between all microservices in their applications. Mutual TLS reduces the burden of maintaining user names or passwords and can also use the turnkey security framework. This pattern’s approach is compatible if your organization has a large number of connected devices or must comply with strict security guidelines.

This pattern helps increase your organization's security posture by implementing end-to-end encryption for applications running on Amazon Elastic Kubernetes Service (Amazon EKS). This pattern provides a sample application and code in the GitHub End-to-end encryption on Amazon EKS repository to show how a microservice runs with end-to-end encryption on Amazon EKS. The pattern's approach uses cert-manager, an add-on to Kubernetes, with Let's Encrypt as the certificate authority (CA). Let's Encrypt is a cost-effective solution to manage certificates and provides free certificates that are valid for 90 days. Cert-manager automates the on-demand provisioning and rotating of certificates when a new microservice is deployed on Amazon EKS.

Intended audience

This pattern is recommended for users who have experience with Kubernetes, TLS, Amazon Route 53, and Domain Name System (DNS).

Prerequisites and limitations

Prerequisites

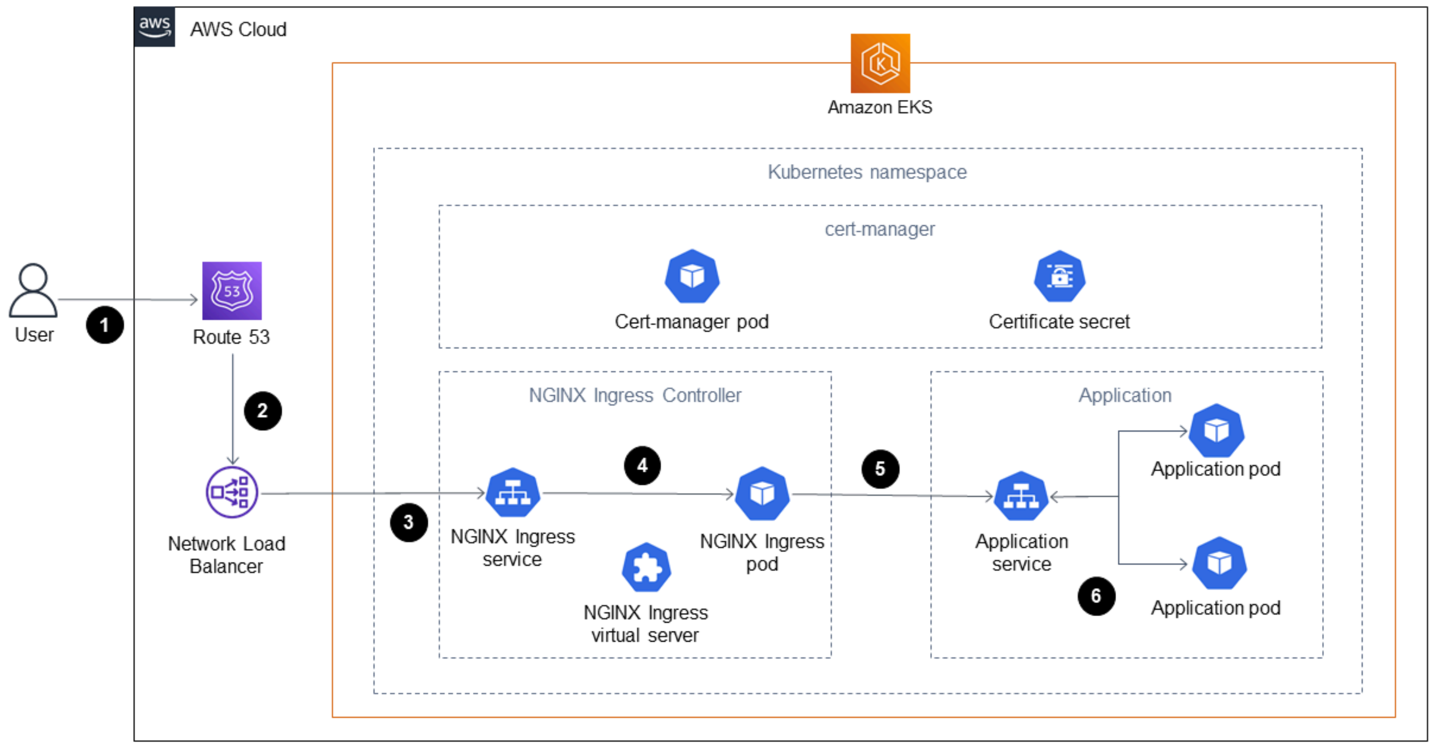
- An active AWS account.
- An existing Amazon EKS cluster.
- AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows.
- The `kubectl` command line utility, installed and configured to access the Amazon EKS cluster. For more information about this, see Installing kubectl in the Amazon EKS documentation.
- An existing DNS name to test the application. For more information about this, see Registering domain names using Amazon Route 53 in the Amazon Route 53 documentation.
- The latest Helm version, installed on your local machine. For more information about this, see Using Helm with Amazon EKS in the Amazon EKS documentation and the GitHub Helm repository.
- The GitHub End-to-end encryption on Amazon EKS repository, cloned to your local machine.
- Replace the following values in the `policy.json` and `trustpolicy.json` files from the cloned GitHub End-to-end encryption on Amazon EKS repository:
  - `<account number>` – Replace with the AWS account ID for the account that you want to deploy the solution in.
  - `<zone id>` – Replace with the domain name's Route 53 zone ID.
  - `<node_group_role>` – Replace with the name of the AWS Identity and Access Management (IAM) role associated with the Amazon EKS nodes.
  - `<namespace>` – Replace with the Kubernetes namespace in which you deploy the NGINX Ingress Controller and the sample application.
  - `<application-domain-name>` – Replace with the DNS domain name from Route 53.

Limitations

- This pattern doesn't describe how to rotate certificates and only demonstrates how to use certificates with microservices on Amazon EKS.

Architecture

The following diagram shows the workflow and architecture components for this pattern.



The diagram shows the following workflow:

1. A client sends a request to access the application to the DNS name.
2. The Route 53 record is a CNAME to the Network Load Balancer.
3. The Network Load Balancer forwards the request to the NGINX Ingress Controller that is configured with a TLS listener. Communication between the NGINX Ingress Controller and the Network Load Balancer follows HTTPS protocol.
4. The NGINX Ingress Controller carries out path-based routing based on the client's request to the application service.
5. The application service forwards the request to the application pod. The application is designed to use the same certificate by calling secrets.
6. Pods run the sample application using the cert-manager certificates. The communication between the NGINX Ingress Controller and the pods uses HTTPS.

Note: Cert-manager runs in its own namespace. It uses a Kubernetes cluster role to provision certificates as secrets in specific namespaces. You can attach those namespaces to application pods and NGINX Ingress Controller.

Tools

AWS services

- Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.
- Elastic Load Balancing automatically distributes your incoming traffic across multiple targets, containers, and IP addresses.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Route 53 is a highly available and scalable DNS web service.

Other tools

- cert-manager is an add-on to Kubernetes that requests certificates, distributes them to Kubernetes containers, and automates certificate renewal.
- NGINX Ingress Controller is a traffic management solution for cloud-native apps in Kubernetes and containerized environments.

Epics

Create and configure a public hosted zone with Route 53

Task	Description	Skills required
Create a public hosted zone in Route 53.	Sign in to the AWS Management Console, open the Amazon Route 53 console, choose **Hosted zones**, and then choose **Create hosted zone**. Create a public hosted zone and record the zone ID. For more information about this, see Creating a public hosted	AWS DevOps

Task	Description	Skills required
	zone in the Amazon Route 53 documentation. **Note** ACME DNS01 uses the DNS provider to post a challenge for cert-manager to issue the certificate. This challenge asks you to prove that you control the DNS for your domain name by putting a specific value in a TXT record under that domain name. After Let's Encrypt gives your ACME client a token, your client creates a TXT record derived from that token and your account key, and it puts that record at `_acme-challenge.<YOURDOMAIN>`. Then Let's Encrypt queries the DNS for that record. If it finds a match, you can proceed to issue a certificate.	

Configure an IAM role to allow cert-manager to access the public hosted zone

Task	Description	Skills required
Create the IAM policy for cert-manager.	An IAM policy is required to provide cert-manager with permission to validate that you own the Route 53 domain. The `policy.json` sample IAM policy is provided in the `1-IAMRole` directory in the cloned GitHub [End-to-end encryption on Amazon EKS](#) repository. Enter the following command in AWS CLI to create the IAM policy. `aws iam create-policy \ --policy-name PolicyForCertManager \ --policy-document file://policy.json`	AWS DevOps
Create the IAM role for cert-manager.	After you create the IAM policy, you must create an IAM role. The `trustpolicy.json` sample IAM role is provided in the `1-IAMRole` directory. Enter the following command in AWS CLI to create the IAM role. `aws iam create-role \`	AWS DevOps

Task	Description	Skills required

	``
	  --role-name RoleForCe
	rtManager \
	  --assume-role-poli
	cy-document file://tr
	ustpolicy.json
	``	

Attach the policy to the role.	Enter the following command in AWS CLI to attach the IAM policy to the IAM role. Replace `AWS_ACCOUNT_ID` with the ID of your AWS account.
	
	``
	aws iam attach-role-
	policy \
	  --policy-arn
	 arn:aws:iam::AWS_A
	CCOUNT_ID:policy/P
	olicyForCertManager \
	  --role-name RoleForCe
	rtManager
	``
		AWS DevOps

Set up the NGINX Ingress Controller in Amazon EKS

Task	Description	Skills required
Deploy the NGINX Ingress Controller.	Install the most recent version of `nginx-ingress` using Helm. You can modify the `nginx-ingress` configuration according to your requirements before deploying it. This pattern uses an annotated, internal-facing Network Load Balancer and that is available in the	AWS DevOps

Task	Description	Skills required
	`5-Nginx-Ingress-Controller` directory. Install the NGINX Ingress Controller by running the following Helm command from the `5-Nginx-Ingress-Controller` directory. `helm install test-nginx nginx-stable/nginx-ingress -f 5-Nginx-Ingress-Controller/values_internal_nlb.yaml`	
Verify that the NGINX Ingress Controller is installed.	Enter the `helm list` command. The output should show that the NGINX Ingress Controller is installed.	AWS DevOps

Task	Description	Skills required

Create a Route 53 A record.	The A record points to the Network Load Balancer created by NGINX Ingress Controller.
	
	1. Get the DNS name of the Network Load Balancer. For instructions, see Getting the DNS name for an ELB load balancer.
	2. On the Amazon Route 53 console, choose **Hosted Zones**.
	3. Select the public hosted zone that you want to create the record in, and then choose **Create record**.
	4. Enter a name for the record.
	5. In **Record type**, choose **A - Routes traffic to IPv4 and some AWS resources**.
	6. Enable **Alias**.
	7. In **Route traffic to**, do the following:
	  a. Choose **Alias to Network Load Balancer**.
	  b. Choose the AWS Region in which the Network Load Balancer is deployed.
		AWS DevOps

Task	Description	Skills required
	c. Enter the DNS name of the Network Load Balancer. 8. Choose **Create records**.	

Set up NGINX VirtualServer on Amazon EKS

Task	Description	Skills required
Deploy NGINX VirtualServer.	The NGINX VirtualServer resource is a load balancing configuration that is an alternative to the ingress resource. The configuration to create the NGINX VirtualServer resource is available in the `nginx_virtualserver.yaml` file in the `6-Nginx-Virtual-Server` directory. Enter the following command in `kubectl` to create the NGINX VirtualServer resource. `kubectl apply -f nginx_virtualserver.yaml` **Important** Make sure that you update the application domain name, certificate secret,	AWS DevOps

Task	Description	Skills required
	and application service name in the `nginx_virtualserver.yaml` file.	
Verify that NGINX VirtualServer is created.	Enter the following command in `kubectl` to verify that the NGINX VirtualServer resource was successfully created. `kubectl get virtualserver` **Note** Verify that the `Host` column matches your application's domain name.	AWS DevOps

Task	Description	Skills required
Deploy the NGINX web server with TLS enabled.	This pattern uses a NGINX web server with TLS enabled as the application for testing end-to-end encryption. The configuration files required to deploy the test applicati on are available in the `demo-webserver` directory. Enter the following command in `kubectl` to deploy the test application. `kubectl apply -f nginx-tls-ap.yaml`	AWS DevOps

Task	Description	Skills required
Verify that the test application resources are created.	Enter the following commands in `kubectl` to verify that the required resources are created for the test application: • `kubectl get deployments` **Note:** Validate the `Ready` column and `Available` column. • `kubectl get pods | grep -i example-deploy` **Note:** Pods should be in `running` state. • `kubectl get configmap` • `kubectl get svc`	AWS DevOps

Task	Description	Skills required
Validate the application.	1. Enter the following command by replacing the `<application-domain-name>` with the Route53 DNS name that you created earlier. `curl --verbose https://<application-domain-name>` 2. Verify that you can access the application.	AWS DevOps

Related resources

AWS resources

- Creating records by using the Amazon Route 53 console (Amazon Route 53 documentation)
- Using a Network Load Balancer with the NGINX ingress controller on Amazon EKS (AWS blog post)

Other resources

- Route 53 (cert-manager documentation)
- Configuring DNS01 Challenge Provider (cert-manager documentation)
- Let's encrypt DNS challenge (Let's Encrypt documentation)

Simplify Amazon EKS multi-tenant application deployment by using Flux

Nadeem Rahaman, Aditya Ambati, Aniket Dekate, and Shrikant Patil, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

Many companies that offer products and services are data-regulated industries that are required to maintain data barriers between their internal business functions. This pattern describes how you can use the multi-tenancy feature in Amazon Elastic Kubernetes Service (Amazon EKS) to build a data platform that achieves logical and physical isolation between tenants or users that share a single Amazon EKS cluster. The pattern provides isolation through the following approaches:

- Kubernetes namespace isolation
- Role-based access control (RBAC)
- Network policies
- Resource quotas
- AWS Identity and Access Management (IAM) roles for service accounts (IRSA)

In addition, this solution uses Flux to keep the tenant configuration immutable when you deploy applications. You can deploy your tenant applications by specifying the tenant repository that contains the Flux `kustomization.yaml` file in your configuration.

This pattern implements the following:

- An AWS CodeCommit repository, AWS CodeBuild projects, and an AWS CodePipeline pipeline, which are created by manually deploying Terraform scripts.
- Network and compute components required for hosting the tenants. These are created through CodePipeline and CodeBuild by using Terraform.
- Tenant namespaces, network policies, and resource quotas, which are configured through a Helm chart.
- Applications that belong to different tenants, deployed by using Flux.

We recommend that you carefully plan and build your own architecture for multi-tenancy based on your unique requirements and security considerations. This pattern provides a starting point for your implementation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2.11.4 or later, installed and configured
- Terraform version 0.12 or later installed on your local machine
- Terraform AWS Provider version 3.0.0 or later
- Kubernetes Provider version 2.10 or later
- Helm Provider version 2.8.0 or later
- Kubectl Provider version 1.14 or later

Limitations

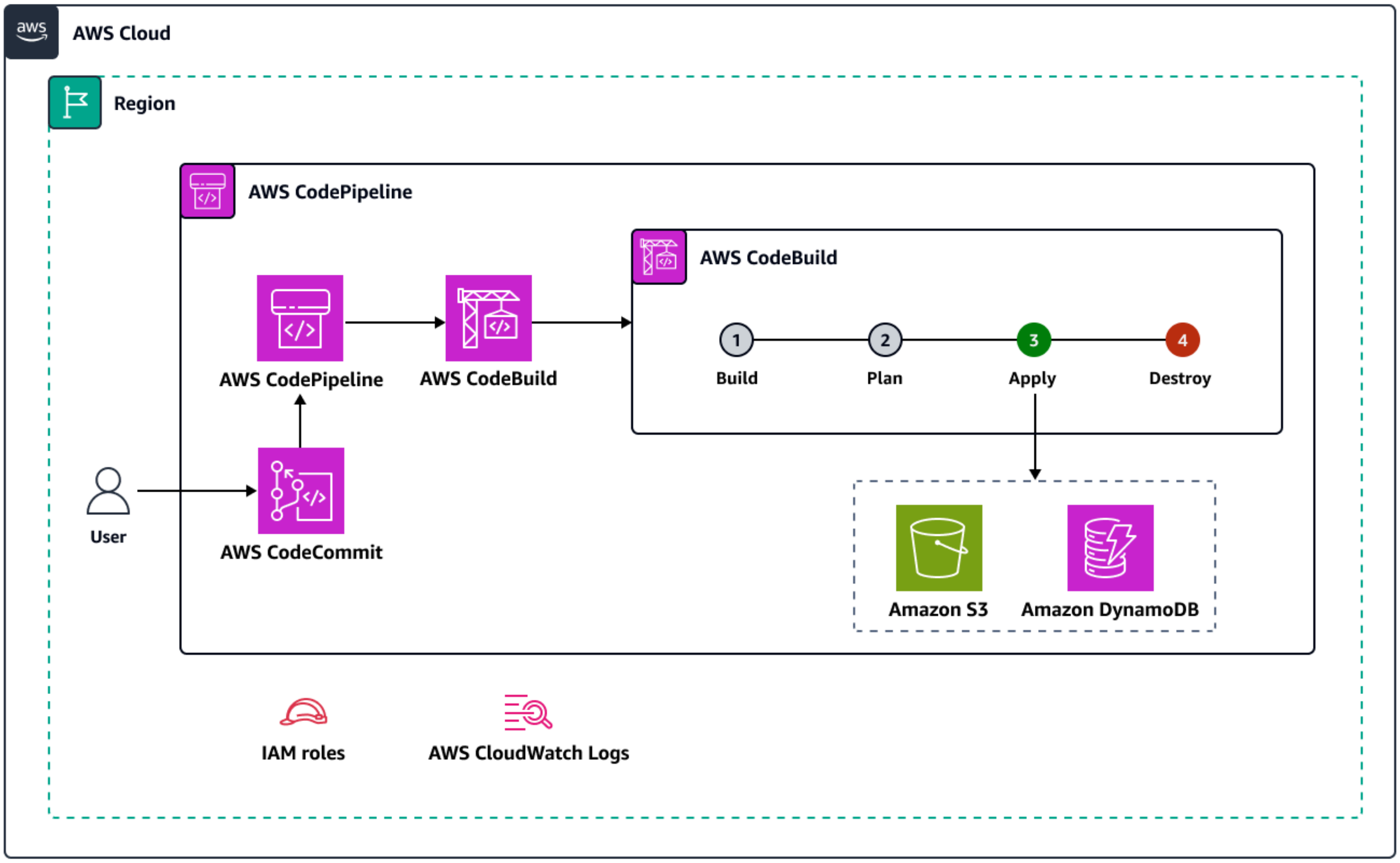
- **Dependency on Terraform manual deployments:** The workflow's initial setup, including creating CodeCommit repositories, CodeBuild projects, and CodePipeline pipelines, relies on manual Terraform deployments. This introduces a potential limitation in terms of automation and scalability, because it requires manual intervention for infrastructure changes.
- **CodeCommit repository dependency:** The workflow relies on CodeCommit repositories as the source code management solution and is tightly coupled with AWS services.

Architecture

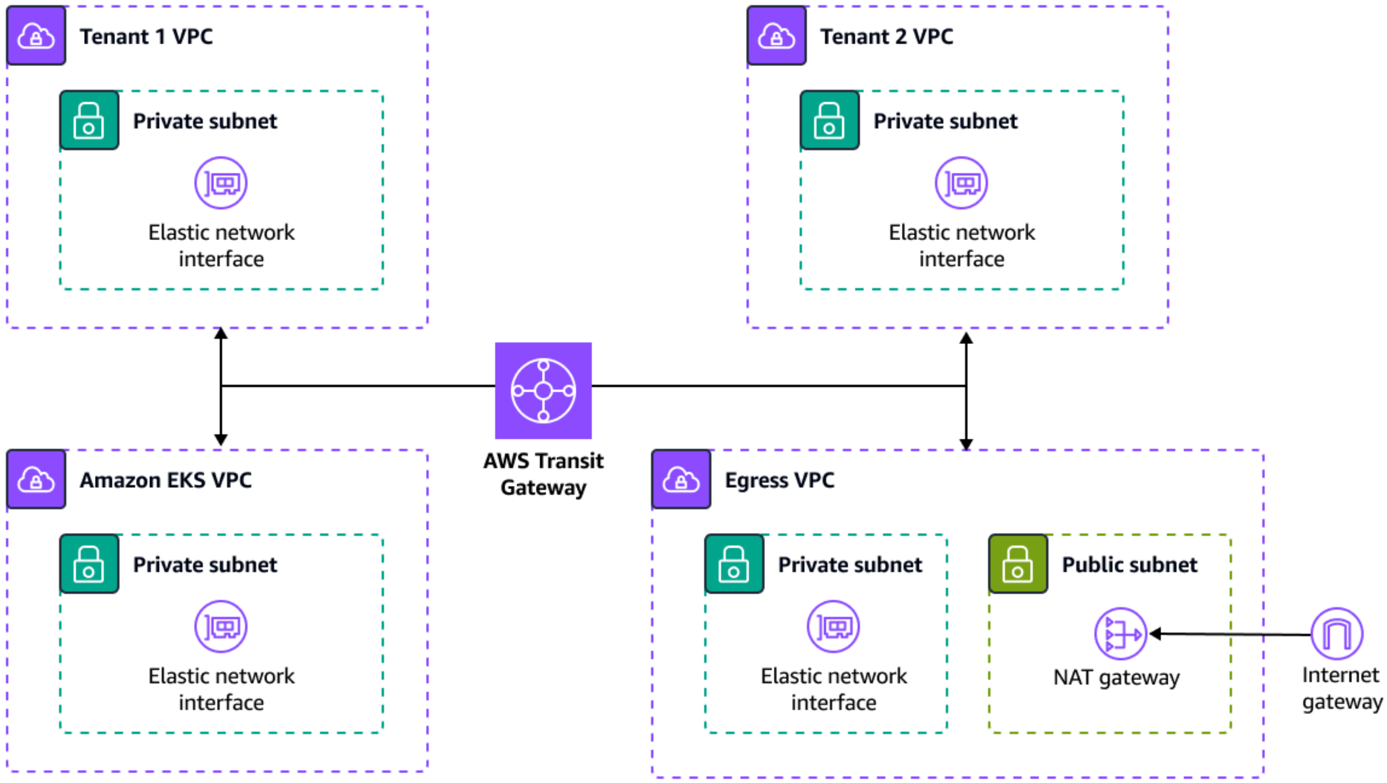
Target architectures

This pattern deploys three modules to build the pipeline, network, and compute infrastructure for a data platform, as illustrated in the following diagrams.

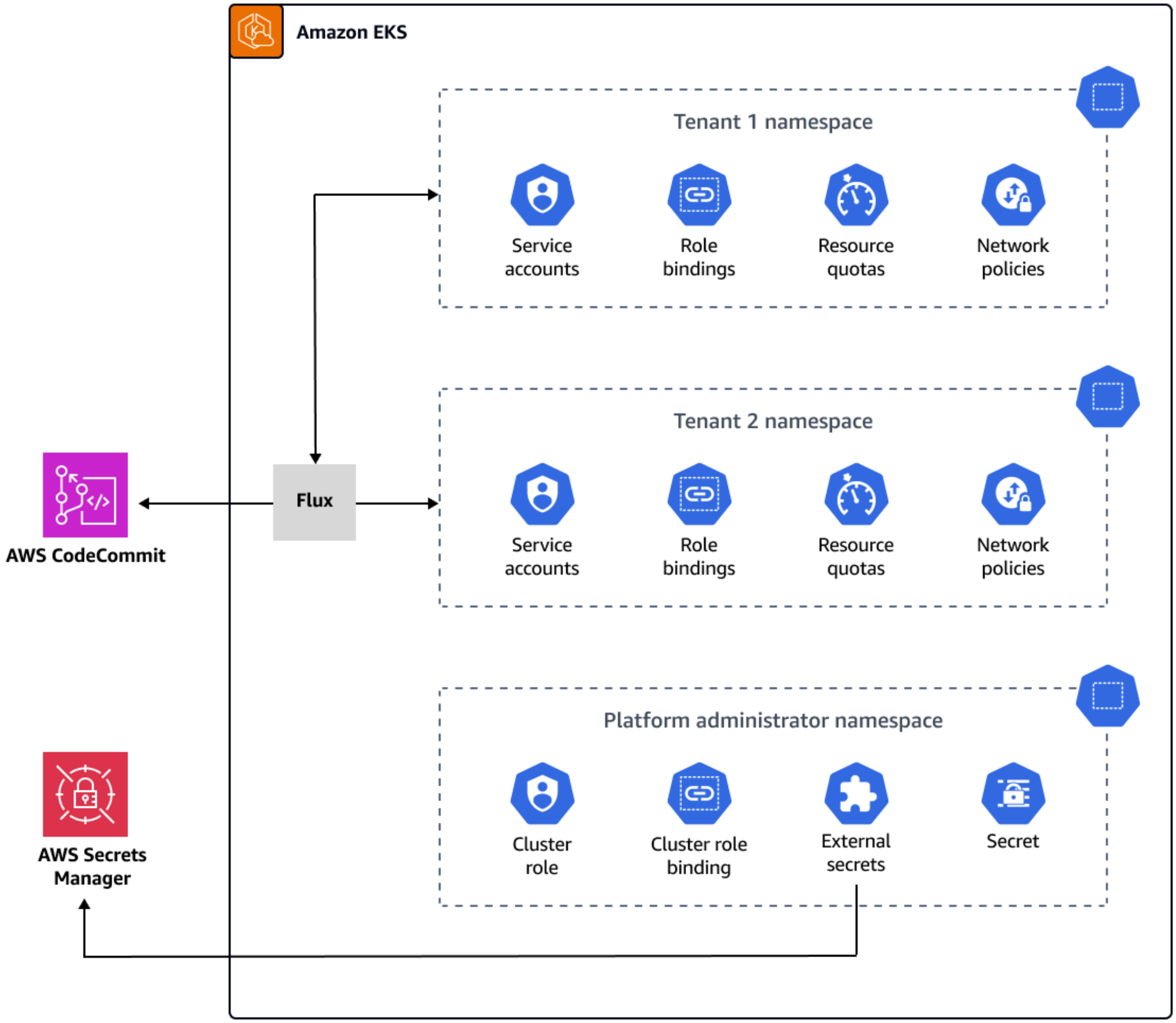
Pipeline architecture:



Network architecture:



Compute architecture:



Tools

AWS services

- [AWS CodeBuild](#) is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- [AWS CodeCommit](#) is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.

- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Transit Gateway is a central hub that connects virtual private clouds (VPCs) and on-premises networks.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- Cilium Network Policies support Kubernetes L3 and L4 networking policies. They can be extended with L7 policies to provide API-level security for HTTP, Kafka, and gRPC, and other similar protocols.
- Flux is a Git-based continuous delivery (CD) tool that automates application deployments on Kubernetes.
- Helm is an open source package manager for Kubernetes that helps you install and manage applications on your Kubernetes cluster.
- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub EKS Multi-Tenancy Terraform Solution repository.

Best practices

For guidelines and best practices for using this implementation, see the following:

- Amazon EKS multi-tenancy best practices
- Flux documentation

Epics

Create pipelines for Terraform build, test, and deploy stages

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Clone the project repository.</td><td>Clone the GitHub EKS Multi-Tenancy Terraform Solution repository by running the following command in a terminal window:
<pre><code>git clone https://github.com/aws-samples/aws-eks-multitenancy-deployment.git</code></pre></td><td>AWS DevOps</td></tr>
<tr><td>Bootstrap the Terraform S3 bucket and Amazon DynamoDB.</td><td>1. In the <code>bootstrap</code> folder, open the <code>bootstrap.sh</code> file and update the variable values for the S3 bucket name, DynamoDB table name, and AWS Region:
<pre><code>S3_BUCKET_NAME="<S3_BUCKET_NAME>"
DYNAMODB_TABLE_NAME="<DYNAMODB_NAME>"
REGION="<AWS_REGION>"</code></pre>2. Run the <code>bootstrap.sh</code> script. The script requires the AWS CLI, which you installed as part of prerequisites.
<pre><code>cd bootstrap</code></pre></td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	`./bootstrap.sh`	

Task	Description	Skills required

Update the `run.sh` and `locals.tf` files.	1. After the bootstrap process completes successfully, copy the S3 bucket and DynamoDB table name from the `variables` section of the `bootstrap.sh` script:
	`# Variables`
	`S3_BUCKET_NAME="<S3_BUCKET_NAME>"`
	`DYNAMODB_TABLE_NAME="<DYNAMODB_NAME"`
	2. Paste those values to the `run.sh` script, which is in the root directory of the project:
	`BACKEND_BUCKET_ID="<SAME_NAME_AS_S3_BUCKET_NAME>"`
	`DYNAMODB_ID="<SAME_NAME_AS_DYNAMODB_NAME>"`
	3. Upload the project code to a CodeCommit repository. You can automatically create this repository through Terraform by setting the following variable to `true` in the `demo/pipeline/locals.tf` file:
		AWS DevOps

Task	Description	Skills required
	`create_new_repo = true` 4. Update the `locals.tf` file according to your requirements to create pipeline resources.	

Deploy the pipeline module.	To create pipeline resources, run the following Terraform commands manually. There is no orchestration for running these commands automatically.
	`./run.sh -m pipeline -e demo -r <AWS_REGION> -t init`
	`./run.sh -m pipeline -e demo -r <AWS_REGION> -t plan`
	`./run.sh -m pipeline -e demo -r <AWS_REGION> -t apply`
		AWS DevOps

Create the network infrastructure

Task	Description	Skills required
Start the pipeline.	1. In the `templates` folder, make sure that the `buildspec` files have the following variable set to `network`:	AWS DevOps

Task	Description	Skills required
	`TF_MODULE_TO_BUILD: "network"` 2. On the CodePipeline console, on the pipeline details page, start the pipeline by choosing **Release change**. After this first run, the pipeline starts automatically whenever you commit a change to the CodeCommit repository main branch. The pipeline includes the following stages: • `validate` initializes Terraform, runs Terraform security scans by using the checkov and tfsec tools, and uploads the scan reports to the S3 bucket. • `plan` shows the Terraform plan and uploads the plan to the S3 bucket. • `apply` applies the Terraform plan output from the S3 bucket and creates AWS resources. • `destroy` removes the AWS resources created during the `apply` stage. To enable	

Task	Description	Skills required
	this optional stage, set the following variable to `true` in the `demo/pipeline/locals.tf` file: `enable_destroy_stage = true`	

Task	Description	Skills required
Validate the resources created through the network module.	Confirm that the following AWS resources were created after the pipeline deployed successfully: • An egress VPC with three public and three private subnets, internet gateway, and NAT gateway. • An Amazon EKS VPC with three private subnets. • Tenant 1 and Tenant 2 VPCs with three private subnets each. • A transit gateway with all VPC attachments and routes to each private subnet. • A static transit gateway route for the Amazon EKS egress VPC with a destination CIDR block of `0.0.0.0/0` . This is required to enable all VPCs to have outbound internet access through the Amazon EKS egress VPC.	AWS DevOps

Create the compute infrastructure

Task	Description	Skills required

Update `locals.tf` to enable the CodeBuild project's access to the VPC.	To deploy the add-ons for the Amazon EKS private cluster, the CodeBuild project must be attached to the Amazon EKS VPC.
	
	1. In the `demo/pipeline` folder, open the `locals.tf` file, and set the `vpc_enabled` variable to `true`.
	2. Run the `run.sh` script to apply the changes to the pipeline module:
	
	`demo/pipeline/locals.tf`
	`./run.sh -m pipeline -env demo -region <AWS_REGION> -tfcmd init`
	`./run.sh -m pipeline -env demo -region <AWS_REGION> -tfcmd plan`
	`./run.sh -m pipeline -env demo -region <AWS_REGION> -tfcmd apply`
		AWS DevOps

Update the `buildspec` files to build the compute module.	In the `templates` folder, in all `buildspec` YAML files, set the value of the `TF_MODULE_TO_BUILD`	AWS DevOps

Task	Description	Skills required
	variable from `network` to `compute`: `TF_MODULE_TO_BUILD: "compute"`	

Task	Description	Skills required

Update the `values` file for the tenant management Helm chart.	1. Open the `values.yaml` file in the following location:
	`cd cfg-terraform/demo/compute/cfg-tenant-mgmt`
	The file looks like this:
	`---`
	`global:`
	`  clusterRoles:`
	`    operator: platform-tenant`
	`    flux: flux-tenant-applier`
	`  flux:`
	`    tenantCloneBaseUrl: ${TEANT_BASE_URL}`
	`    repoSecret: ${TENANT_REPO_SECRET}`
	`tenants:`
	`  tenant-1:`
	`    quotas:`
	`      limits:`
	`        cpu: 1`
	`        memory: 1Gi`
	`    flux:`
	`      path: overlays/tenant-1`
	`  tenant-2:`
	`    quotas:`
	`      limits:`
	`        cpu: 1`
	`        memory: 2Gi`
	`    flux:`
		AWS DevOps

Task	Description	Skills required
	`path: overlays/tenant-2` 2. In the `global` and `tenants` sections, update the configuration based on your requirements: • `tenantCloneBaseUrl` – Path to the repository that hosts the code for all tenants (we use the same Git repository for all tenants) • `repoSecret` – Kubernetes secret that holds the SSH keys and known hosts to authenticate to the global tenant Git repository • `quotas` – Kubernetes resources quotas that you want to apply for each tenant • `flux path` – Path to the tenant application YAML files in the global tenant repository	

Task	Description	Skills required
Validate compute resources.	After you update the files in the previous steps, CodePipeline starts automatically. Confirm that it created the following AWS resources for the compute infrastructure: • Amazon EKS cluster with private endpoint • Amazon EKS worker nodes • Amazon EKS add-ons: external secrets, `aws-loadbalancer-controller` , and `metrics-server` • GitOps module, Flux Helm chart, Cilium Helm chart, and tenant management Helm chart	AWS DevOps

Check tenant management and other resources

Task	Description	Skills required
Validate the tenant management resources in Kubernetes.	Run the following commands to check that tenant management resources were created successfully with the help of Helm. 1. Tenant namespaces were created, as specified in `values.yaml` :	AWS DevOps

Task	Description	Skills required

	`kubectl get ns -A`
	2. Quotas are assigned to each tenant namespace, as specified in `values.yaml` :
	`kubectl get quota --namespace=<tenant_namespace>`
	3. Details of quotas are correct for each tenant namespace:
	`kubectl describe quota cpu-memory-resource-quota-limit -n <tenant_namespace>`
	4. Cilium Network Policies were applied to each tenant namespace:
	`kubectl get CiliumNetworkPolicy -A`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Verify tenant application deployments.</td><td>Run the following commands to verify that the tenant applications were deployed.

1. Flux is able to connect to the CodeCommit repository that's specified in the GitOps module:

<code>kubectl get gitrepositories -A</code>

2. The Flux kustomization controller has deployed the YAML files in the CodeCommit repository:

<code>kubectl get kustomizations -A</code>

3. All application resources are deployed in their tenant namespaces:

<code>kubectl get all -n <tenant_namespace></code>

4. An ingress has been created for each tenant:

<code>kubectl get ingress -n <tenant_namespace></code></td><td></td></tr>
</table>

Troubleshooting

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td>

You encounter an error message that's similar to the following:

```
Failed to checkout and determine
revision: unable to clone unknown
error: You have successfully
authenticated over SSH. You
can use Git to interact with AWS
CodeCommit.
```

</td><td>

Follow these steps to troubleshoot the issue:

1. Verify the tenant application repository: An empty or misconfigured repository might be causing the error. Make sure that the tenant application repository contains the required code.
2. Redeploy the `tenant_mgmt` module: In the `tenant_mgmt` module configuration file, locate the `app` block, and then set the `deploy` parameter to `0`:

   ```
   deploy = 0
   ```

   After you run the Terraform `apply` command, change the `deploy` parameter value back to `1`:

   ```
   deploy = 1
   ```

3. Recheck the status: After you run the previous steps, use the following command to check whether the issue persists:

   ```
   kubectl get gitrepositories -A
   ```

   If it persists, consider diving deeper into the Flux logs for more details or refer to the Flux general troubleshooting guide.

</td></tr>
</table>

Related resources

- Amazon EKS Blueprints for Terraform

- [Amazon EKS Best Practices Guides, Multi-tenancy section]
- [Flux website]
- [Helm website]

Additional information

Here's an example repository structure for deploying tenant applications:

```
applications
sample_tenant_app
### README.md
### base
#   ### configmap.yaml
#   ### deployment.yaml
#   ### ingress.yaml
#   ### kustomization.yaml
#   ### service.yaml
### overlays
    ### tenant-1
    #   ### configmap.yaml
    #   ### deployment.yaml
    #   ### kustomization.yaml
    ### tenant-2
        ### configmap.yaml
        ### kustomization.yaml
```

Subscribe multiple email endpoints to an SNS topic by using a custom resource

Ricardo Morais, Amazon Web Services

Summary

Note, August 2022: AWS CloudFormation now supports the subscription of multiple resources through the **AWS::SNS::Topic** object and its **Subscription** attribute.

This pattern describes how to subscribe multiple email addresses to receive notifications from an Amazon Simple Notification Service (Amazon SNS) topic. It uses an AWS Lambda function as a custom resource in an AWS CloudFormation template. The Lambda function is associated with an input parameter that specifies the email endpoints for the SNS topic.

Currently, you can use the AWS CloudFormation template objects AWS::SNS::Topic and AWS::SNS::Subscription to subscribe single endpoints to SNS topics. To subscribe multiple endpoints, you have to invoke the object multiple times. By using the Lambda function as a custom resource, you can subscribe multiple endpoints through an input parameter. You can use this Lambda function as a custom resource in any AWS CloudFormation template.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An AWS profile configured in your local environment with an access key and secret key.
- Permissions for the following:
  - AWS Identity and Access Management (IAM) role and policy
  - AWS Lambda function
  - Amazon Simple Storage Service (Amazon S3) for uploading the Lambda function
  - Amazon SNS topic and policy
  - AWS CloudFormation stacks

Limitations

- The code supports Linux and macOS workstations.

Product versions

- AWS Command Line Interface (AWS CLI) version 2 or later.

Architecture

Target technology stack

- AWS CloudFormation
- Amazon SNS
- AWS Lambda

Tools

Tools

- [AWS CLI version 2]

Code

The attachment includes the following files:

- Lambda function: `lambda_function.py`
- AWS CloudFormation template: `template.yaml`
- Two parameter files to handle multiple or single email endpoint subscriptions: `parameters-multiple-values.json` (used as the default) and `parameters-one-value.json`

To deploy the stack, you can use either parameter file. To specify multiple email endpoints:

```
./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION>
```

To specify a single email endpoint:

```
./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION> -f parameters-one-value.json
```

Epics

Option 1 - Deploy an SNS topic with one email subscription

Task	Description	Skills required
Configure the email endpoint for SNS topic subscriptions.	Edit the file `parameters-one-value.json` (attached), and change the value of the `pSNSNotificationsEmail` parameter to reflect the email address you want to use, such as `someone@example.com`.	
Deploy the AWS CloudFormation stack that creates the resources and subscription.	Run the **deploy.sh** command with your AWS profile name, AWS Region, and the `parameters-one-value.json` file. `./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION> -f parameters-one-value.json`	IAM role with proper permissions

Option 2 - Deploy an SNS topic with two or more email subscriptions

Task	Description	Skills required
Configure the email endpoints for SNS topic subscriptions.	Edit the file `parameters-multiple-values.json` (attached), and change the value of the `pSNSNotificationsE`	

Task	Description	Skills required
	`mail` parameter to reflect the email addresses you want to use, separated by commas, as follows: `someone1@example.com, someone2@example.com` .	
Deploy the AWS CloudFormation stack that creates the resources and subscription.	Run the **deploy.sh** command with your AWS profile name and AWS Region. You don't have to specify the `parameters-multiple-values.json` file because it's used by default. `./deploy.sh -p <YOUR_AWS_PROFILE_NAME> -r <YOUR_AWS_PROFILE_REGION>`	IAM role with proper permissions

Option 3 - Deploy an SNS topic through an AWS CloudFormation template

Task	Description	Skills required
Create an SNS topic.	Create an SNS topic through an AWS CloudFormation template, without specifying subscription endpoints in the `AWS::SNS::Topic` template object. You can use `template.yaml` in the attachment as a starting point.	IAM role with proper permissions

Task	Description	Skills required
Create an SNS topic policy.	Create an SNS topic policy in the AWS CloudFormation template.	IAM role with proper permissions
Subscribe the email endpoints list to the SNS topic.	Based on the list of email endpoints (one or more), subscribe the endpoints to the SNS topic you created.	IAM role with proper permissions

Related resources

References

- AWS CloudFormation custom resources (AWS documentation)
- AWS CloudFormation custom resource creation with Python, AWS Lambda, and crhelper (blog post)

Required tools

- AWS CLI version 2

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Coordinate resource dependency and task execution by using the AWS Fargate WaitCondition hook construct

Stan Fan, Amazon Web Services

Summary

This pattern describes the WaitCondition hook (`waitcondition-hook-for-aws-fargate-task`) npm package, which is a cloud-native solution designed for orchestrating AWS Fargate tasks in Amazon Elastic Container Service (Amazon ECS) clusters.

The WaitCondition hook is an AWS Cloud Development Kit (AWS CDK) construct that's specifically tailored for integration with AWS CloudFormation. The WaitCondition hook provides the following key capabilities:

- Acts as a wait condition mechanism, pausing CloudFormation stack execution until a specified Fargate task completes, which helps with orderly deployments and resource provisioning.
- Supports TypeScript and Python, making it ideal for AWS CDK projects.
- Allows developers and architects to orchestrate deployments by coordinating task completion and resource management for containerized applications on AWS.
- Enables running Fargate tasks with one or multiple containers embedded in a CloudFormation lifecycle. and can handle task failures and roll back the CloudFormation stack after a task failure.
- Provides flexibility to add dependencies between resources and the Fargate task execution results, enabling custom tasks or invoking other endpoints. For instance, you can pause a CloudFormation stack and wait for a database migration (done by a Fargate task) and provision other resources that might depend on the success of the database migration.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Cloud Development Kit (AWS CDK) Command Line Interface (CLI) installed on a local workstation. For more information, see AWS CDK CLI reference in the AWS CDK documentation.
- Node package manager (npm), installed on a local workstation and configured for the AWS CDK in TypeScript. For more information, see Downloading and installing Node.js and npm in the npm documentation.

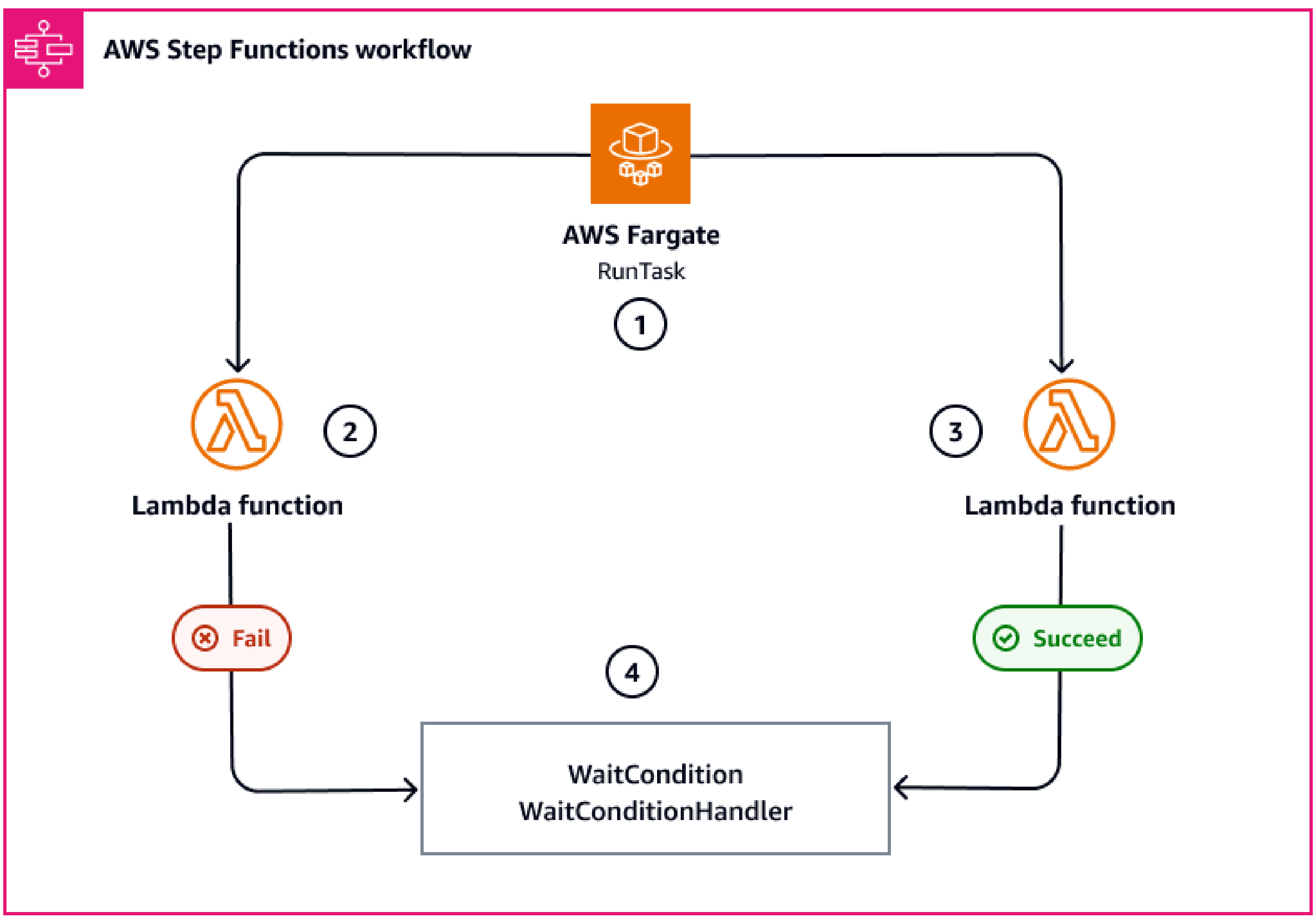
- Yarn installed on a local workstation. For more information, see Installation in the Yarn documentation.

Limitations

- This solution is deployed to a single AWS account.
- The expected return code of the container is `0` for success. Any other return code indicates failure, and the CloudFormation stack will roll back.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Architecture

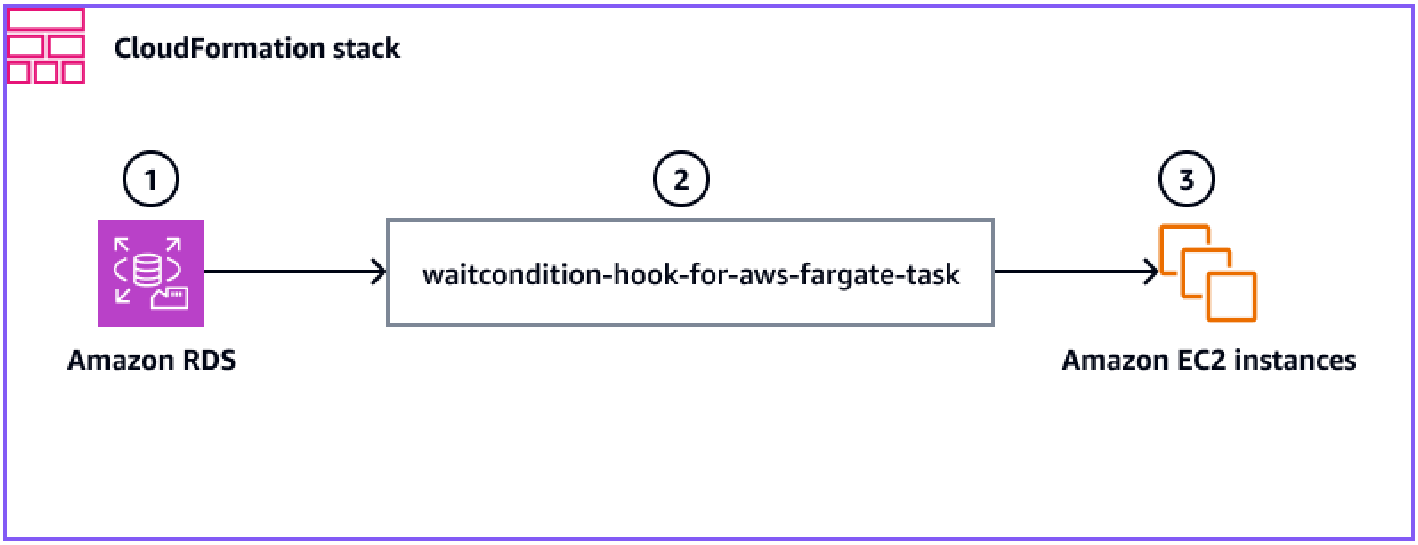
The following diagram shows the construct architecture.



The diagram shows the workflow of `waitcondition-hook-for-aws-fargate-task`:

1. `WaitCondition` and `WaitConditionHandler` are provisioned to listen to the response from the AWS Lambda functions.
2. Depending on the result of the task, either the `CallbackFunction` or the `ErrorHandlerFunction` is triggered by the finish of the Fargate task.
3. The Lambda function sends a SUCCEED or FAILURE signal to `WaitConditionHandler`.
4. `WaitConditionHandler` continues to provision the resources if the execution result of the Fargate task succeeds, or rolls back the stack if the task failed.

The following diagram shows an example of a workflow to perform a database migration.



The example workflow uses the `waitcondition-hook-for-aws-fargate-task` construct to perform a database migration, as follows:

1. An Amazon Relational Database Service (Amazon RDS) instance is provisioned.
2. The `waitcondition-hook-for-aws-fargate-task` construct runs the database migration task and pauses the stack as an Amazon Elastic Compute Cloud (Amazon EC2) instance.
3. If the migration task finishes successfully, it sends a Succeed signal to CloudFormation. Otherwise, it sends a Fail signal to CloudFormation and rolls back the stack.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define cloud infrastructure in code and provision it through AWS CloudFormation.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.
- AWS Fargate helps you run containers without needing to manage servers or Amazon EC2 instances. It's used in conjunction with Amazon ECS.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.
- Yarn is an open source package manager that you can use to manage dependencies in JavaScript projects. Yarn can assist you with installing, updating, configuring, and removing packages dependencies.

Code repository

The code for this pattern is available in the GitHub waitcondition-hook-for-aws-fargate-task repository.

Best practices

- When building your AWS CDK app, follow the Best practices for developing and deploying cloud infrastructure with the AWS CDK in the AWS CDK v2 documentation.
- For the AWS Fargate task, follow the Best practices for Amazon ECS container images in the Amazon ECS documentation.

Epics

Set up the AWS CDK

Task	Description	Skills required
Install the AWS CDK.	To install the AWS CDK on your local machine or other environment, run the following command: `npm install -g aws-cdk@latest`	Cloud architect, App developer
Bootstrap the AWS CDK.	Bootstrapping is the process of preparing an environment for deployment. To bootstrap your AWS CDK toolkit for the target AWS account and AWS Region, run the following command: `cdk bootstrap aws://ACCOUNT-NUMBER-1/REGION-1` This command creates a CloudFormation stack named `CDKToolkit` .	Cloud architect

Run the WaitCondition hook for AWS Fargate tasks construct

Task	Description	Skills required
Create the CDK project.	Create a CDK project using the language that you prefer. This pattern uses TypeScript. To create a CDK project using TypeScript, run the following command: `cdk init app —language typescript`	Cloud architect
Install the package.	Execute `npm install` on the root path of your CDK project. After the CDK library has been installed, run the following command to install `waitcondition-hook-for-aws-fargate-task` : `yarn add waitcondition-hook-for-aws-fargate-task`	Cloud architect
Build your CDK application and Amazon ECS components.	Build your CDK project. An Amazon ECS task definition resource is required. For information about creating a task definition, see [Amazon ECS task definitions](#) in the Amazon ECS documentation. The following example uses this construct:	Cloud architect

Task	Description	Skills required

	<pre>import * as cdk from
	 'aws-cdk-lib';
	import { Vpc } from
	 'aws-cdk-lib/aws-e
	c2';
	import * as ecr from
	 'aws-cdk-lib/aws-e
	cr';
	import * as ecs from
	 'aws-cdk-lib/aws-e
	cs';
	import { Construct }
	 from 'constructs';
	import { FargateRu
	nner } from 'waitcond
	ition-hook-for-aws-
	fargate-task';
	import { Queue } from
	 'aws-cdk-lib/aws-s
	qs';
	
	export class FargateRu
	nnerStack extends
	 cdk.Stack {
	    constructor(scope:
	 Construct, id: string,
	 props?: cdk.Stack
	Props) {
	        super(scope,
	 id, props);
	        // Define the
	 VPC
	        const vpc = new
	 Vpc(this, 'MyVpc')
	        // Define the
	 Fargate Task
	        const taskDefin
	ition = new ecs.Farga
	teTaskDefinition(t
	his, 'MyTask', {});</pre>	

Task	Description	Skills required
	``` // Import exiting ecr repo const repo = ecr.Repository.fro mRepositoryName(this, 'MyRepo', 'RepoName'); // Add a container to the task taskDefin ition.addContainer ('MyContainer', { image: ecs.ContainerImage .fromEcrRepository (repo), }); // Create the Fargate runner const myFargate Runner = new FargateRu nner(this, 'MyRunner ', { fargateTa skDef: taskDefinition, timeout: ` ${60 * 5}`, vpc: vpc, }); // Create the SQS queue const myQueue = new Queue(this, 'MyQueue', {}); // Add dependenc y myQueue.n ode.addDependency(myFargateRunner); } } ```	

Task	Description	Skills required

Synth and launch the CDK application.	1. To generate the assets and CloudFormation template, run the following command in your CDK root path:
	`cdk synth`
	2. After the `synth` command is successful, run the following command to deploy the resources:
	`cdk deploy`
	
	The `waitcondition-hook-for-aws-fargate-task` construct runs the Fargate task.
		Cloud architect

Clean up

Task	Description	Skills required
Clean up resources.	To clean up the resources provisioned from the previous step, run the following command: `cdk destroy`	Cloud architect

Troubleshooting

Issue	Solution
General CloudFormation stack failure	To help troubleshoot general CloudFormation stack failures, add the `--no-rollback` flag as shown in the following example: `cdk deploy --no-rollback` This command will pause the CloudForm ation stack from rolling back, which gives you resources to troubleshoot. For more informati on, see Choose how to handle failures when provisioning resources in the AWS CloudForm ation documentation.
AWS Step Functions failure	An AWS Step Functions state machine might fail to execute for different reasons. With –`disable-rollback` configured, use the following steps to troubleshoot: 1. Sign in to the AWS Management Console, enter **Step Functions** in the **Search** field, and then choose the Step Functions service. 2. In the left navigation pane, choose **State machines** and then select the state machine that is provisioned by the CloudFormation stack. 3. In **Executions**, choose the name of the execution that failed unexpectedly. 4. In the **Event** view, choose the failed step. For more information, see Troubleshooting issues in Step Functions and Viewing

Issue	Solution
	execution details in the Step Functions console in the AWS Step Functions documentation.
AWS Lambda function failure	This construct provisions two Lambda functions: `CallbackFunction` and `ErrorhandlerFunction` . They can fail for various reasons such as unhandled exceptions. Use the following steps to troubleshoot: 1. Sign in to the AWS Management Console, enter **CloudWatch** in the **Search** field, and then choose the CloudWatch service. 2. In the left navigation pane, choose **Log groups**. 3. In the **Search** field, enter the name of the Lambda function. 4. Choose the log group name that's associate d with the Lambda function. 5. To go to the Lambda function execution result, choose the latest log stream. For more information, see Troubleshooting issues in Lambda in the AWS Lambda documentation.

Related resources

AWS documentation

- AWS CDK Construct API Reference
- Getting started with the AWS CDK
- Learn how to create and use Amazon ECS resources
- Learn how to get started with Step Functions

- What is AWS CDK?

Other resources

- Waitcondition Hook for AWS Fargate task (npm)
- waitcondition-hook-for-aws-fargate-task 1.0.6 (pypi.org)

Use third-party Git source repositories in AWS CodePipeline

Kirankumar Chandrashekar, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern describes how to use AWS CodePipeline with third-party Git source repositories.

AWS CodePipeline is a continuous delivery service that automates tasks for building, testing, and deploying your software. The service currently supports Git repositories managed by GitHub, AWS CodeCommit, and Atlassian Bitbucket. However, some enterprises use third-party Git repositories that are integrated with their single sign-on (SSO) service and Microsoft Active Directory for authentication. You can use these third-party Git repositories as sources for CodePipeline by creating custom actions and webhooks.

A webhook is an HTTP notification that detects events in another tool, such as a GitHub repository, and connects those external events to a pipeline. When you create a webhook in CodePipeline, the service returns a URL that you can use in your Git repository webhook. If you push code to a specific branch of the Git repository, the Git webhook initiates the CodePipeline webhook through this URL, and sets the source stage of the pipeline to **In Progress**. When the pipeline is in this state, a job worker polls CodePipeline for the custom job, runs the job, and sends a success or failure status to CodePipeline. In this case, because the pipeline is in the source stage, the job worker gets the contents of the Git repository, zips the contents, and uploads it to the Amazon Simple Storage Service (Amazon S3) bucket where artifacts for the pipeline are stored, using the object key provided by the polled job. You can also associate a transition for the custom action with an event in Amazon CloudWatch, and initiate the job worker based on the event. This setup enables you to use third-party Git repositories that the service doesn't natively support as sources for CodePipeline.

Prerequisites and limitations

Prerequisites

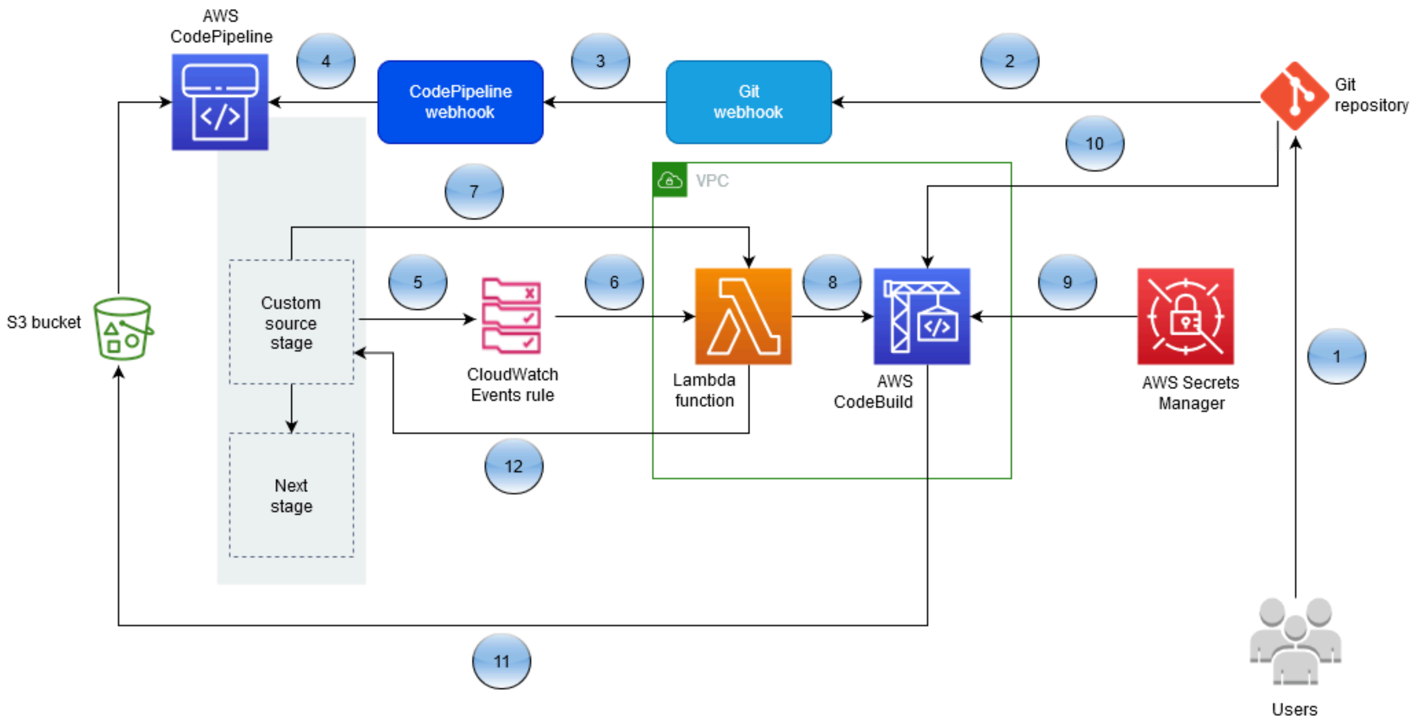
- An active AWS account
- A Git repository that supports webhooks and can connect to a CodePipeline webhook URL through the internet

- AWS Command Line Interface (AWS CLI) installed and configured to work with the AWS account

Architecture

The pattern involves these steps:

1. The user commits code to a Git repository.
2. The Git webhook is called.
3. The CodePipeline webhook is called.
4. The pipeline is set to **In Progress**, and the source stage is set to the **In Progress** state.
5. The source stage action initiates a CloudWatch Events rule, indicating that it was started.
6. The CloudWatch event initiates a Lambda function.
7. The Lambda function gets the details of the custom action job.
8. The Lambda function initiates AWS CodeBuild and passes it all the job-related information.
9. CodeBuild gets the public SSH key or user credentials for HTTPS Git access from Secrets Manager.
10. CodeBuild clones the Git repository for a specific branch.
11. CodeBuild zips the archive and uploads it to the S3 bucket that serves as the CodePipeline artifact store.



Tools

- [AWS CodePipeline](#) – AWS CodePipeline is a fully managed [continuous delivery](#) service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deployment phases of your release process for each code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. You can integrate AWS CodePipeline with third-party services such as GitHub or with your own custom plugin.
- [AWS Lambda](#) – AWS Lambda lets you run code without provisioning or managing servers. With Lambda, you can run code for virtually any type of application or backend service with no administration necessary. You upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically initiate from other AWS services or call it directly from any web or mobile app.
- [AWS CodeBuild](#) – AWS CodeBuild is a fully managed [continuous integration](#) service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools.

- AWS Secrets Manager – AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. The service enables you to rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets by calling Secrets Manager APIs, without having to hardcode sensitive information in plain text. Secrets Manager offers secret rotation with built-in integration for Amazon Relational Database Service (Amazon RDS), Amazon Redshift, and Amazon DocumentDB. The service can be extended to support other types of secrets, including API keys and OAuth tokens. In addition, Secrets Manager lets you control access to secrets by using fine-grained permissions, and audit secret rotation centrally for resources in the AWS Cloud, third-party services, and on-premises environments.
- Amazon CloudWatch – Amazon CloudWatch is a monitoring and observation service built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers. CloudWatch provides you with data and actionable insights to monitor your applications, respond to systemwide performance changes, optimize resource utilization, and get a unified view of operational health. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, providing you with a unified view of AWS resources, applications, and services that run on AWS and on-premises servers. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service that lets you store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features to help you organize your data and configure finely tuned access controls to meet your specific business, organizational, and compliance requirements.

Epics

Create a custom action in CodePipeline

Task	Description	Skills required
Create a custom action using AWS CLI or AWS CloudForm ation.	This step involves creating a custom source action that can be used in the source	General AWS

Task	Description	Skills required
	stage of a pipeline in your AWS account in a particula r region. You must use AWS CLI or AWS CloudForm ation (not the console) to create the custom source action. For more informati on about the commands and steps described in this and other epics, see the "Related resources" section at the end of this pattern. In AWS CLI, use the create-custom-action-type command. Use --configu ration-properties to provide all the parameters required for the job worker to process when it polls CodePipeline for a job. Make sure to note the values provided to the --provider and --action-version options, so that you can use the same values when creating the pipeline with this custom source stage. You can also create the custom source action in AWS CloudForm ation by using the resource type AWS::CodePipeline::CustomActionType.	

Set up authentication

Task	Description	Skills required
Create an SSH key pair.	Create a Secure Shell (SSH) key pair. For instructions, see the GitHub documentation.	Systems/DevOps engineer
Create a secret in AWS Secrets Manager.	Copy the contents of the private key from the SSH key pair and create a secret in AWS Secrets Manager. This secret is used for authentic ation when accessing the Git repository.	General AWS
Add the public key to the Git repository.	Add the public key from the SSH key pair to the Git repository account settings, for authentication against the private key.	Systems/DevOps engineer

Create a pipeline and webhook

Task	Description	Skills required
Create a pipeline that includes the custom source action.	Create a pipeline in CodePipel ine. When you configure the source stage, choose the custom source action that you created previously. You can do this in the AWS CodePipel ine console or in AWS CLI. CodePipeline prompts you for the configuration propertie s that you set on the custom action. This information is	General AWS

Task	Description	Skills required
	required for the job worker to process the job for the custom action. Follow the wizard and create the next stage for the pipeline.	
Create a CodePipeline webhook.	Create a webhook for the pipeline you created with the custom source action. You must use AWS CLI or AWS CloudFormation (not the console) to create the webhook. In AWS CLI, run the put-webhook command and provide the appropria te values for the webhook options. Make a note of the webhook URL that the command returns. If you're using AWS CloudFormation to create the webhook, use the resource type AWS::Code Pipeline::Webhook. Make sure to output the webhook URL from the created resource, and make a note of it.	General AWS

Task	Description	Skills required
Create a Lambda function and CodeBuild project.	In this step, you use Lambda and CodeBuild to create a job worker that will poll CodePipeline for job requests for the custom action, run the job, and return the status result to CodePipeline. Create a Lambda function that is initiated by an Amazon CloudWatch Events rule when the custom source action stage of the pipeline transitions to "In Progress." When the Lambda function is initiated, it should get the custom action job details by polling for jobs. You can use the PollForJobs API to return this information. After the polled job information is obtained, the Lambda function should return an acknowledgment, and then process the information with the data it obtains from the configuration properties for the custom action. When the worker is ready to talk to the Git repository, you might initiate a CodeBuild project, because it's convenient to handle Git tasks by using the SSH client.	General AWS, code developer

Create an event in CloudWatch

Task	Description	Skills required
Create a CloudWatch Events rule.	Create a CloudWatch Events rule that initiates the Lambda function as a target whenever the pipeline's custom action stage transitions to "In Progress."	General AWS

Related resources

Creating a custom action in CodePipeline

- Create and add a custom action in CodePipeline
- AWS::CodePipeline::CustomActionType resource

Setting up authentication

- Creating and Managing Secrets with AWS Secrets Manager

Creating a pipeline and webhook

- Create a Pipeline in CodePipeline
- put-webhook command reference
- AWS::CodePipeline::Webhook resource
- PollForJobs API reference
- Create and Add a Custom Action in CodePipeline
- Create a build project in AWS CodeBuild

Creating an event

- Detect and react to changes in pipeline state with Amazon CloudWatch Events

Additional references

- Working with pipelines in CodePipeline
- AWS Lambda developer guide

Create a CI/CD pipeline to validate Terraform configurations by using AWS CodePipeline

Aromal Raj Jayarajan and Vijesh Vijayakumaran Nair, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern shows how to test HashiCorp Terraform configurations by using a continuous integration and continuous delivery (CI/CD) pipeline deployed by AWS CodePipeline.

Terraform is a command-line interface application that helps you use code to provision and manage cloud infrastructure and resources. The solution provided in this pattern creates a CI/CD pipeline that helps you validate the integrity of your Terraform configurations by running five CodePipeline stages:

1. `"checkout"` pulls the Terraform configuration that you're testing from an AWS CodeCommit repository.
2. `"validate"` runs infrastructure as code (IaC) validation tools, including tfsec, TFLint, and checkov. The stage also runs the following Terraform IaC validation commands: `terraform validate` and `terraform fmt`.
3. `"plan"` shows what changes will be applied to the infrastructure if the Terraform configuration is applied.
4. `"apply"` uses the generated plan to provision the required infrastructure in a test environment.
5. `"destroy"` removes the test infrastructure that was created during the `"apply"` stage.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI), installed and configured
- Git, installed and configured on your local machine
- Terraform, installed and configured on your local machine

Limitations

- This pattern's approach deploys AWS CodePipeline into one AWS account and AWS Region only. Configuration changes are required for multi-account and multi-Region deployments.
- The AWS Identity and Access Management (IAM) role that this pattern provisions (**codepipeline_iam_role**) follows the principle of least privilege. This IAM role's permissions must be updated based on the specific resources that your pipeline needs to create.

Product versions

- AWS CLI version 2.9.15 or later
- Terraform version 1.3.7 or later

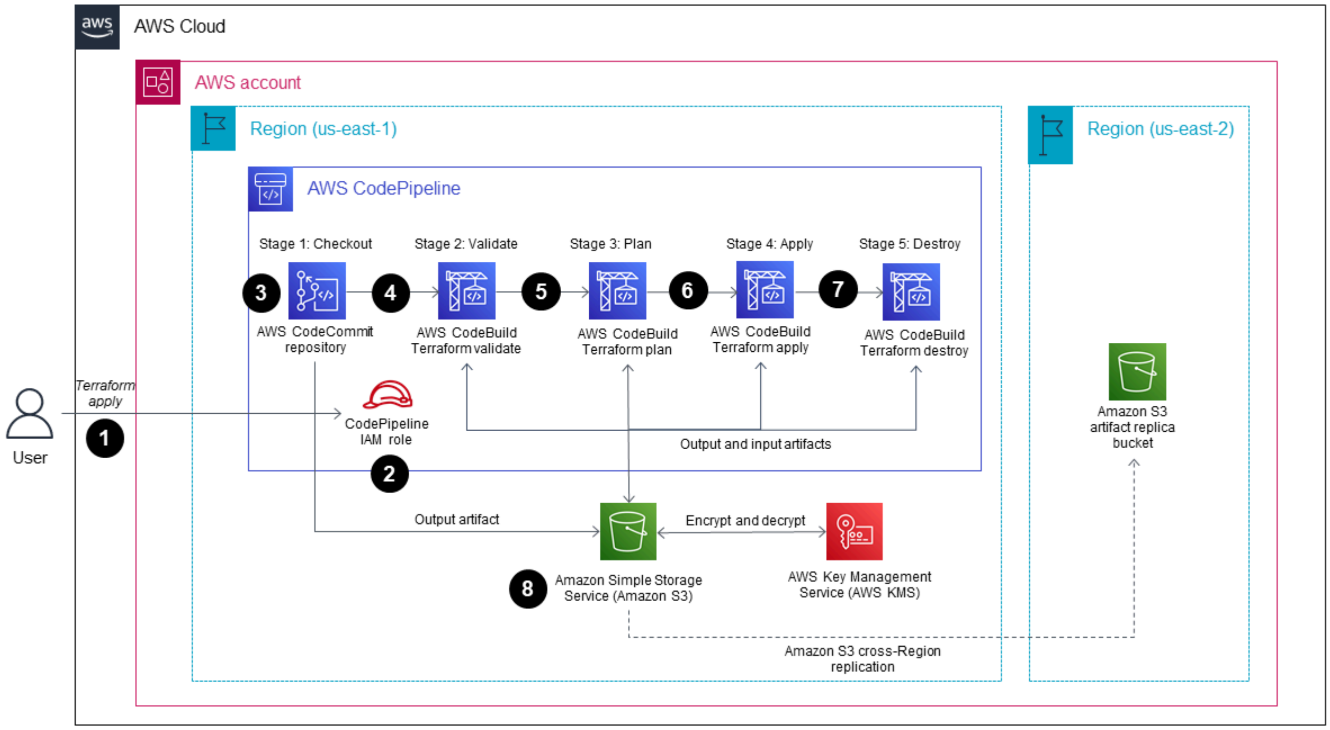
Architecture

Target technology stack

- AWS CodePipeline
- AWS CodeBuild
- AWS CodeCommit
- AWS IAM
- Amazon Simple Storage Service (Amazon S3)
- AWS Key Management Service (AWS KMS)
- Terraform

Target architecture

The following diagram shows an example CI/CD pipeline workflow for testing Terraform configurations in CodePipeline.



The diagram shows the following workflow:

1. In CodePipeline, an AWS user initiates the actions proposed in a Terraform plan by running the `terraform apply` command in the AWS CLI.
2. AWS CodePipeline assumes an IAM service role that includes the policies required to access CodeCommit, CodeBuild, AWS KMS, and Amazon S3.
3. CodePipeline runs the `"checkout"` pipeline stage to pull the Terraform configuration from an AWS CodeCommit repository for testing.
4. CodePipeline runs the `"validate"` stage to test the Terraform configuration by running IaC validation tools and running Terraform IaC validation commands in a CodeBuild project.
5. CodePipeline runs the `"plan"` stage to create a plan in the CodeBuild project based on the Terraform configuration. The AWS user can review this plan before the changes are applied to the test environment.
6. Code Pipeline runs the `"apply"` stage to implement the plan by using the CodeBuild project to provision the required infrastructure in the test environment.
7. CodePipeline runs the `"destroy"` stage, which uses CodeBuild to remove the test infrastructure that was created during the `"apply"` stage.

8. An Amazon S3 bucket stores pipeline artifacts, which are encrypted and decrypted by using an AWS KMS customer managed key.

Tools

Tools

AWS services

- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other services

- HashiCorp Terraform is a command-line interface application that helps you use code to provision and manage cloud infrastructure and resources.

Code

The code for this pattern is available in the GitHub aws-codepipeline-terraform-cicdsamples repository. The repository contains the Terraform configurations required to create the target architecture outlined in this pattern.

Epics

Provision the solution components

Task	Description	Skills required
Clone the GitHub repository.	Clone the GitHub [aws-codepipeline-terraform-cicdsamples](https://github.com/aws-samples/aws-codepipeline-terraform-cicd-samples) repository by running the following command in a terminal window: `git clone https://github.com/aws-samples/aws-codepipeline-terraform-cicd-samples.git` For more information, see [Cloning a repository](#) in the GitHub documentation.	DevOps engineer
Create a Terraform variable definitions file.	Create a `terraform.tfvars` file based on your use case requirements. You can update the variables in the `examples/terraform.tfvars` file that's in the cloned repository. For more information, see [Assigning values to root module variables](#) in the Terraform documentation. **Note** The repository's `Readme.md` file	DevOps engineer

Task	Description	Skills required
	includes more information on the required variables.	
Configure AWS as the Terraform provider.	1. In a code editor, open the cloned repository's `main.tf` file. 2. Add the necessary configurations for establishing connectivity to the target AWS account. For more information, see AWS provider in the Terraform documentation.	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Update the Terraform provider configuration for creating the Amazon S3 replication bucket.</td><td>
1. Open the repository’s S3 directory by running the following command:
<code>cd ./modules/s3</code>
2. Update the Terraform provider configuration for creating the Amazon S3 replication bucket by updating the <code>region</code> value in the <code>tf</code> file. Make sure that you enter the Region that you want Amazon S3 to replicate objects to.
3. (Optional) By default, Terraform uses local state files for state managemen t. If you want to add Amazon S3 as the remote backend, you must update the Terraform configura tion. For more informati on, see Backend configura tion in the Terraform documentation.
Note
Replication activates automatic, asynchron ous copying of objects
</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	across Amazon S3 buckets.	
Initialize the Terraform configuration.	To initialize your working directory that contains the Terraform configuration files, run the following command in the cloned repository's root folder: `terraform init`	DevOps engineer
Create the Terraform plan.	To create a Terraform plan, run the following command in the cloned repository's root folder: `terraform plan --var-file=terraform.tfvars -out=tfplan` **Note** Terraform evaluates the configuration files to determine the target state for the declared resources. It then compares the target state against the current state and creates a plan.	DevOps engineer

Task	Description	Skills required
Verify the Terraform plan.	Review the Terraform plan and confirm that it configure s the required architecture in your target AWS account.	DevOps engineer
Deploy the solution.	1. To apply the Terraform plan, run the following command in the cloned repository’s root folder: `terraform apply "tfplan"` 2. Enter **yes** to confirm that you want to deploy the resources. **Note** Terraform creates, updates, or destroys infrastructure to achieve the target state declared in the configuration files.	DevOps engineer

Validate Terraform configurations by running the pipeline

Task	Description	Skills required
Set up the source code repository.	1. From the Terraform output, get the **source repository** details for the repository that contains the Terraform	DevOps engineer

Task	Description	Skills required
	configurations that you want to validate. 2. Sign in to the AWS Management Console. Then, open the [CodeCommit console](). 3. Create a new branch in the source repository named `main`. For instructions, see [Create a branch in AWS CodeCommit]() in the CodeCommit documentation. 4. Clone the `main` branch of the **source repository** to your local workstation. For instructions, see [Setup steps for HTTPS connections to AWS CodeCommit repositories on Windows with the AWS CLI credential helper]() in the CodeCommit documentation. 5. Copy the `templates` folder from the GitHub [aws-codepipeline-terraform-cicdsamples]() repository by running the following command: `cp -r templates $YOUR_CODECOMMIT_REPO_ROOT`	

Task	Description	Skills required
	Note The `templates` folder contains the build specification files and the validation script for the root directory of the source repository. 6. Add your required Terraform IaC configurations to the root folder of the **source repository**. 7. Add the details for the remote backend in your project's Terraform configuration. For more information, see S3 in the Terraform documentation. 8. **(Optional)** Update the variables in the `templates` folder to either activate or deactivate the preconfigured scans, tool change versions, and to specify your directory in custom script files. For more information, see the *Additional information* section of this pattern.	

Task	Description	Skills required
	9. Push the changes to the `main` branch of the **source repository**.	

Task	Description	Skills required
Validate the pipeline stages.	1. Sign in to the AWS Management Console and open the [CodePipeline console](). 2. In the output generated from the `terraform apply "tfplan"` command in the previous Epic section, find the name of the generated **CodePipeline**. 3. Open the pipeline in the CodePipeline console and choose **Release change**. 4. Review each pipeline stage and confirm it's working as expected. For more information, see [View pipeline details and history (console)]() in the *AWS CodePipeline User Guide*. **Important** When a change is committed to the main branch of the source repository, the test pipeline is activated automatic ally.	DevOps engineer

Task	Description	Skills required
Verify the report output.	1. On the [CodePipeline console](#), in the left navigation pane, choose **Build**. Then, choose **Report history**. 2. Review the tfsec and checkov scan reports that the pipeline generates. These reports can help you identify issues through visualizations and graphical representations. **Note** The `<project_name>-validate` CodeBuild project generates vulnerability reports for your code during the `"validate"` stage.	DevOps engineer

Clean up your resources

Task	Description	Skills required
Clean up the pipeline and associated resources.	To delete the test resources from your AWS account, run the following command in the cloned repository's root folder:	DevOps engineer

Task	Description	Skills required
	`terraform destroy --var-file=terraform.tfvars`	

Troubleshooting

Issue	Solution
You receive an **AccessDenied** error during the `"apply"` stage.	1. Review the execution logs of the CodeBuild project associated with the `"apply"` stage to identify any missing IAM permissions. For more information, see View build details in AWS CodeBuild in the *AWS CodeBuild User Guide*. 2. In a code editor, open the cloned repository's `modules` folder. Then, navigate to the `iam-role` folder and open the `main.tf` file that's in that folder. 3. In the `codepipeline_policy` statement, add the IAM policies that are required for provisioning resources in your AWS account.

Related resources

- Module blocks (Terraform documentation)
- How to use CI/CD to deploy and configure AWS security services with Terraform (AWS blog post)
- Using service-linked roles (IAM documentation)
- create-pipeline (AWS CLI documentation)
- Configure server-side encryption for artifacts stored in Amazon S3 for CodePipeline (AWS CodePipeline documentation)
- Quotas for AWS CodeBuild (AWS CodeBuild documentation)

- [Data protection in AWS CodePipeline](#) (AWS CodePipeline documentation)

Additional information

Custom Terraform modules

The following is a list of custom Terraform modules that are used in this pattern:

- `codebuild_terraform` creates the CodeBuild projects that form each stage of the pipeline.
- `codecommit_infrastructure_source_repo` captures and creates the source CodeCommit repository.
- `codepipeline_iam_role` creates the required IAM roles for the pipeline.
- `codepipeline_kms` creates the required AWS KMS key for Amazon S3 object encryption and decryption.
- `codepipeline_terraform` creates the test pipeline for the source CodeCommit repository.
- `s3_artifacts_bucket` creates an Amazon S3 bucket to manage pipeline artifacts.

Build specification files

The following is a list of build specification (buildspec) files that this pattern uses to run each pipeline stage:

- `buildspec_validate.yml` runs the "`validate`" stage.
- `buildspec_plan.yml` runs the "`plan`" stage.
- `buildspec_apply.yml` runs the "`apply`" stage.
- `buildspec_destroy.yml` runs the "`destroy`" stage.

Build specification file variables

Each buildspec file uses the following variables to activate different build-specific settings:

Variable	Default value	Description
`CODE_SRC_DIR`	"."	Defines the source CodeCommit directory

`TF_VERSION`	"1.3.7"	Defines the Terraform version for the build environment

The `buildspec_validate.yml` file also supports the following variables to activate different build-specific settings:

Variable	Default value	Description
`SCRIPT_DIR`	"./templates/scripts"	Defines the script directory
`ENVIRONMENT`	"dev"	Defines the environment name
`SKIPVALIDATIONFAILURE`	"Y"	Skips validation on failures
`ENABLE_TFVALIDATE`	"Y"	Activates Terraform validate
`ENABLE_TFFORMAT`	"Y"	Activates Terraform format
`ENABLE_TFCHECKOV`	"Y"	Activates checkov scan
`ENABLE_TFSEC`	"Y"	Activates tfsec scan
`TFSEC_VERSION`	"v1.28.1"	Defines the tfsec version

More patterns

- Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer
- Associate an AWS CodeCommit repository in one AWS account with Amazon SageMaker AI Studio Classic in another account
- Automate account creation by using the Landing Zone Accelerator on AWS
- Automate adding or updating Windows registry entries using AWS Systems Manager
- Automate backups for Amazon RDS for PostgreSQL DB instances by using AWS Batch
- Automate deployment of nested applications using AWS SAM
- Automate deployment of Node Termination Handler in Amazon EKS by using a CI/CD pipeline
- Automate RabbitMQ configuration in Amazon MQ
- Automate the replication of Amazon RDS instances across AWS accounts
- Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline
- Automatically generate a PynamoDB model and CRUD functions for Amazon DynamoDB by using a Python application
- Automatically validate and deploy IAM policies and roles by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros
- Back up Sun SPARC servers in the Stromasys Charon-SSP emulator on the AWS Cloud
- Build a data pipeline to ingest, transform, and analyze Google Analytics data using the AWS DataOps Development Kit
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager
- Build a pipeline for hardened container images using EC2 Image Builder and Terraform
- Build an MLOps workflow by using Amazon SageMaker AI and Azure DevOps
- Centralize DNS resolution by using AWS Managed Microsoft AD and on-premises Microsoft Active Directory
- Chain AWS services together using a serverless approach
- Clean up AWS Account Factory for Terraform (AFT) resources safely after state file loss
- Configure logging for .NET applications in Amazon CloudWatch Logs by using NLog
- Continuously deploy a modern AWS Amplify web application from an AWS CodeCommit repository

- Create a custom Docker container image for SageMaker and use it for model training in AWS Step Functions
- Create a pipeline in AWS Regions that don't support AWS CodePipeline
- Create alarms for custom metrics using Amazon CloudWatch anomaly detection
- Customize default role names by using AWS CDK aspects and escape hatches
- Deploy a pipeline that simultaneously detects security issues in multiple code deliverables
- Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code
- Deploy Kubernetes resources and packages using Amazon EKS and a Helm chart repository in Amazon S3
- Deploy multiple-stack applications using AWS CDK with TypeScript
- Deploy SQL Server failover cluster instances on Amazon EC2 and Amazon FSx by using Terraform
- Deploy the Security Automations for AWS WAF solution by using Terraform
- Develop advanced generative AI chat-based assistants by using RAG and ReAct prompting
- Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates
- Set up event-driven auto scaling in Amazon EKS by using Amazon EKS Pod Identity and KEDA
- Generate personalized and re-ranked recommendations using Amazon Personalize
- Get Amazon SNS notifications when the key state of an AWS KMS key changes
- Govern permission sets for multiple accounts by using Account Factory for Terraform
- Identify duplicate container images automatically when migrating to an Amazon ECR repository
- Implement path-based API versioning by using custom domains in Amazon API Gateway
- Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK
- Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet
- Integrate Stonebranch Universal Controller with AWS Mainframe Modernization
- Mainframe modernization: DevOps on AWS with Rocket Software Enterprise Suite
- Manage AWS IAM Identity Center permission sets as code by using AWS CodePipeline
- Manage AWS permission sets dynamically by using Terraform
- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK
- Manage AWS Organizations policies as code by using AWS CodePipeline and Amazon Bedrock
- Migrate DNS records in bulk to an Amazon Route 53 private hosted zone

- Monitor use of a shared Amazon Machine Image across multiple AWS accounts
- Orchestrate an ETL pipeline with validation, transformation, and partitioning using AWS Step Functions
- Automate blue/green deployments of Amazon Aurora global databases by using IaC principles
- Preserve routable IP space in multi-account VPC designs for non-workload subnets
- Provision a Terraform product in AWS Service Catalog by using a code repository
- Replicate filtered Amazon ECR container images across accounts or Regions
- Rotate database credentials without restarting containers
- Run AWS Systems Manager Automation tasks synchronously from AWS Step Functions
- Set up a CI/CD pipeline for hybrid workloads on Amazon ECS Anywhere by using AWS CDK and GitLab
- Set up a CI/CD pipeline for database migration by using Terraform
- Set up Multi-AZ infrastructure for a SQL Server Always On FCI by using Amazon FSx
- Set up UiPath RPA bots automatically on Amazon EC2 by using AWS CloudFormation
- Simplify application authentication with mutual TLS in Amazon ECS by using Application Load Balancer
- Tenant onboarding in SaaS architecture for the silo model using C# and AWS CDK
- Use Terraform to automatically enable Amazon GuardDuty for an organization
- Use Amazon Bedrock agents to automate creation of access entry controls in Amazon EKS through text-based prompts
- Validate Account Factory for Terraform (AFT) code locally
- Visualize AI/ML model results using Flask and AWS Elastic Beanstalk

Infrastructure

Topics

- Access a bastion host by using Session Manager and Amazon EC2 Instance Connect
- Centralize DNS resolution by using AWS Managed Microsoft AD and on-premises Microsoft Active Directory
- Centralize monitoring by using Amazon CloudWatch Observability Access Manager
- Check EC2 instances for mandatory tags at launch
- Clean up AWS Account Factory for Terraform (AFT) resources safely after state file loss
- Connect to an Amazon EC2 instance by using Session Manager
- Create a pipeline in AWS Regions that don't support AWS CodePipeline
- Customize default role names by using AWS CDK aspects and escape hatches
- Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing
- Extend VRFs to AWS by using AWS Transit Gateway Connect
- Get Amazon SNS notifications when the key state of an AWS KMS key changes
- Preserve routable IP space in multi-account VPC designs for non-workload subnets
- Provision a Terraform product in AWS Service Catalog by using a code repository
- Register multiple AWS accounts with a single email address by using Amazon SES
- Set up DNS resolution for hybrid networks in a single-account AWS environment
- Set up UiPath RPA bots automatically on Amazon EC2 by using AWS CloudFormation
- Set up a highly available PeopleSoft architecture on AWS
- Set up disaster recovery for Oracle JD Edwards EnterpriseOne with AWS Elastic Disaster Recovery
- Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization
- Successfully import an S3 bucket as an AWS CloudFormation stack
- Synchronize data between Amazon EFS file systems in different AWS Regions by using AWS DataSync
- Test AWS infrastructure by using LocalStack and Terraform Tests
- Upgrade SAP Pacemaker clusters from ENSA1 to ENSA2
- Use consistent Availability Zones in VPCs across different AWS accounts
- Use user IDs in IAM policies for access control and automation
- Validate Account Factory for Terraform (AFT) code locally

- More patterns

Access a bastion host by using Session Manager and Amazon EC2 Instance Connect

Piotr Chotkowski and Witold Kowalik, Amazon Web Services

Summary

A *bastion host*, sometimes called a *jump box*, is a server that provides a single point of access from an external network to the resources located in a private network. A server exposed to an external public network, such as the internet, poses a potential security risk for unauthorized access. It's important to secure and control access to these servers.

This pattern describes how you can use Session Manager and Amazon EC2 Instance Connect to securely connect to an Amazon Elastic Compute Cloud (Amazon EC2) bastion host deployed in your AWS account. Session Manager is a capability of AWS Systems Manager. The benefits of this pattern include:

- The deployed bastion host doesn't have any open, inbound ports exposed to the public internet. This reduces the potential attack surface.
- You don't need to store and maintain long-term Secure Shell (SSH) keys in your AWS account. Instead, each user generates a new SSH key pair each time they connect to the bastion host. AWS Identity and Access Management (IAM) policies that are attached to the user's AWS credentials control access to the bastion host.

Intended audience

This pattern is intended for readers who have experience with basic understanding of Amazon EC2, Amazon Virtual Private Cloud (Amazon VPC), and Hashicorp Terraform.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2, installed and configured
- Session Manager plugin for the AWS CLI, installed
- Terraform CLI, installed

- Storage for the Terraform state, such as an Amazon Simple Storage Service (Amazon S3) bucket and an Amazon DynamoDB table that serve as a remote backend to store the Terraform state. For more information on using remote backends for the Terraform state, see Amazon S3 Backends (Terraform documentation). For a code sample that sets up remote state management with an Amazon S3 backend, see remote-state-s3-backend (Terraform Registry). Note the following requirements:
  - The Amazon S3 bucket and DynamoDB table must be in the same AWS Region.
  - When creating the DynamoDB table, the partition key must be `LockID` (case-sensitive), and the partition key type must be `String`. All other table settings must be at their default values. For more information, see About primary keys and Create a table in the DynamoDB documentation.
- An SSH client, installed

Limitations

- This pattern is intended as a proof of concept (PoC) or as a basis for further development. It should not be used in its current form in production environments. Before deployment, adjust the sample code in the repository to meet your requirements and use case.
- This pattern assumes that the target bastion host uses Amazon Linux 2 as its operating system. While it is possible to use other Amazon Machine Images (AMIs), other operating systems are out of scope for this pattern.

> **Note**
>
> Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs.

- In this pattern, the bastion host is located in a private subnet without an NAT gateway and internet gateway. This design isolates the Amazon EC2 instance from the public internet. You can add a specific network configuration that allows it to communicate with the internet. For more information, see Connect your virtual private cloud (VPC) to other networks in the Amazon VPC documentation. Similarly, following the principle of least privilege, the bastion host doesn't have access to any other resources in your AWS account unless you explicitly grant permissions. For more information, see Resource-based policies in the IAM documentation.

Product versions

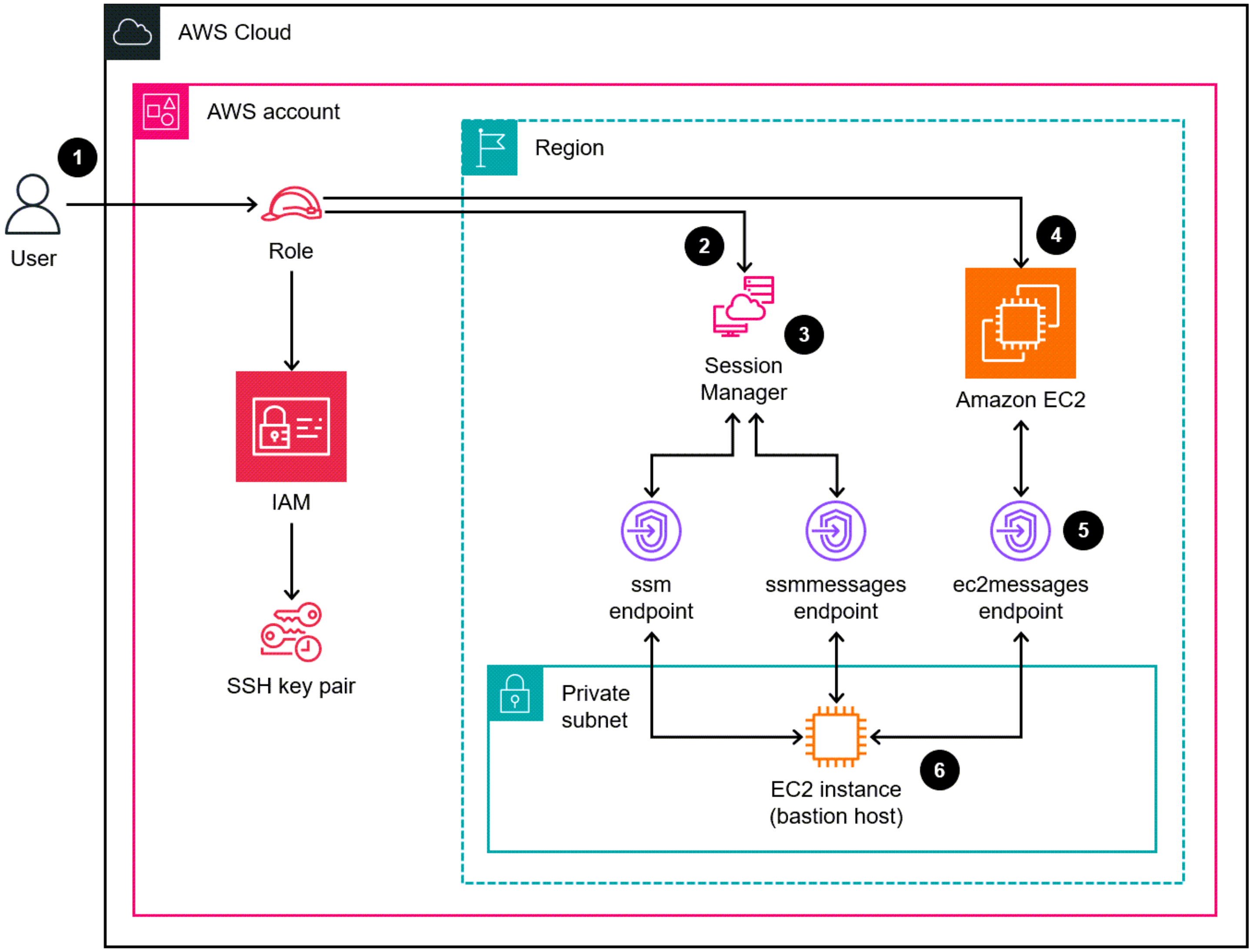
- AWS CLI version 2
- Terraform version 1.3.9

Architecture

Target technology stack

- A VPC with a single private subnet
- The following interface VPC endpoints:
  - `amazonaws.<region>.ssm` – The endpoint for the AWS Systems Manager service.
  - `amazonaws.<region>.ec2messages` – Systems Manager uses this endpoint to make calls from SSM Agent to the Systems Manager service.
  - `amazonaws.<region>.ssmmessages` – Session Manager uses this endpoint to connect to your Amazon EC2 instance through a secure data channel.
- A `t3.nano` Amazon EC2 instance running Amazon Linux 2
- IAM role and instance profile
- Amazon VPC security groups and security group rules for the endpoints and Amazon EC2 instance

Target architecture



The diagram shows the following process:

1. The user assumes an IAM role that has permissions to do the following:
   - Authenticate, authorize, and connect to the Amazon EC2 instance
   - Start a session with Session Manager
2. The user initiates an SSH session through Session Manager.
3. Session Manager authenticates the user, verifies the permissions in the associated IAM policies, checks the configuration settings, and sends a message to SSM Agent to open a two-way connection.
4. The user pushes the SSH public key to the bastion host through Amazon EC2 metadata. This must be done before each connection. The SSH public key remains available for 60 seconds.

5. The bastion host communicates with the interface VPC endpoints for Systems Manager and Amazon EC2.
6. The user accesses the bastion host through Session Manager by using a TLS 1.2 encrypted bidirectional communication channel.

Automation and scale

The following options are available to automate deployment or to scale this architecture:

- You can deploy the architecture through a continuous integration and continuous delivery (CI/CD) pipeline.
- You can modify the code to change the instance type of the bastion host.
- You can modify the code to deploy multiple bastion hosts. In the `bastion-host/main.tf` file, in the `aws_instance` resource block, add the `count` meta-argument. For more information, see the Terraform documentation.

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale. This pattern uses Session Manager, a capability of Systems Manager.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Other tools

- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources. This pattern uses Terraform CLI.

Code repository

The code for this pattern is available in the GitHub Access a bastion host by using Session Manager and Amazon EC2 Instance Connect repository.

Best practices

- We recommend using automated code-scanning tools to improve the security and quality of the code. This pattern was scanned by using Checkov, a static code-analysis tool for IaC. At a minimum, we recommend that you perform basic validation and formatting checks by using the `terraform validate` and `terraform fmt -check -recursive` Terraform commands.
- It's a good practice to add automated tests for IaC. For more information about the different approaches for testing Terraform code, see Testing HashiCorp Terraform (Terraform blog post).
- During deployment, Terraform uses the replaces the Amazon EC2 instance each time a new version of the Amazon Linux 2 AMI is detected. This deploys the new version of the operating system, including patches and upgrades. If the deployment schedule is infrequent, this can pose a security risk because the instance doesn't have the latest patches. It is important to frequently update and apply security patches to deployed Amazon EC2 instances. For more information, see Update management in Amazon EC2.
- Because this pattern is a proof of concept, it uses AWS managed policies, such as `AmazonSSMManagedInstanceCore`. AWS managed policies cover common use cases but don't grant least-privilege permissions. As needed for your use case, we recommend that you create custom policies that grant least-privilege permissions for the resources deployed in this architecture. For more information, see Get started with AWS managed policies and move toward least-privilege permissions.
- Use a password to protect access to SSH keys and store keys in a secure location.
- Set up logging and monitoring for the bastion host. Logging and monitoring are important parts of maintaining systems, from both an operational and security perspective. There are multiple ways to monitor connections and activity in your bastion host. For more information, see the following topics in the Systems Manager documentation:
  - Monitoring AWS Systems Manager
  - Logging and monitoring in AWS Systems Manager
  - Auditing session activity

- Logging session activity

Epics

Deploy the resources

Task	Description	Skills required
Clone the code repository.	1. In a command-line interface, change your working directory to the location where you want to store the sample files. 2. Enter the following command. `git clone https://github.com/aws-samples/secured-bastion-host-terraform.git`	DevOps engineer, Developer
Initialize the Terraform working directory.	This step is necessary for only the first deployment. If you are redeploying the pattern, skip to the next step. In the root directory of the cloned repository, enter the following command, where: • `$S3_STATE_BUCKET` is the name of Amazon S3 bucket that contains the Terraform state • `$PATH_TO_STATE_FILE` is the key to the Terraform	DevOps engineer, Developer, Terraform

Task	Description	Skills required
	state file, such as `infra/bastion-host/tetfstate` • `$AWS_REGION` is the Region where the Amazon S3 bucket is deployed `terraform init \ -backend-config="bucket=$S3_STATE_BUCKET" \ -backend-config="key=$PATH_TO_STATE_FILE" \ -backend-config="region=$AWS_REGION` **Note** Alternatively, you can open the **config.tf** file and, in the `terraform` section, manually provide these values.	

Task	Description	Skills required
Deploy the resources.	1. In the root directory of the cloned repository, enter the following command. `terraform apply -var-file="dev.tfvars"` 2. Review the list of all changes that will be applied to your AWS account, and then confirm the deployment. 3. Wait until all resources are deployed.	DevOps engineer, Developer, Terraform

Set up the local environment

Task	Description	Skills required
Configure the SSH connection.	Update the SSH configuration file to allow SSH connections through Session Manager. For instructions, see [Allowing SSH connections for Session Manager](). This allows authorized users to enter a proxy command that starts a Session Manager session and transfers all data through a two-way connection.	DevOps engineer
Generate the SSH keys.	Enter the following command to generate a local private and public SSH key pair. You	DevOps engineer, Developer

Task	Description	Skills required
	use this key pair to connect to the bastion host. `ssh-keygen -t rsa -f my_key`	

Connect to the bastion host by using Session Manager

Task	Description	Skills required
Get the instance ID.	1. In order to connect to the deployed bastion host, you need the ID of the Amazon EC2 instance. Do one of the following to locate the ID: • Open the [Amazon EC2 console](). In the navigation pane, choose **Instances** . Locate the bastion host instance. • In the AWS CLI, enter the following command. `aws ec2 describe-instances` To filter the results, enter the following command, where `$BASTION_HOST_TAG` is the tag you assigned to the bastion host. The default value for this tag is	General AWS

Task	Description	Skills required

	sandbox-dev-bastion-host .
	
	`aws ec2 describe-instances \`
	`    --filters "Name=tag:Name,Values=$BASTION_HOST_TAG" \`
	`    --output text \`
	`    --query 'Reservations[*].Instances[*].InstanceId' \`
	`    --output text`
	
	2. Copy the ID of the Amazon EC2 instance. You use this ID later.	

Task	Description	Skills required
Send the SSH public key.	**Note** In this section, you upload the public key to the instance metadata of the bastion host. After the key is uploaded, you have 60 seconds to start a connection with the bastion host. After 60 seconds, the public key is removed. For more informati on, see the Troublesh ooting section of this pattern. Complete the next steps quickly to prevent the key from being removed before you connect to the bastion host. 1. Send the SSH key to the bastion host by using Amazon EC2 Instance Connect. Enter the following command, where: • `$INSTANCE_ID` is the ID of the Amazon EC2 instance	General AWS

Task	Description	Skills required
	• `$PUBLIC_KEY_FILE` is the path to your public key file, such as `my_key.pub` **Important** Be sure to use the public key and not the private key. `aws ec2-instance-connect send-ssh-public-key \ --instance-id $INSTANCE_ID \ --instance-os-user ec2-user \ --ssh-public-key file://$PUBLIC_KEY_FILE` 2. Wait until you receive a message indicating that the key has been successfully uploaded. Continue to the next step immediately.	

Task	Description	Skills required
Connect to the bastion host.	1. Enter the following command to connect to the bastion host through Session Manager, where: • `$PRIVATE_KEY_FILE` is the path to your private key, such as `my_key` • `$INSTANCE_ID` is the ID of the Amazon EC2 instance `ssh -i $PRIVATE_KEY_FILE ec2-user@$INSTANCE_ID` 2. Confirm the connection by entering `yes`. This opens an SSH connection by using Session Manager. **Note** There are other options for opening an SSH connection with the bastion host. For more information, see *Alternative approaches to establish an SSH connection with the bastion host* in the [Additional informati	General AWS

Task	Description	Skills required
	on section of this pattern.	

(Optional) Clean up

Task	Description	Skills required
Remove the deployed resources.	1. In order to remove all deployed resources, run the following command from the root directory of the cloned repository. `terraform destroy -var-file="dev.tfvars"` 2. Confirm the removal of the resources.	DevOps engineer, Developer, Terraform

Troubleshooting

Issue	Solution
`TargetNotConnected` error when trying to connect to the bastion host	1. Reboot the bastion host according to the instructions in Reboot your instance in the Amazon EC2 documentation. 2. After the instance has successfully rebooted, resend the public key to the bastion host and reattempt the connection.
`Permission denied` error when trying to connect to the bastion host	After the public key is uploaded to the bastion host, you have only 60 seconds to start the connection. After 60 seconds, the key is

Issue	Solution
	automatically removed, and you can't use it to connect to the instance. If this occurs, you can repeat the step to resend the key to the instance.

Related resources

AWS documentation

- AWS Systems Manager Session Manager (Systems Manager documentation)
- Install the Session Manager plugin for the AWS CLI (Systems Manager documentation)
- Allowing SSH connections for Session Manager (Systems Manager documentation)
- About using EC2 Instance Connect (Amazon EC2 documentation)
- Connect using EC2 Instance Connect (Amazon EC2 documentation)
- Identity and access management for Amazon EC2 (Amazon EC2 documentation)
- Using an IAM role to grant permissions to applications running on Amazon EC2 instances (IAM documentation)
- Security best practices in IAM (IAM documentation)
- Control traffic to resources using security groups (Amazon VPC documentation)

Other resources

- Terraform Developer webpage
- Command: validate (Terraform documentation)
- Command: fmt (Terraform documentation)
- Testing HashiCorp Terraform (HashiCorp blog post)
- Checkov webpage

Additional information

Alternative approaches to establish an SSH connection with the bastion host

Port forwarding

You can use the `-D 8888` option to open an SSH connection with dynamic port forwarding. For more information, see the [instructions](#) at explainshell.com. The following is an example of a command to open an SSH connection by using port forwarding.

```
ssh -i $PRIVATE_KEY_FILE -D 8888 ec2-user@$INSTANCE_ID
```

This is kind of connection opens a SOCKS proxy that can forward traffic from your local browser through the bastion host. If you are using Linux or MacOS, to see all options, enter `man ssh`. This displays the SSH reference manual.

Using the provided script

Instead of manually running the steps described in *Connect to the bastion host by using Session Manager* in the Epics section, you can use the **connect.sh** script included in the code repository. This script generates the SSH key pair, pushes the public key to the Amazon EC2 instance, and initiates a connection with the bastion host. When you run the script, you pass the tag and key name as arguments. The following is an example of the command to run the script.

```
./connect.sh sandbox-dev-bastion-host my_key
```

Centralize DNS resolution by using AWS Managed Microsoft AD and on-premises Microsoft Active Directory

Brian Westmoreland, Amazon Web Services

Summary

This pattern provides guidance for centralizing DNS resolution within an AWS multi-account environment by using both AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) and Amazon Route 53. In this pattern the AWS DNS namespace is a subdomain of the on-premises DNS namespace. This pattern also provides guidance on how to configure the on-premises DNS servers to forward queries to AWS when the on-premises DNS solution uses Microsoft Active Directory.

Prerequisites and limitations

Prerequisites

- An AWS multi-account environment set up by using AWS Organizations.
- Network connectivity established between AWS accounts.
- Network connectivity established between AWS and the on-premises environment (by using AWS Direct Connect or any type of VPN connection).
- AWS Command Line Interface (AWS CLI) configured on a local workstation.
- AWS Resource Access Manager (AWS RAM) used to share Route 53 rules between accounts. Therefore, sharing must be enabled within the AWS Organizations environment, as described in the Epics section.

Limitations

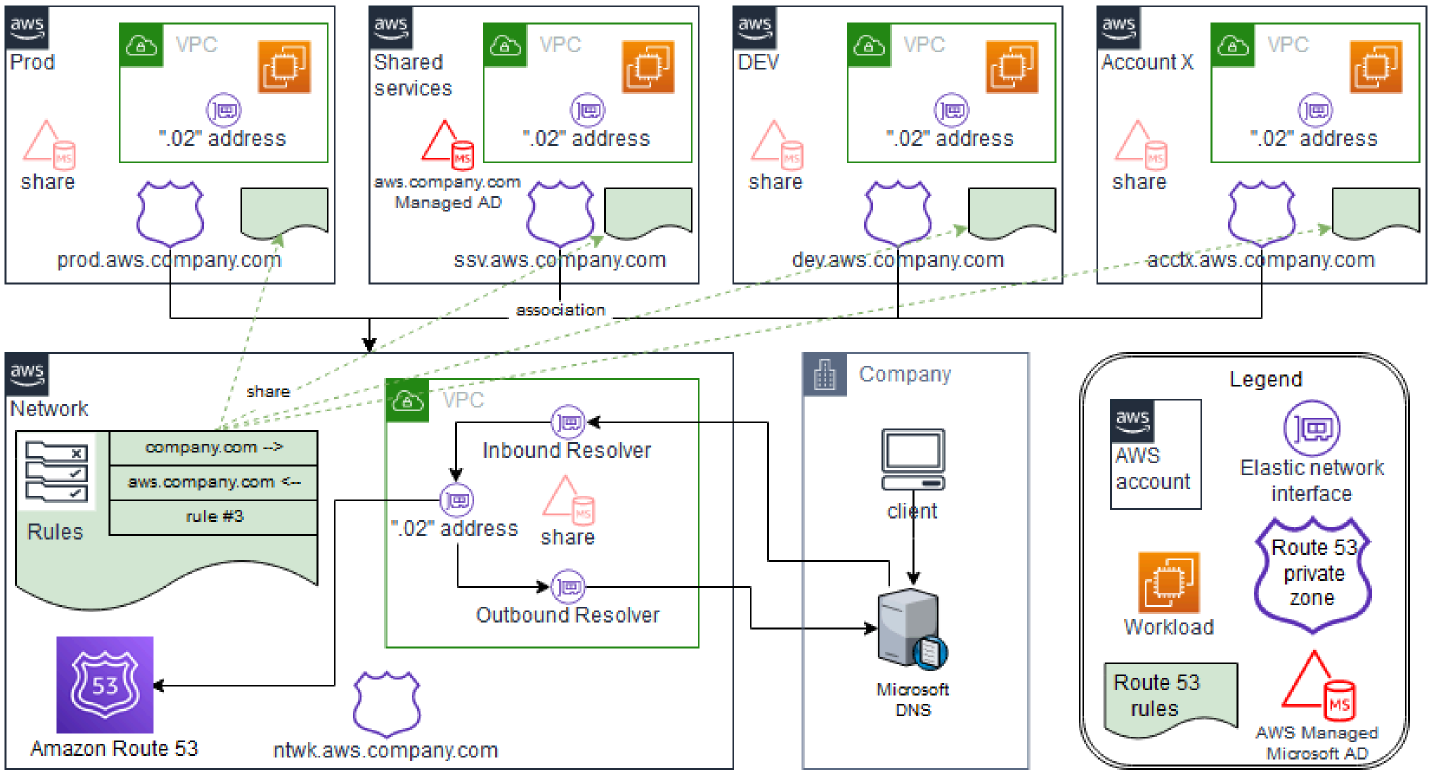
- AWS Managed Microsoft AD Standard Edition has a limit of 5 shares.
- AWS Managed Microsoft AD Enterprise Edition has a limit of 125 shares.
- The solution in this pattern is limited to AWS Regions that support sharing through AWS RAM.

Product versions

- Microsoft Active Directory running on Windows Server 2008, 2012, 2012 R2, or 2016.

Architecture

Target architecture



In this design, AWS Managed Microsoft AD is installed in the shared services AWS account. Although it is not a requirement, this pattern assumes this configuration. If you configure AWS Managed Microsoft AD in a different AWS account, you might have to modify the steps in the Epics section accordingly.

This design uses Route 53 Resolvers to support name resolution through the use of Route 53 rules. If the on-premises DNS solution uses Microsoft DNS, creating a conditional forwarding rule for the AWS namespace (`aws.company.com`), which is a subdomain of the company DNS namespace (`company.com`), is not straightforward. If you try to create a traditional conditional forwarder, it will result in an error. This is because Microsoft Active Directory is already considered authoritative for any subdomain of `company.com`. To get around this error, you must first create a delegation for `aws.company.com` to delegate authority of that namespace. You can then create the conditional forwarder.

The virtual private cloud (VPC) for each spoke account can have its own unique DNS namespace based on the root AWS namespace. In this design, each spoke account appends an abbreviation of the account name to the base AWS namespace. After the private hosted zones in the spoke account have been created, the zones are associated with the local VPC in the spoke account as well as with the VPC in the central AWS network account. This enables the central AWS network account to answer DNS queries related to the spoke accounts. This way, both Route 53 and AWS Managed Microsoft AD work together to share the responsibility of managing the AWS namespace (`aws.company.com`).

Automation and scale

This design uses Route 53 Resolver endpoints to scale DNS queries between AWS and your on-premises environment. Each Route 53 Resolver endpoint comprises multiple elastic network interfaces (spread across multiple Availability Zones), and each network interface can handle up to 10,000 queries per second. Route 53 Resolver supports up to 6 IP addresses per endpoint, so altogether this design supports up to 60,000 DNS queries per second spread across multiple Availability Zones for high availability.

Additionally, this pattern automatically accounts for future growth within AWS. The DNS forwarding rules configured on premises do not have to be modified to support new VPCs and their associated private hosted zones that are added to AWS.

Tools

AWS services

- AWS Directory Service for Microsoft Active Directory enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Resource Access Manager (AWS RAM) helps you securely share your resources across AWS accounts to reduce operational overhead and provide visibility and auditability.
- Amazon Route 53 is a highly available and scalable DNS web service.

Tools

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell. In this pattern, the AWS CLI is used to configure Route 53 authorizations.

Epics

Create and share an AWS Managed Microsoft AD directory

Task	Description	Skills required
Deploy AWS Managed Microsoft AD.	1. Create and configure a new directory. For detailed steps, see [Creating your AWS Managed Microsoft AD](#) in the *AWS Directory Service Administration Guide*. 2. Record the IP addresses of the AWS Managed Microsoft AD domain controllers. These will be referenced in a later step.	AWS administrator
Share the directory.	After the directory has been built, share it with other AWS accounts in the AWS organization. For instructions, see [Share your directory](#) in the *AWS Directory Service Administration Guide*. **Note** AWS Managed Microsoft AD Standard Edition has a limit of 5 shares. Enterprise Edition has a limit of 125 shares.	AWS administrator

Configure Route 53

Task	Description	Skills required
Create Route 53 Resolvers.	Route 53 Resolvers facilitate DNS query resolution between AWS and the on-premises data center. 1. Install Route 53 Resolvers by following the instructions in the *Route 53 Developer Guide*. 2. Configure Route 53 Resolvers in private subnets in at least two Availability Zones within the central AWS network account VPC for high availability. **Note** Although using the central AWS network account VPC isn't a requirement, the remaining steps assume this configuration.	AWS administrator
Create Route 53 rules.	Your specific use case might require a large number of Route 53 rules, but you will need to configure the following rules as a baseline:	AWS administrator

Task	Description	Skills required

	• An outgoing rule for the on-premises namespace (`company.com`) by using the central network account outbound Route 53 Resolvers. The target IP addresses are the on-premises DNS servers.
	  ◦ Associate this rule with the central network account VPC.
	• An outgoing rule for the AWS namespace (`aws.company.com`) by using the central network account outbound Route 53 Resolvers. The target IP addresses are the central network account inbound Route 53 Resolver IP addresses.
	  ◦ **Do not** associate this rule with the central AWS network account VPC (which houses the Route 53 Resolvers).
	• A second outgoing rule for the AWS namespace (`aws.company.com`) that points to the AWS Managed Microsoft AD domain controllers (use the IPs from the previous epic).	

Task	Description	Skills required
	• Associate this rule with the central AWS network account VPC (which houses the Route 53 Resolvers). • **Do not** share or associate this rule with other AWS accounts. For more information, see [Managing forwarding rules]() in the *Route 53 Developer Guide*.	

Task	Description	Skills required
Configure a Route 53 Profile.	A Route 53 Profile is used to share the rules with spoke accounts. 1. Create a new Route 53 Profile in the central networking account by following the [instructions](#) in the *Route 53 Developer Guide*. 2. Add the rule for the on-premises namespace (`company.com`) to the Profile. 3. Add the first rule for the AWS namespace (`aws.company.com`), which targets the IP addresses of the Route 53 inbound resolvers, to the Profile. 4. Share the Route 53 Profile with the AWS organization. 5. Accept the Route 53 Profile resource share in each spoke account. 6. Associate the Route 53 Profile with each spoke account VPC.	AWS administrator

Configure on-premises Active Directory DNS

Task	Description	Skills required
Create the delegation.	Use the Microsoft DNS snap-in (`dnsmgmt.msc`) to create a new delegation for the `company.com` namespace within Active Directory. The name of the delegated domain should be `aws`. This makes the fully qualified domain name (FQDN) of the delegation `aws.company.com` . Use the IP addresses of the AWS Managed Microsoft AD domain controllers for the name server IP values, and use `server.aws.company.com` for the name. (This delegation is only for redundancy, because a conditional forwarder will be created for this namespace that takes precedence over the delegation.)	Active Directory
Create the conditional forwarder.	Use the Microsoft DNS snap-in (`dnsmgmt.msc`) to create a new conditional forwarder for `aws.company.com` . Use the IP addresses of the AWS inbound Route 53 Resolvers in the central DNS	Active Directory

Task	Description	Skills required
	AWS account for the target of the conditional forwarder.	

Create Route 53 private hosted zones for spoke AWS accounts

Task	Description	Skills required
Create the Route 53 private hosted zones.	Create a Route 53 private hosted zone in each spoke account. Associate this private hosted zone with the spoke account VPC. For detailed steps, see [Creating a private hosted zone](#) in the *Route 53 Developer Guide*.	AWS administrator
Create authorizations.	Use the AWS CLI to create an authorization for the central AWS network account VPC. Run this command from the context of each spoke AWS account: `aws route53 create-vpc-association-authorization --hosted-zone-id <hosted-zone-id> \ --vpc VPCRegion=<region>,VPCId=<vpc-id>` where:	AWS administrator

Task	Description	Skills required
	• `<hosted-zone-id>` is the Route 53 private hosted zone in the spoke account. • `<region>` and `<vpc-id>` are the AWS Region and VPC ID of the central AWS network account VPC.	
Create associations.	Create the Route 53 private hosted zone association for the central AWS network account VPC by using the AWS CLI. Run this command from the context of the central AWS network account: `aws route53 associate-vpc-with-hosted-zone --hosted-zone-id <hosted-zone-id> \ --vpc VPCRegion=<region>,VPCId=<vpc-id>` where: • `<hosted-zone-id>` is the Route 53 private hosted zone in the spoke account. • `<region>` and `<vpc-id>` are the AWS Region and VPC ID of the central AWS network account.	AWS administrator

Related resources

- Simplify DNS management in a multi-account environment with Route 53 Resolver (AWS blog post)
- Creating your AWS Managed Microsoft AD (AWS Directory Service documentation)
- Sharing an AWS Managed Microsoft AD directory (AWS Directory Service documentation)
- What is Amazon Route 53 Resolver? (Amazon Route 53 documentation)
- Creating a private hosted zone (Amazon Route 53 documentation)
- What are Amazon Route 53 Profiles? (Amazon Route 53 documentation)

Centralize monitoring by using Amazon CloudWatch Observability Access Manager

Anand Krishna Varanasi, JAGDISH KOMAKULA, Ashish Kumar, Jimmy Morgan, Sarat Chandra Pothula, Vivek Thangamuthu, and Balaji Vedagiri, Amazon Web Services

Summary

Observability is crucial to monitoring, understanding, and troubleshooting applications. Applications that span multiple accounts, as with AWS Control Tower or landing zone implementations, generate a large number of logs and trace data. To quickly troubleshoot problems or understand user analytics or business analytics, you need a common observability platform across all accounts. The Amazon CloudWatch Observability Access Manager gives you access to, and control over, multiple account logs from a central location.

You can use the Observability Access Manager to view and manage observability data logs generated by source accounts. Source accounts are individual AWS accounts that generate observability data for their resources. Observability data is shared between source accounts and monitoring accounts. The shared observability data can include metrics in Amazon CloudWatch, logs in Amazon CloudWatch Logs, and traces in AWS X-Ray. For more information, see the Observability Access Manager documentation.

This pattern is for users who have applications or infrastructure that run in multiple AWS accounts and need a common place to view logs. It explains how you can set up Observability Access Manager by using Terraform, to monitor the status and health of these applications or infrastructure. You can install this solution in multiple ways:

- As a standalone Terraform module that you set up manually
- By using a continuous integration and continuous delivery (CI/CD) pipeline
- By integrating with other solutions such as AWS Control Tower Account Factory for Terraform (AFT)

The instructions in the Epics section cover the manual implementation. For AFT installation steps, see the README file for the GitHub Observability Access Manager repository.

Prerequisites and limitations

Prerequisites

- Terraform installed or referenced in your system or in automated pipelines. (We recommend that you use the latest version.)
- An account that you can use as a central monitoring account. Other accounts create links to the central monitoring account in order to view logs.
- (Optional) A source code repository such as GitHub, AWS CodeCommit, Atlassian Bitbucket, or similar system. A source code repository isn't necessary if you're using automated CI/CD pipelines.
- (Optional) Permissions to create pull requests (PRs) for code review and code collaboration in GitHub.

Limitations

Observability Access Manager has the following service quotas, which cannot be changed. Consider these quotas before you deploy this feature. For more information, see CloudWatch service quotas in the CloudWatch documentation.

- **Source account links**: You can link each source account to a maximum of five monitoring accounts.
- **Sinks**: You can build multiple sinks for an account, but only one sink per AWS Region is allowed.

In addition:

- Sinks and links must be created in the same AWS Region; they cannot be cross-Region.

Cross-Region and cross-account monitoring

For cross-Region, cross-account monitoring, you can choose one of these options:

- Create cross-account and cross-Region CloudWatch dashboards for alarms and metrics. This option doesn't support logs and traces.
- Implement centralized logging by using Amazon OpenSearch Service.
- Create one sink per Region from all tenant accounts, push metrics to a centralized monitoring account (as described in this pattern), and then use CloudWatch metric streams to send the data to a common external destination or to third-party monitoring products such as Datadog, Dynatrace, Sumo Logic, Splunk, or New Relic.

Architecture

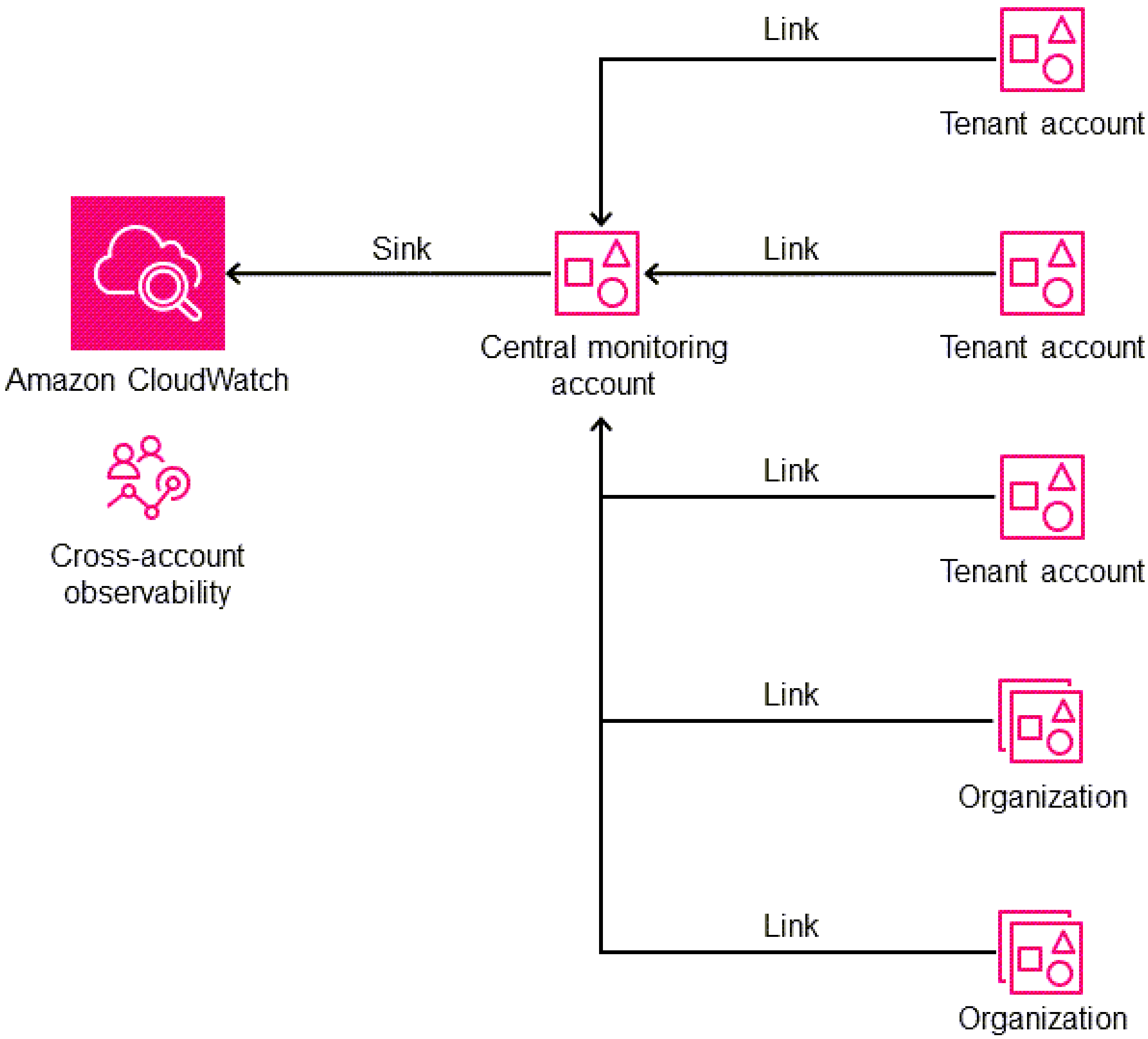
Components

CloudWatch Observability Access Manager consists of two major components that enable cross-account observability:

- A *sink* provides the ability for source accounts to send observability data to the central monitoring account. A sink basically provides a gateway junction for source accounts to connect to. There can be only one sink gateway or connection, and multiple accounts can connect to it.
- Each source account has a *link* to the sink gateway junction, and observability data is sent through this link. You must create a sink before you create links from each source account.

Architecture

The following diagram illustrates Observability Access Manager and its components.



Tools

AWS services

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.
- AWS Control Tower Account Factory for Terraform (AFT) sets up a Terraform pipeline to help you provision and customize accounts in AWS Control Tower. You can optionally use AFT to set up Observability Access Manager at scale across multiple accounts.

Code repository

The code for this pattern is available in the GitHub Observability Access Manager repository.

Best practices

- In AWS Control Tower environments, mark the logging account as the central monitoring account (sink).
- If you have multiple organizations with multiple accounts in AWS Organizations, we recommend that you include the organizations instead of individual accounts in the configuration policy. If you have a small number of accounts or if the accounts aren't part of an organization in the sink configuration policy, you might decide to include individual accounts instead.

Epics

Set up the sink module

Task	Description	Skills required
Clone the repository.	Clone the GitHub Observability Access Manager repository: `git clone https://github.com/aws-samples/cloudwatch-obervability-access-manager-terraform`	AWS DevOps, Cloud administrator, AWS administrator
Specify property values for the sink module.	In the `main.tf` file (in the `deployments/aft-ac`	AWS DevOps, Cloud administrator, AWS administrator

Task	Description	Skills required
	`count-customizations/LOGGING/terraform/` folder of the repository), specify values for the following properties: • `sink_name` : The name of the CloudWatch sink. • `allowed_oam_resource_types` : Observability Access Manager currently supports CloudWatch metrics, log groups, and AWS X-Ray traces. • `allowed_source_accounts` : The source accounts that are allowed to send logs to the central CloudWatch sink account. • `allowed_source_organizations` : The source AWS Control Tower organizations that are allowed to send logs to the central CloudWatch sink account. For more information, see AWS::Oam::Sink in the AWS CloudFormation documentation.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install the sink module.</td><td>Export the credentials of the AWS account that you have selected as the monitorin g account, and install the Observability Access Manager sink module:
<pre><code>Terraform Init
Terrafom Plan
Terraform Apply</code></pre></td><td>AWS DevOps, Cloud administr ator, AWS administrator</td></tr>
</table>

Set up the link module

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Specify property values for the link module.</td><td>In the <code>main.tf</code> file (in the <code>deployments/aft-ac count-customizatio ns/LOGGING/terrafo rm/</code> folder of the repositor y), specify values for the following properties:
<code>account_label</code> : Use one of the following values:
<code>$AccountName</code> : The name of the account.
<code>$AccountEmail</code> : A globally unique email address, which includes the email domain (for example, <code>hello@exa mple.com</code>)</td><td>AWS DevOps, Cloud administr ator, Cloud architect</td></tr>
</table>

Task	Description	Skills required
	• `$AccountEmailNoDomain` : An email address without the domain name. • `allowed_oam_resource_types` : Observability Access Manager currently supports CloudWatch metrics, log groups, and AWS X-Ray traces. For more information, see AWS::Oam::Link in the AWS CloudFormation documentation.	

Install the link module for individual accounts.	Export the credentials of individual accounts and install the Observability Access Manager link module:
	`Terraform Plan`
	`Terraform Apply`
	You can set up the link module individually for each account, or use AFT to automatically install this module across a large number of accounts.
		AWS DevOps, Cloud administrator, Cloud architect

Approve sink-to-link connections

Task	Description	Skills required
Check the status message.	1. Sign in to the monitoring account. 2. Open the [CloudWatch console](https://console.aws.amazon.com/cloudwatch/). 3. In the left navigation pane, choose **Settings**. On the right, you should see the status message **Monitoring account enabled** with a green checkmark. This means that the monitoring account has an Observability Access Manager sink that the links of other accounts will connect to.	
Approve the link-to-sink connections.	1. Choose the **Resources to link accounts** option below the status message. The information confirms that this is the monitoring account, lists the data that is shared from the tenant source accounts (**Logs**, **Metrics, Traces**), and shows the account label as **$AccountName.** This screen provides two options for linking tenant accounts to the monitoring account: organization-	AWS DevOps, Cloud administrator, Cloud architect

Task	Description	Skills required
	level approval or account-level approval. For each option, you can choose to download an AWS CloudFormation template for the approval or approve each account individually. 2. For simplicity, choose **Any Account** to approve at each account level. This option provides an approval link for the account. 3. Choose **Copy URL** to copy the link. 4. Sign in to each source account. 5. In a browser window, paste the link, and choose **Approve link connect to sink**. 6. Repeat for additional source accounts. For more information, see Link monitoring accounts with source accounts in the CloudWatch documentation.	

Verify cross-account observability data

Task	Description	Skills required
View cross-account data.	1. Sign in to the central monitoring account. 2. Open the [CloudWatch console](#). 3. In the left navigation pane, choose options to view cross-account logs, metrics, and traces.	AWS DevOps, Cloud administr ator, Cloud architect

(Optional) Enable source accounts to trust monitoring account

Task	Description	Skills required
View metrics, dashboards, logs, widgets, and alarms from other accounts.	As an additional feature, you can share the CloudWatc h metrics, dashboards, logs, widgets, and alarms with other accounts. Each account uses an IAM role called **CloudWatch-CrossAc countSharingRole** to gain access to this data. Source accounts that have a trust relationship with the central monitoring account can assume this role and view data from the monitoring account. CloudWatch provides a sample CloudFormation script	AWS DevOps, Cloud administr ator, Cloud architect

Task	Description	Skills required
	to create the role. Choose **Manage role in IAM and** run this script in the accounts where you want to view data. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": [ "arn:aws:iam::XXXXXXXXX:root", "arn:aws:iam::XXXXXXXXX:root", "arn:aws:iam::XXXXXXXXX:root", "arn:aws:iam::XXXXXXXXX:root" ] }, "Action": "sts:AssumeRole" } ] }` For more information, see [Enabling cross-account functionality in CloudWatc]()	

Task	Description	Skills required
	h in the CloudWatch documentation.	

(Optional) View cross-account cross-Region from the monitoring account

Task	Description	Skills required
Set up cross-account, cross-Region access.	In the central monitoring account, you can optionally add an account selector to easily switch between accounts and view their data without having to authenticate. 1. Sign in to the central monitoring account. 2. Open the CloudWatch console. 3. In the left navigation pane, choose **Settings**. 4. In the **View cross-account cross-region** section, choose **Configure**. 5. Choose **Enable**, and then select the **Show selector in the console** check box. 6. Choose one of these options: • **Account Id Input:** This option prompts you to manually input the account ID whenever you	AWS DevOps, Cloud administrator, Cloud architect

Task	Description	Skills required
	want to change accounts to view cross-account data. • **AWS Organization account selector:** If you have integrated CloudWatch with AWS Organizations, this option provides a dropdown selector with a complete list of accounts in the organization. • **Custom account selector:** This option lets you manually input a list of account IDs to populate the selector. 7. Choose **Save changes.** For more information, see Cross-account cross-Region CloudWatch console in the CloudWatch documentation.	

Related resources

- CloudWatch cross-account observability (Amazon CloudWatch documentation)
- Amazon CloudWatch Observability Access Manager API Reference (Amazon CloudWatch documentation)
- Resource: aws_oam_sink (Terraform documentation)
- Data Source: aws_oam_link (Terraform documentation)
- CloudWatchObservabilityAccessManager (AWS Boto3 documentation)

Check EC2 instances for mandatory tags at launch

Susanne Kangnoh and Archit Mathur, Amazon Web Services

Summary

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

You can use tagging to categorize your AWS resources in different ways. EC2 instance tagging is useful when you have many resources in your account and you want to quickly identify a specific resource based on the tags. You can assign custom metadata to your EC2 instances by using tags. A tag consists of a user-defined key and value. We recommend that you create a consistent set of tags to meet your organization's requirements.

This pattern provides an AWS CloudFormation template to help you monitor EC2 instances for specific tags. The template creates an Amazon CloudWatch Events event that watches for the AWS CloudTrail **TagResource** or **UntagResource** events, to detect new EC2 instance tagging or tag removal. If a predefined tag is missing, it calls an AWS Lambda function, which sends out a violation message to an email address that you provide, by using Amazon Simple Notification Service (Amazon SNS).

Prerequisites and limitations

Prerequisites

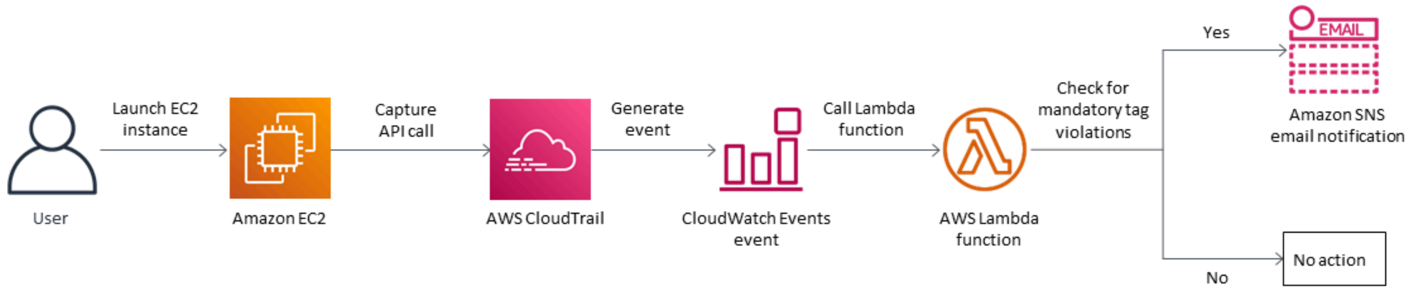
- An active AWS account.
- An Amazon Simple Storage Service (Amazon S3) bucket to upload the provided Lambda code.
- An email address where you would like to receive violation notifications.

Limitations

- This solution supports CloudTrail **TagResource** or **UntagResource** events. It does not create notifications for any other events.
- This solution checks only for tag keys. It does not monitor key values.

Architecture

Workflow architecture



Automation and scale

- You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run the template only once in each Region or account.

Tools

AWS services

- Amazon EC2 – Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.
- AWS CloudTrail – CloudTrail is an AWS service that helps you with governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or AWS service are recorded as events in CloudTrail.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- AWS Lambda – Lambda is a compute service that supports running code without needing to provision or manage servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

- [Amazon S3](#) – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- [Amazon SNS](#) – Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end-users, and devices to instantly send and receive notifications from the cloud.

Code

This pattern includes an attachment with two files:

- `index.zip` is a compressed file that includes the Lambda code for this pattern.
- `ec2-require-tags.yaml` is a CloudFormation template that deploys the Lambda code.

See the *Epics* section for information about how to use these files.

Epics

Deploy the Lambda code

Task	Description	Skills required
Upload the code to an S3 bucket.	Create a new S3 bucket or use an existing S3 bucket to upload the attached `index.zip` file (Lambda code). This bucket must be in the same AWS Region as the resources (EC2 instances) that you want to monitor.	Cloud architect
Deploy the CloudFormation template.	Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the `ec2-require-tags.yaml` file that's provided in the attachment. In the next epic, provide	Cloud architect

Task	Description	Skills required
	values for the template parameters.	

Complete the parameters in the CloudFormation template

Task	Description	Skills required
Provide the S3 bucket name.	Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudForm ation template and the EC2 instances that you want to monitor.	Cloud architect
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `index.zip` or `controls/index.zip`).	Cloud architect
Provide an email address.	Provide an active email address where you want to receive violation notifications.	Cloud architect
Define a logging level.	Specify the logging level and verbosity. `Info` designate s detailed informational messages on the applicati on's progress and should be used only for debugging	Cloud architect

Task	Description	Skills required
	. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situation s.	
Enter the required tag keys.	Enter the tag keys that you want to check for. If you want to specify multiple keys, separate them with commas, without spaces. (For example, `ApplicationId,Crea tedBy,Environment, Organization` searches for four keys.) The CloudWatc h Events event searches for these tag keys and sends a notification if they are not found.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the email subscript ion.	When the CloudFormation template deploys successfu lly, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.	Cloud architect

Related resources

- Creating a bucket (Amazon S3 documentation)
- Uploading objects (Amazon S3 documentation)
- Tag your Amazon EC2 resources (Amazon EC2 documentation)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail (Amazon CloudWatch documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Clean up AWS Account Factory for Terraform (AFT) resources safely after state file loss

Gokendra Malviya, Amazon Web Services

Summary

When you use AWS Account Factory for Terraform (AFT) to manage your AWS Control Tower environment, AFT generates a Terraform state file to track the state and configuration of the resources created by Terraform. Losing the Terraform state file can create significant challenges for resource management and cleanup. This pattern provides a systematic approach to safely identify and remove AFT-related resources while maintaining the integrity of your AWS Control Tower environment.

The process is designed to ensure proper removal of all AFT components, even without the original state file reference. This process provides a clear path to successfully re-establish and reconfigure AFT in your environment, to help ensure minimal disruption to your AWS Control Tower operations.

For more information about AFT, see the [AWS Control Tower documentation](#).

Prerequisites and limitations

Prerequisites

- A thorough understanding of [AFT architecture](#).
- Administrator access to the following accounts:
  - AFT Management account
  - AWS Control Tower Management account
  - Log Archive account
  - Audit account
- Verification that no service control policies (SCPs) contain restrictions or limitations that would block the deletion of AFT-related resources.

Limitations

- This process can clean up resources effectively, but it cannot recover lost state files, and some resources might require manual identification.

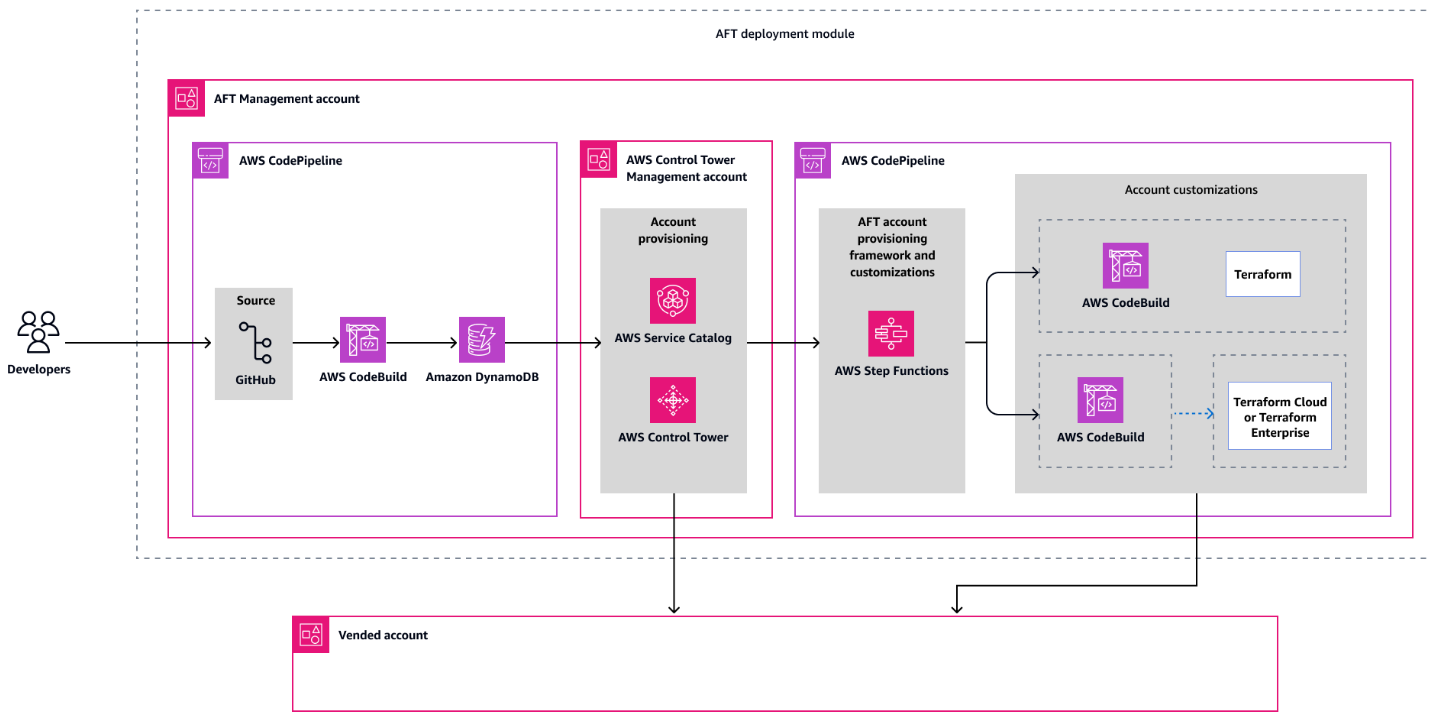
- The duration of the cleanup process depends on your environment's complexity and might take several hours.
- This pattern has been tested with AFT version 1.12.2 and deletes the following resources. If you're using a different version of AFT, you might have to delete additional resources.

Service name	Number of resources
AWS CodeBuild	6
AWS CodeCommit	4
AWS CodePipeline	4
Amazon DynamoDB	5
Amazon Elastic Compute Cloud (Amazon EC2)	16
Amazon EventBridge	4
AWS Identity and Access Management (IAM) roles	40
AWS Key Management Service (AWS KMS)	2
AWS Lambda	17
Amazon Simple Storage Service (Amazon S3)	2
Amazon Simple Notification Service (Amazon SNS)	2
Amazon Simple Queue Service (Amazon SQS)	2
AWS Systems Manager	62
AWS Step Functions	4

> **Important**
>
> The resources that are deleted by the steps in this pattern cannot be recovered. Before you follow these steps, verify the resource names carefully and make sure that they were created by AFT.

Architecture

The following diagram shows the AFT components and high-level workflow. AFT sets up a Terraform pipeline that helps you provision and customize your accounts in AWS Control Tower. AFT follows a GitOps model to automate the processes of account provisioning in AWS Control Tower. You create a Terraform file for an account request and commit it to a repository, which provides the input that triggers the AFT workflow for account provisioning. After account provisioning is complete, AFT can run additional customization steps automatically.



In this architecture:

- **AWS Control Tower Management account** is an AWS account that's dedicated to the AWS Control Tower service. This is also typically referred to as the *AWS payer account* or *AWS Organizations Management account*.

- **AFT Management account** is an AWS account that's dedicated to AFT management operations. This is different from your organization's management account.
- **Vended account** is an AWS account that contains all the baseline components and controls that you selected. AFT uses AWS Control Tower to vend a new account.

For additional information about this architecture, see Introduction to AFT in the AWS Control Tower workshop.

Tools

AWS services

- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices.
- AWS Account Factory for Terraform (AFT) sets up a Terraform pipeline to help you provision and customize accounts and resources in AWS Control Tower.
- AWS Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources. Using Organizations, you can create accounts and allocate resources, group accounts to organize your workflows, apply policies for governance, and simplify billing by using a single payment method for all your accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them. This pattern requires IAM roles and permissions.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Best practices

- For AWS Control Tower, see Best practices for AWS Control Tower administrators in the AWS Control Tower documentation.
- For IAM, see Security best practices in the IAM documentation.

Epics

Delete AFT resources in the AFT Management account

Task	Description	Skills required
Delete resources that are identified by the AFT tag.	1. Sign in to the AFT management account with administrator permissions. 2. Open the [AWS Resource Groups console](https://console.aws.amazon.com/resource-groups/). 3. Select the Region where AWS Control Tower has been deployed. 4. In the navigation pane, choose **Tag Editor**. 5. For **Resource types**, choose **All supported resource types**. 6. For **Tags**, type **managed_b y** as the tag key and **AFT** as the tag value. 7. Choose **Search resources**. This search displays all resources that were created by AFT. 8. Identify the resource names and delete them by using the correspon ding service consoles. For example, to delete Parameter Store resources: a. Open the [AWS Systems Manager console](https://console.aws.amazon.com/systems-manager/).	AWS administrator, AWS DevOps, DevOps engineer

Task	Description	Skills required
	b. In the navigation pane, choose **Parameter Store**. c. In the search box, click to display the dropdown, choose **Name**, choose **equals**, and then type **/aft**. d. Delete the parameters in batches of 10. (This is the maximum number you can delete at the same time.) For AFT version 1.12.2, there will be approximately 62 Parameter Store resources to delete. All parameter names will start with **/aft**. However, not all resources can be identified by AWS Resource Groups. In the following steps, you will find and delete the remaining resources.	

Task	Description	Skills required
Delete IAM roles.	1. Sign in to the AFT management account with administrator permissions. 2. Open the IAM console. 3. Delete these roles in the order listed (the order is important because of dependencies): • `aft-*` • `AWSAFTAdmin` • `AWSAFTExecution` • `AWSAFTService` • `codebuild_trigger_role`	AWS administrator, AWS DevOps, DevOps engineer
Delete the AWS Backup backup vault.	1. Open the AWS Backup console. 2. Locate the backup vault named `aws_backup_vault` . 3. Confirm that the vault doesn't contain any active backups. 4. Delete `aws_backup_vault` .	AWS administrator, AWS DevOps, DevOps engineer

Task	Description	Skills required
Delete Amazon CloudWatch resources.	1. Open the [CloudWatch console](). 2. Delete the following resources in the order listed: a. Event bus: Delete `aws_cloudwatch_event_bus` . b. Logs: Search for the prefix **AFT** and delete all related log groups. c. Query definitions: Delete the following queries: • `Customization Logs by Account ID` • `Customization Logs by Customization Request ID`	AWS administrator, AWS DevOps, DevOps engineer
Delete AWS KMS resources.	1. Switch to the secondary Region, which serves as the backend for state tracking of AFT's own state. 2. Open the [AWS KMS console](). 3. Delete the alias named **AFT**.	AWS administrator, AWS DevOps, DevOps engineer

Delete AFT resources in the Log Archive account

Task	Description	Skills required
Delete S3 buckets.	1. Sign in to the Log Archive account with administrator permissions. 2. Open the [Amazon S3 console](https://console.aws.amazon.com/s3/). 3. Empty the following buckets: • `aws-aft-logs-471112509802-us-east-1` • `aws-aft-s3-access-logs-471112509802-us-east-1` (Replace 111122223333 with your account ID.) 4. Delete the two buckets.	AWS administrator, AWS DevOps, DevOps engineer
Delete IAM roles.	1. Open the [IAM console](https://console.aws.amazon.com/iam/). 2. Verify that the following roles aren't being used by any active services: • `AWSAFTService` • `AWSAFTExecution` 3. Delete the two roles.	AWS administrator, AWS DevOps, DevOps engineer

Delete AFT resources in the Audit account

Task	Description	Skills required
Delete IAM roles.	1. Sign in to the Audit account with administrator permissions. 2. Open the [IAM console](https://console.aws.amazon.com/iam/). 3. Verify that the following roles aren't being used by any active services: • `AWSAFTService` • `AWSAFTExecution` 4. Delete the two roles.	AWS administrator, AWS DevOps, DevOps engineer

Delete AFT resources in the AWS Control Tower Management account

Task	Description	Skills required
Delete IAM roles.	1. Sign in to the AWS Control Tower Management account with administrator permissions. 2. Open the [IAM console](https://console.aws.amazon.com/iam/). 3. Verify that the following roles aren't being used by any active services: • `AWSAFTService` • `AWSAFTExecution` • `aft-control-tower-events-rule` 4. Delete the three roles.	AWS administrator, AWS DevOps, DevOps engineer

Task	Description	Skills required
Delete EventBridge rules.	1. Open the [Amazon EventBridge console](). 2. In the left navigation pane, choose **Rules**. 3. Find and select the rule named `aft-capture-ct-events` . 4. Choose **Delete** and confirm the deletion when prompted.	AWS administrator, AWS DevOps, DevOps engineer

Troubleshooting

Issue	Solution
Detaching the internet gateway was unsuccessful.	While you're deleting resources that are identified by the **AFT** tag, if you encounter this issue when you detach or delete the internet gateway, you first have to delete VPC endpoints: 1. Sign in to the AFT Management account, and then open the [Amazon VPC console](). 2. In the navigation pane, in the **Filter by VPC** list, choose the VPC named **aft-management-vpc**. 3. In the navigation pane, choose **Endpoints**. 4. Select the endpoints that are associated with the VPC **aft-management-vpc**. • Double-check the VPC ID column before deletion to avoid removing the wrong endpoints.

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td></td><td>Be careful to delete only the endpoints that are associated with the AFT VPC.
<ol start="5">
Choose Actions, Delete VPC endpoints.
In the confirmation dialog box, type delete and then choose Delete.
Wait for the endpoint status to change to Deleted.

Deletion might take a few minutes to complete.</td></tr>
<tr><td>You're unable to find the specified CloudWatc h queries.</td><td>If you are unable to find the CloudWatch queries that were created by AFT, follow these steps:
Sign in to the AFT Management account, and then open the CloudWatch console.
In the navigation pane, under Logs, choose Logs Insights.
In the upper-right corner, choose the Saved and sample queries icon.

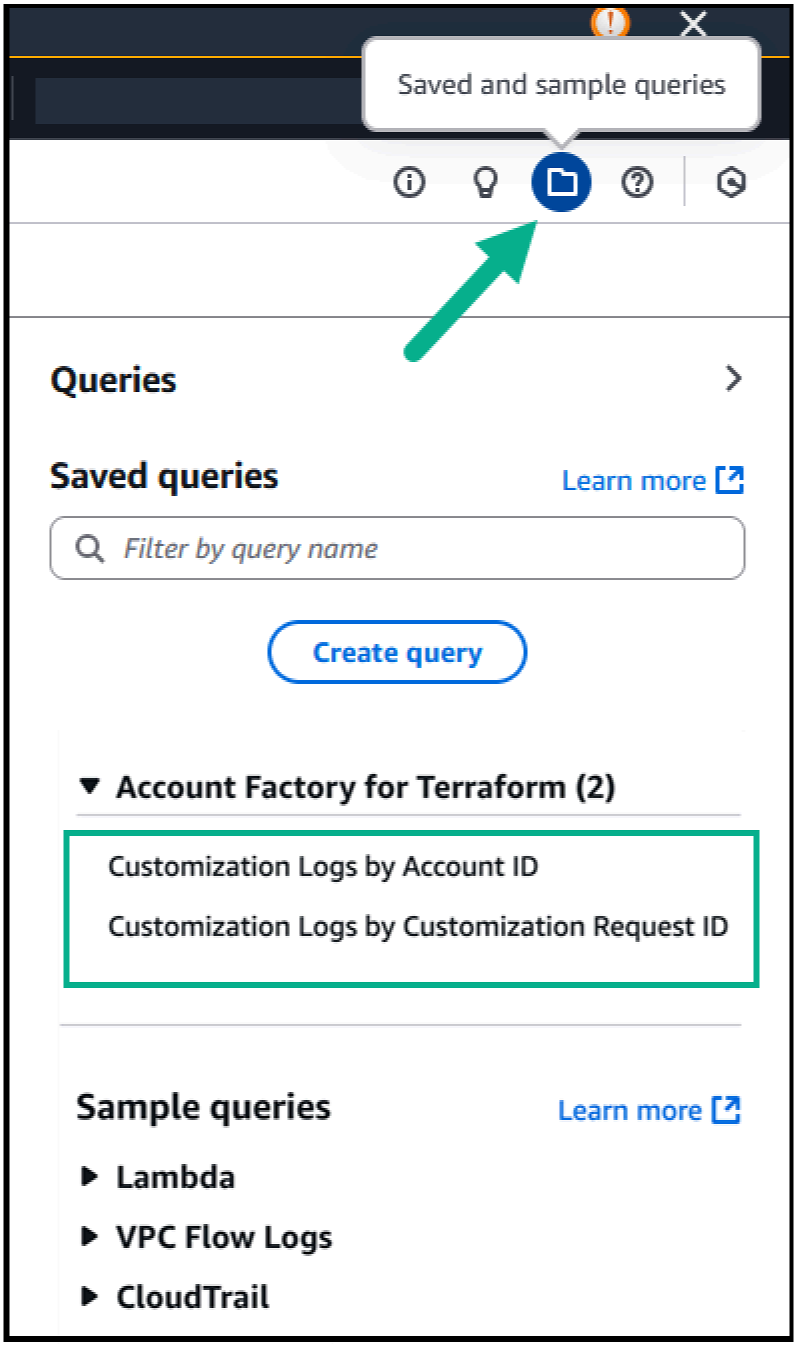
You should now be able to see AFT queries. For a screenshot, see the Additional information section.
Select the following queries, and then choose Actions, Delete to remove them.
<code>Customization Logs by Account ID</code>
<code>Customization Logs by Customization Request ID</code></td></tr>
</table>

Related resources

- AFT:
  - GitHub Repository
  - Workshop
  - Documentation
- AWS Control Tower documentation

Additional information

To view AFT queries on the CloudWatch Logs Insights dashboard, choose the **Saved and sample queries** icon from the upper-right corner, as illustrated in the following screenshot:


Saved and sample queries
Queries
Saved queries
Learn more
Filter by query name
Create query
Account Factory for Terraform (2)
Customization Logs by Account ID
Customization Logs by Customization Request ID
Sample queries
Learn more
Lambda
VPC Flow Logs
CloudTrail

Connect to an Amazon EC2 instance by using Session Manager

Jason Cornick, Abhishek Bastikoppa, and Yaniv Ron, Amazon Web Services

Summary

This pattern describes how to connect to an Amazon Elastic Compute Cloud (Amazon EC2) instance by using the Session Manager, a capability of AWS Systems Manager. Using this pattern, you can run bash commands on an EC2 instance through a web browser. Session Manager doesn't require that you open inbound ports and doesn't require public IP addresses for EC2 instances. Additionally, it removes the need to maintain bastion hosts with different Secure Shell (SSH) keys. You can govern access to Session Manager with AWS Identity and Access Management (IAM) policies and configure logging, which records important information, such as instance access and actions.

In this pattern, you configure an IAM role and associate it to a Linux EC2 instance that you provision by using an Amazon Machine Image (AMI). You then configure logging in Amazon CloudWatch Logs and use Session Manager to start a session with the instance.

Although this pattern connects to a Linux EC2 instance in the Amazon Web Services (AWS) Cloud, you could use this approach to use Session Manager for connections with other servers, such as on-premises servers or other virtual machines.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Permissions to access the managed node. For instructions, see Control user session access to managed nodes.
- VPC endpoints for `ssm`, `ec2`, `ec2messages`, `ssmmessages`, and `s3`. For instructions, see Create VPC endpoints in the Systems Manager documentation.

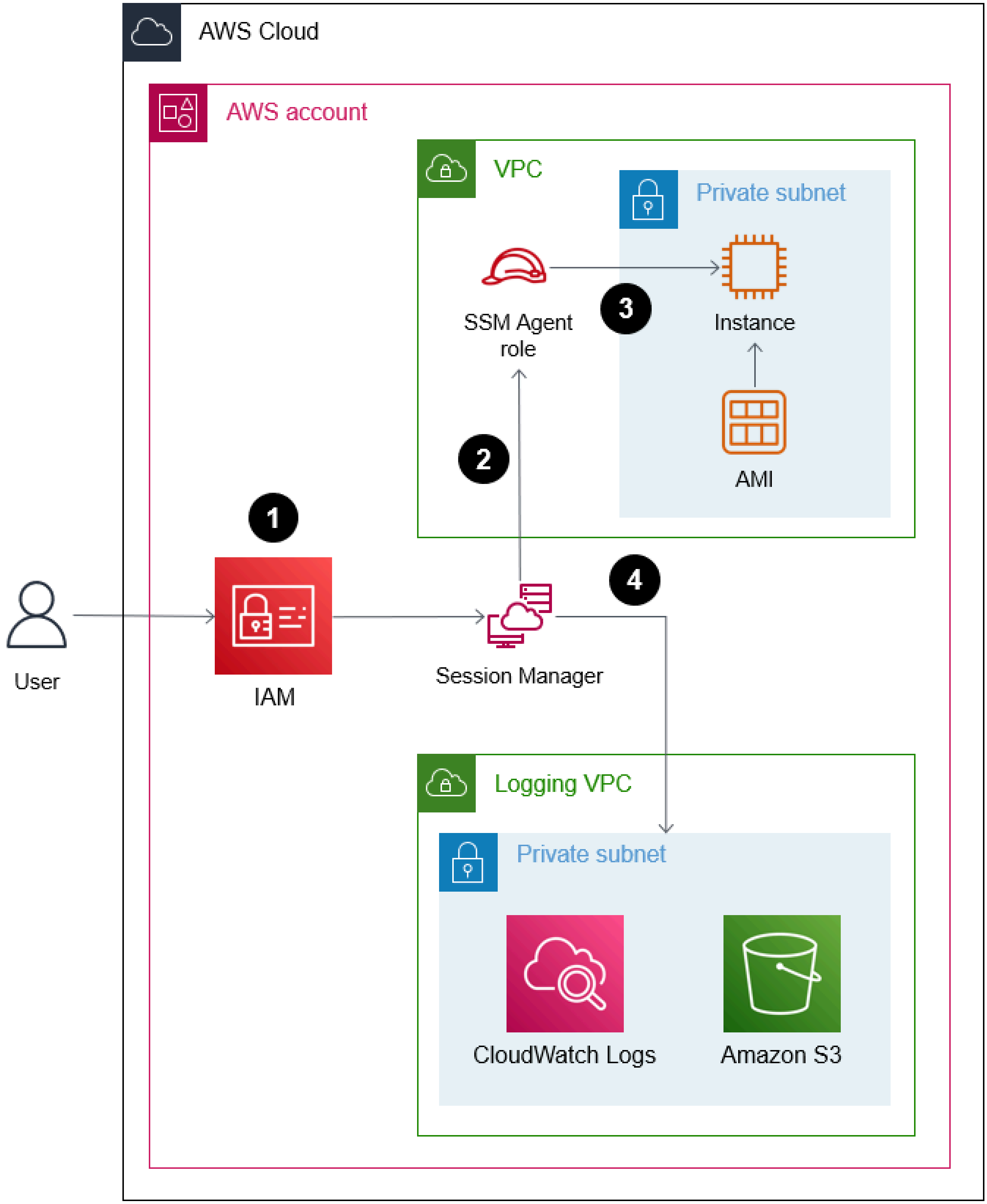
Architecture

Target technology stack

- Session Manager

- Amazon EC2
- CloudWatch Logs

Target architecture


AWS Cloud
AWS account
VPC
Private subnet
SSM Agent role
3
Instance
AMI
2
1
User
IAM
Session Manager
4
Logging VPC
Private subnet
CloudWatch Logs
Amazon S3

1. The user authenticates their identity and credentials through IAM.
2. The user initiates an SSH session through Session Manager and sends API calls to the EC2 instance.
3. The AWS Systems Manager SSM Agent, which is installed on the EC2 instance, connects to Session Manager and runs the commands.
4. For auditing and monitoring purposes, Session Manager sends the logging data to CloudWatch Logs. Alternatively, you can send log data to an Amazon Simple Storage Service (Amazon S3) bucket. For more information, see Logging session data using Amazon S3 (Systems Manager documentation).

Tools

AWS services

- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down. This pattern uses an Amazon Machine Image (AMI) to provision a Linux EC2 instance.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale. This pattern uses Session Manager, a capability of Systems Manager.

Best practices

We recommend that you read more about the security pillar of the AWS Well-Architected Framework and explore encryption options and apply the security recommendations in Setting up Session Manager (Systems Manager documentation).

Epics

Set up the infrastructure

Task	Description	Skills required
Create the IAM role.	Create the IAM role for the SSM Agent. Follow the instructions in [Creating a role for an AWS service](#) (IAM documentation) and note the following: 1. For the AWS service, choose **EC2**. 2. For **Permissions Policies**, choose `AmazonSSMManagedInstanceCore` . 3. In **Role Name**, enter `EC2_SSM_Role` .	AWS systems administrator
Create the EC2 instance.	1. Create the EC2 instance. Follow the instructions in [Launch an instance](#) (Amazon EC2 documentation) and note the following: a. In the **Name and tags** section, choose **Add additional tags**. In **Key**, enter `Name`, and in **Value**, enter `Production_Server_One` . b. Choose an Amazon Linux AMI that has the SSM Agent preinstalled.	AWS systems administrator

Task	Description	Skills required
	For a complete list, see AMIs with SSM Agent preinstalled (Systems Manager documentation). c. In the **Advanced details** section, in **IAM instance profile**, choose **EC2_SSM_Role**. 2. Open the Systems Manager console at https://console.aws.amazon.com/systems-manager/. 3. In the navigation pane, choose **Fleet Manager**. 4. Verify that the instance appears in the list of managed nodes.	

Task	Description	Skills required
Set up logging.	1. Create a log group in CloudWatch Logs. Follow the instructions in [Create a log group](#) (CloudWatch Logs documentation). Name the new log group `SessionManager` . 2. Configure logging for Session Manager. Follow the instructions in [Logging session data using Amazon CloudWatch Logs](#) (Systems Manager documentation) and note the following: a. Don't select **Allow only encrypted CloudWatch log groups**. b. In **Choose a log group from the list**, choose **SessionManager**.	AWS systems administrator

Connect to the instance

Task	Description	Skills required
Connect to the EC2 instance.	1. Start a session in the Systems Manager console. For instructions, see [Start a session](#) (Systems Manager documentation). For **Target instances**, choose the option button to the	AWS systems administrator

Task	Description	Skills required
	left of the **Production_Server_One** instance. 2. After the connection is made, run several bash commands. 3. In the Systems Manager console, end the session. For instructions, see [End a session](#) (Systems Manager documentation).	
Validate logging.	1. In CloudWatch Logs, open the log stream for the log group. For instructions, see [View log data](#) (CloudWatch Logs documentation). 2. In the log data, confirm that the commands you ran in the previous story are listed.	AWS systems administrator

Troubleshooting

Issue	Solution
IAM issues	For support, see Troubleshooting (IAM documentation).

Related resources

- Complete Session Manager prerequisites (Systems Manager documentation)
- Designing and implementing logging and monitoring with Amazon CloudWatch (AWS Prescriptive Guidance)

Create a pipeline in AWS Regions that don’t support AWS CodePipeline

Anand Krishna Varanasi, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

AWS CodePipeline is a continuous delivery (CD) orchestration service that’s part of a set of DevOps tools from Amazon Web Services (AWS). It integrates with a large variety of sources (such as version control systems and storage solutions), continuous integration (CI) products and services from AWS and AWS Partners, and open-source products to provide an end-to-end workflow service for fast application and infrastructure deployments.

However, CodePipeline isn’t supported in all AWS Regions, and it’s useful to have an invisible orchestrator that connects AWS CI/CD services. This pattern describes how to implement an end-to-end workflow pipeline in AWS Regions where CodePipeline isn’t yet supported by using AWS CI/CD services such as AWS CodeCommit, AWS CodeBuild, and AWS CodeDeploy.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Cloud Development Kit (AWS CDK) CLI version 2.28 or later

Architecture

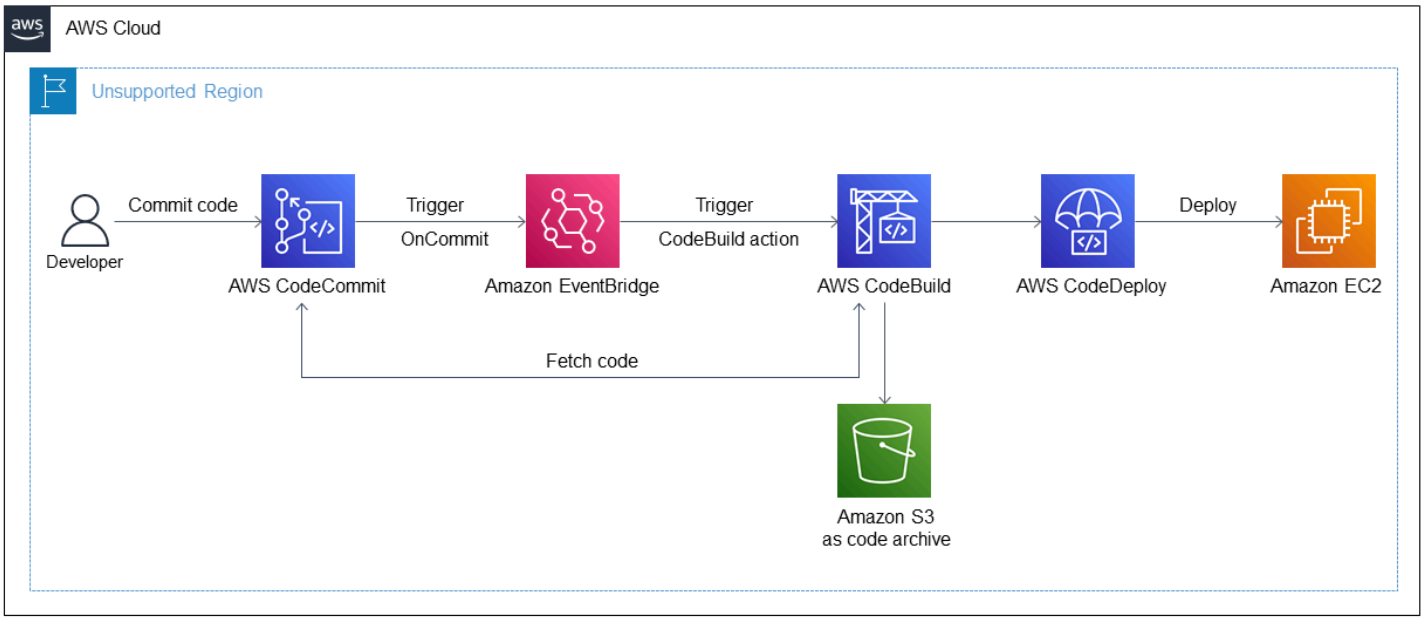
Target technology stack

The following diagram shows a pipeline that was created in a Region that doesn’t support CodePipeline, such as the Africa (Cape Town) Region. A developer pushes the CodeDeploy configuration files (also called *deployment lifecycle hook scripts*) to the Git repository that’s hosted by CodeCommit. (See the GitHub repository provided with this pattern.) An Amazon EventBridge rule automatically initiates CodeBuild.

The CodeDeploy configuration files are fetched from CodeCommit as part of the source stage of the pipeline and transferred to CodeBuild.

In the next phase, CodeBuild performs these tasks:

1. Downloads the application source code TAR file. You can configure the name of this file by using Parameter Store, a capability of AWS Systems Manager.
2. Downloads the CodeDeploy configuration files.
3. Creates a combined archive of application source code and CodeDeploy configuration files that are specific to the application type.
4. Initiates CodeDeploy deployment to an Amazon Elastic Compute Cloud (Amazon EC2) instance by using the combined archive.



Tools

AWS services

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS CodeDeploy automates deployments to Amazon EC2 or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.

Code

The code for this pattern is available in the GitHub CodePipeline Unsupported Regions repository.

Epics

Set up your developer workstation

Task	Description	Skills required
Install the AWS CDK CLI.	For instructions, see the AWS CDK documentation.	AWS DevOps
Install a Git client.	To create commits, you can use a Git client installed on your local computer, and then push your commits to the CodeCommit repository. To set up CodeCommit with your Git client, see the CodeCommit documentation.	AWS DevOps
Install npm.	Install the **npm** package manager. For more information, see the npm documentation.	AWS DevOps

Set up the pipeline

Task	Description	Skills required
Clone the code repository.	Clone the GitHub CodePipeline Unsupported Regions repository to your local	DevOps engineer

Task	Description	Skills required
	machine by running the following command. `git clone https://github.com/aws-samples/invisible-codepipeline-unsupported-regions`	

Task	Description	Skills required
Set parameters in cdk.json.	Open the `cdk.json` file and provide values for the following parameters: `"pipeline_account" :"XXXXXXXXXXXX", "pipeline_region":" us-west-2", "repo_name": "app-dev-repo", "ec2_tag_key": "test-vm", "configName" : "cbdeployconfig", "deploymentGroupName": "cbdeploygroup", "applicationName" : "cbdeployapplication", "projectName" : "CodeBuildProject"` where: • `pipeline_account` is the AWS account where the pipeline will be built. • `pipeline_region` is the AWS Region where the pipeline will be built. • `repo_name` is the name of the CodeCommit repository. • `ec2_tag_key` is the tag attached to the EC2 instance that you want to deploy the code to.	AWS DevOps

Task	Description	Skills required
	• `configName` is the name of the CodeDeploy configuration file. • `deploymentGroupName` is the name of the CodeDeploy deployment group. • `applicationName` is the CodeDeploy application name. • `projectName` is the CodeBuild project name.	

Set up the AWS CDK construct library.	In the cloned GitHub repository, use the following commands to install the AWS CDK construct library, build your application, and synthesize to generate the AWS CloudFormation template for the application.
	`npm i aws-cdk-lib`
	`npm run build`
	`cdk synth`
		AWS DevOps

Deploy the sample AWS CDK application.	Deploy the code by running the following command in an unsupported Region (such as `af-south-1`). `cdk deploy`	AWS DevOps

Set up the CodeCommit repository for CodeDeploy

Task	Description	Skills required
Set up CI/CD for the applicati on.	Clone the CodeCommit repository that you specified in the `cdk.json` file (this is called `app-dev-repo` by default) to set up the CI/CD pipeline for the application. `git clone https://git-codecommit.us-west-2.amazonaws.com/v1/repos/app-dev-repo` where the repository name and Region depend on the values you provided in the `cdk.json` file.	AWS DevOps

Test the pipeline

Task	Description	Skills required
Test the pipeline with deployment instructions.	The `CodeDeploy_Files` folder of the GitHub [CodePipeline Unsupported Regions](#) repository includes sample files that instruct CodeDeploy to deploy the application. The `appspec.yml` file is a CodeDeploy configuration file that contains hooks to control the flow of application deployment. You can use	AWS DevOps

Task	Description	Skills required
	the sample files `index.html`, `start_server.sh`, `stop_server.sh`, and `install_dependencies.sh` to update a website that's hosted on Apache. These are examples—you can use the code in the GitHub repository to deploy any type of application. When the files are pushed to the CodeCommit repository, the invisible pipeline is initiated automatically. For deployment results, check the results of individual phases in the CodeBuild and CodeDeploy consoles.	

Related resources

- Getting started (AWS CDK documentation)
- Introduction to the Cloud Development Kit (CDK) (AWS Workshop Studio)
- AWS CDK Workshop

Customize default role names by using AWS CDK aspects and escape hatches

SANDEEP SINGH and James Jacob, Amazon Web Services

Summary

This pattern demonstrates how to customize the default names of roles that are created by AWS Cloud Development Kit (AWS CDK) constructs. Customizing role names is often necessary if your organization has specific constraints based on naming conventions. For example, your organization might set AWS Identity and Access Management (IAM) permissions boundaries or service control policies (SCPs) that require a specific prefix in role names. In such cases, the default role names generated by AWS CDK constructs might not meet these conventions and might have to be altered. This pattern addresses those requirements by using escape hatches and aspects in the AWS CDK. You use escape hatches to define custom role names, and aspects to apply a custom name to all roles, to ensure adherence to your organization's policies and constraints.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Prerequisites specified in the AWS CDK documentation

Limitations

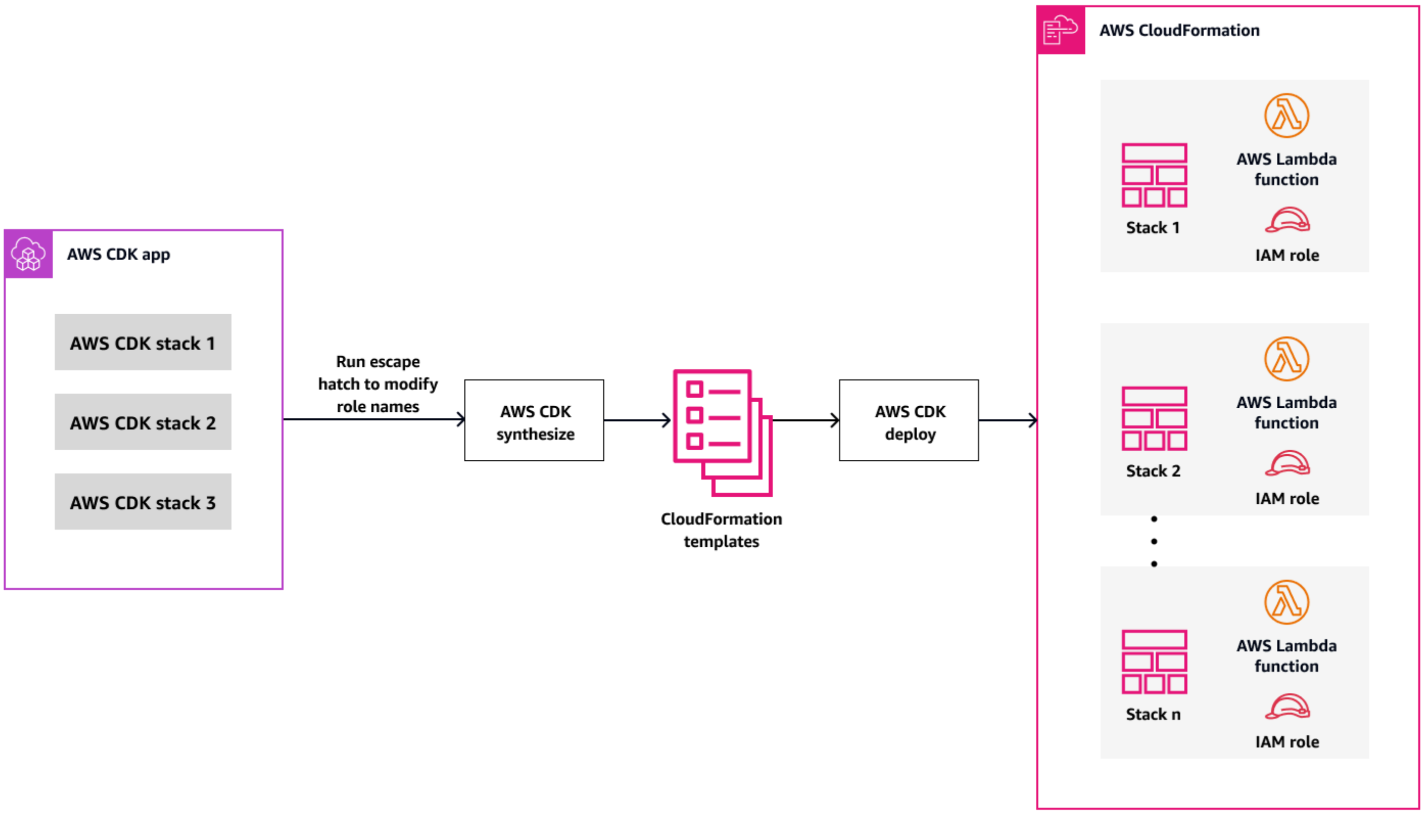
- Aspects filter resources based on resource types, so all roles share the same prefix. If you require different role prefixes for different roles, additional filtering based on other properties is necessary. For example, to assign different prefixes to roles that are associated with AWS Lambda functions, you could filter by specific role attributes or tags, and apply one prefix for Lambda-related roles and a different prefix for other roles.
- IAM role names have a maximum length of 64 characters, so modified role names have to be trimmed to meet this restriction.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

Target technology stack

- AWS CDK
- AWS CloudFormation

Target architecture



- An AWS CDK app consists of one or more AWS CloudFormation stacks, which are synthesized and deployed to manage AWS resources.
- To modify a property of an AWS CDK-managed resource that isn't exposed by a layer 2 (L2) construct, you use an escape hatch to override the underlying CloudFormation properties (in this case, the role name), and an aspect to apply the role to all resources in the AWS CDK app during the AWS CDK stack synthesis process.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CDK Command Line Interface (AWS CDK CLI) (also referred to as the AWS CDK Toolkit) is a command line cloud development kit that helps you interact with your AWS CDK app. The CLI `cdk` command is the primary tool for interacting with your AWS CDK app. It runs your app, interrogates the application model you defined, and produces and deploys the CloudFormation templates that are generated by the AWS CDK.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.

Code repository

The source code and templates for this pattern are available in the GitHub CDK Aspects Override repository.

Best practices

See Best practices for using the AWS CDK in TypeScript to create IaC projects on the AWS Prescriptive Guidance website.

Epics

Install the AWS CDK CLI

Task	Description	Skills required
Install the AWS CDK CLI.	To install the AWS CDK CLI globally, run the command: `npm install -g aws-cdk`	AWS DevOps
Verify the version.	Run the command: `cdk --version` Confirm that you're using version 2 of the AWS CDK CLI.	AWS DevOps

Task	Description	Skills required
Bootstrap the AWS CDK environment.	Before you deploy the AWS CloudFormation templates, prepare the account and AWS Region that you want to use. Run the command: `cdk bootstrap <account>/<Region>` For more information, see [AWS CDK bootstrapping] in the AWS documentation.	AWS DevOps

Deploy the AWS CDK app to demonstrate the use of aspects

Task	Description	Skills required

Set up the project.	1. Clone the GitHub repository for this pattern to your local computer:
	`git clone https://github.com/aws-samples/cdk-aspects-override`
	2. Navigate to the project directory on your local computer.
	3. Install the project dependencies:
	`npm ci`
		AWS DevOps

Task	Description	Skills required
Deploy stacks with default role names assigned by the AWS CDK.	Deploy two CloudFormation stacks (`ExampleStack1` and `ExampleStack2`) that contain the Lambda functions and their associated roles: `npm run deploy:ExampleAppWithoutAspects` The code doesn't explicitly pass role properties, so the role names will be constructed by the AWS CDK. For example output, see the Additional information section.	AWS DevOps

Task	Description	Skills required

Deploy stacks with aspects.	In this step, you apply an aspect that enforces a role name convention by adding a prefix to all IAM roles that are deployed in the AWS CDK project. The aspect is defined in the `lib/aspects.ts` file. The aspect uses an escape hatch to override the role name by adding a prefix. The aspect is applied to the stacks in the `bin/app-with-aspects.ts` file. The role name prefix used in this example is `dev-unicorn`.
	
	1. Edit the `bin/app-with-aspects.ts` file.
	2. In the file, update the variable `ROLE_NAME_PREFIX` with the prefix `dev-unicorn`:
	
	`const app = new cdk.App();`
	
	`// Define a prefix for the role names`
	`const ROLE_NAME_PREFIX = 'dev-unicorn';`
	
	`// Instantiate the RoleNamingConventionAspect with the desired prefix`
		AWS DevOps

Task	Description	Skills required

	`const roleNamingConventionAspect = new RoleNamingConventionAspect(ROLE_NAME_PREFIX);`
	3. Deploy the AWS CDK app with aspects:
	`npm run deploy:ExampleAppWithAspects`
	
	For example output, see the Additional information section.	

Clean up resources

Task	Description	Skills required
Delete your AWS CloudFormation stacks.	After you finish using this pattern, run the following command to clean up resources to avoid incurring additional costs: `cdk destroy --all -f && cdk --app npx ts-node bin/app-with-aspects.ts' destroy --all -f`	AWS DevOps

Troubleshooting

Issue	Solution
You encounter problems using the AWS CDK.	See Troubleshooting common AWS CDK issues in the AWS CDK documentation.

Related resources

- AWS Cloud Development Kit (AWS CDK)
- AWS CDK documentation
- AWS CDK on GitHub
- Escape hatches
- Aspects and the AWS CDK

Additional information

Role names created by AWS CloudFormation without aspects

```
Outputs:
ExampleStack1WithoutAspects.Function1RoleName = example-stack1-without-as-
Function1LambdaFunctionSe-y7FYTY6FXJXA
ExampleStack1WithoutAspects.Function2RoleName = example-stack1-without-as-
Function2LambdaFunctionSe-dDZV4rkWqWnI
...

Outputs:
ExampleStack2WithoutAspects.Function3RoleName = example-stack2-without-as-
Function3LambdaFunctionSe-ygMv49iTyMq0
```

Role names created by AWS CloudFormation with aspects

```
Outputs:
ExampleStack1WithAspects.Function1RoleName = dev-unicorn-
Function1LambdaFunctionServiceRole783660DC
ExampleStack1WithAspects.Function2RoleName = dev-unicorn-
Function2LambdaFunctionServiceRole2C391181
...
```

```
Outputs:
ExampleStack2WithAspects.Function3RoleName = dev-unicorn-
Function3LambdaFunctionServiceRole4CAA721C
```

Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing

Dipin Jain, Amazon Web Services

Summary

The private IP of an Amazon Elastic Compute Cloud (Amazon EC2) instance is retained throughout its lifecycle. However, the private IP might change during a planned or unplanned system crash; for example, during an Amazon Machine Image (AMI) upgrade. In some scenarios, retaining a private static IP can enhance the performance and recovery time of workloads. For example, using a static IP for an Apache Cassandra seed node prevents the cluster from incurring a rebalancing overhead.

This pattern describes how to attach a secondary elastic network interface to EC2 instances to keep the IP static during rehosting. The pattern focuses on Cassandra clusters, but you can use this implementation for any architecture that benefits from private static IPs.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Service (AWS) account

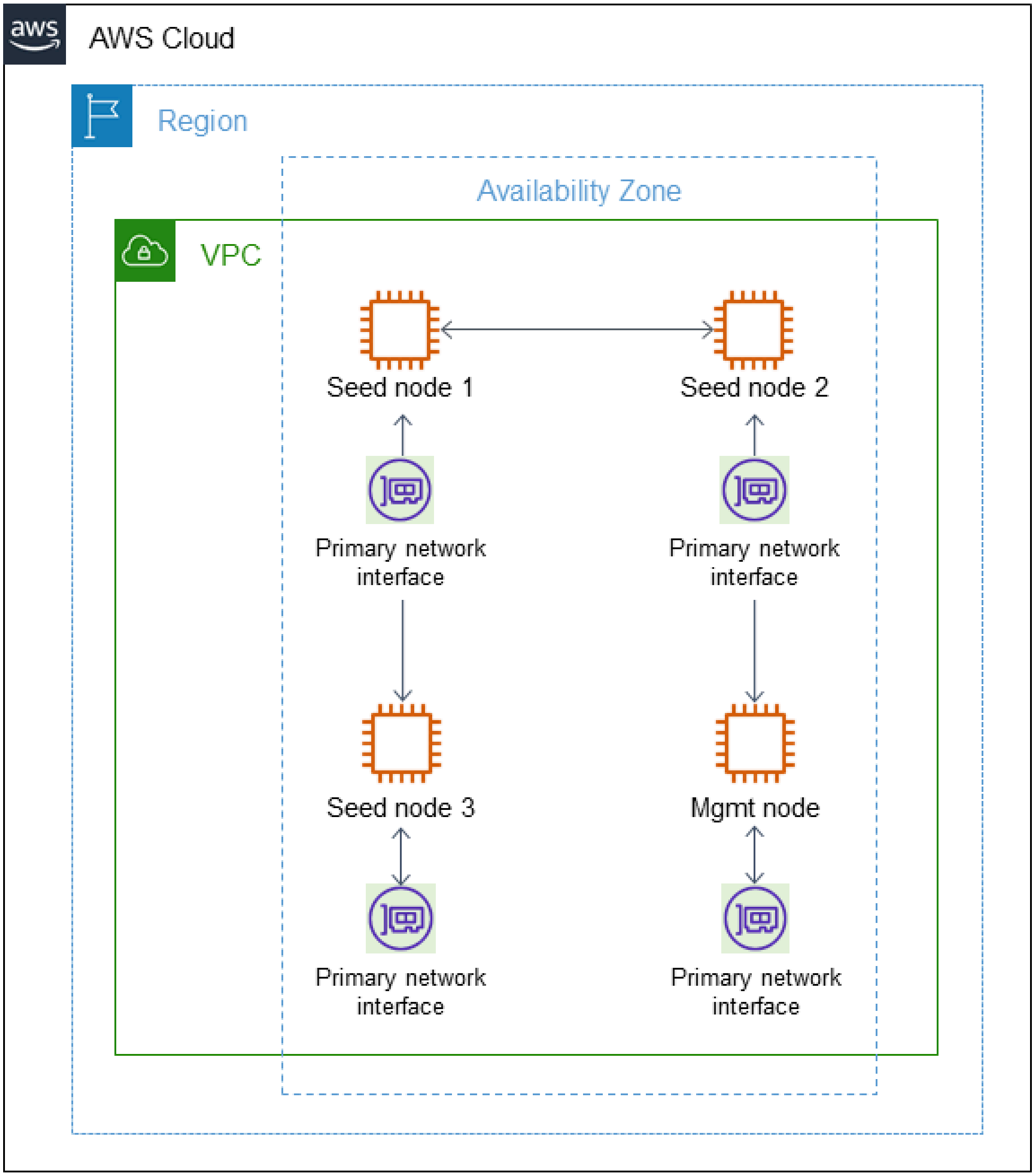
Product versions

- DataStax version 5.11.1
- Operating system: Ubuntu 16.04.6 LTS

Architecture

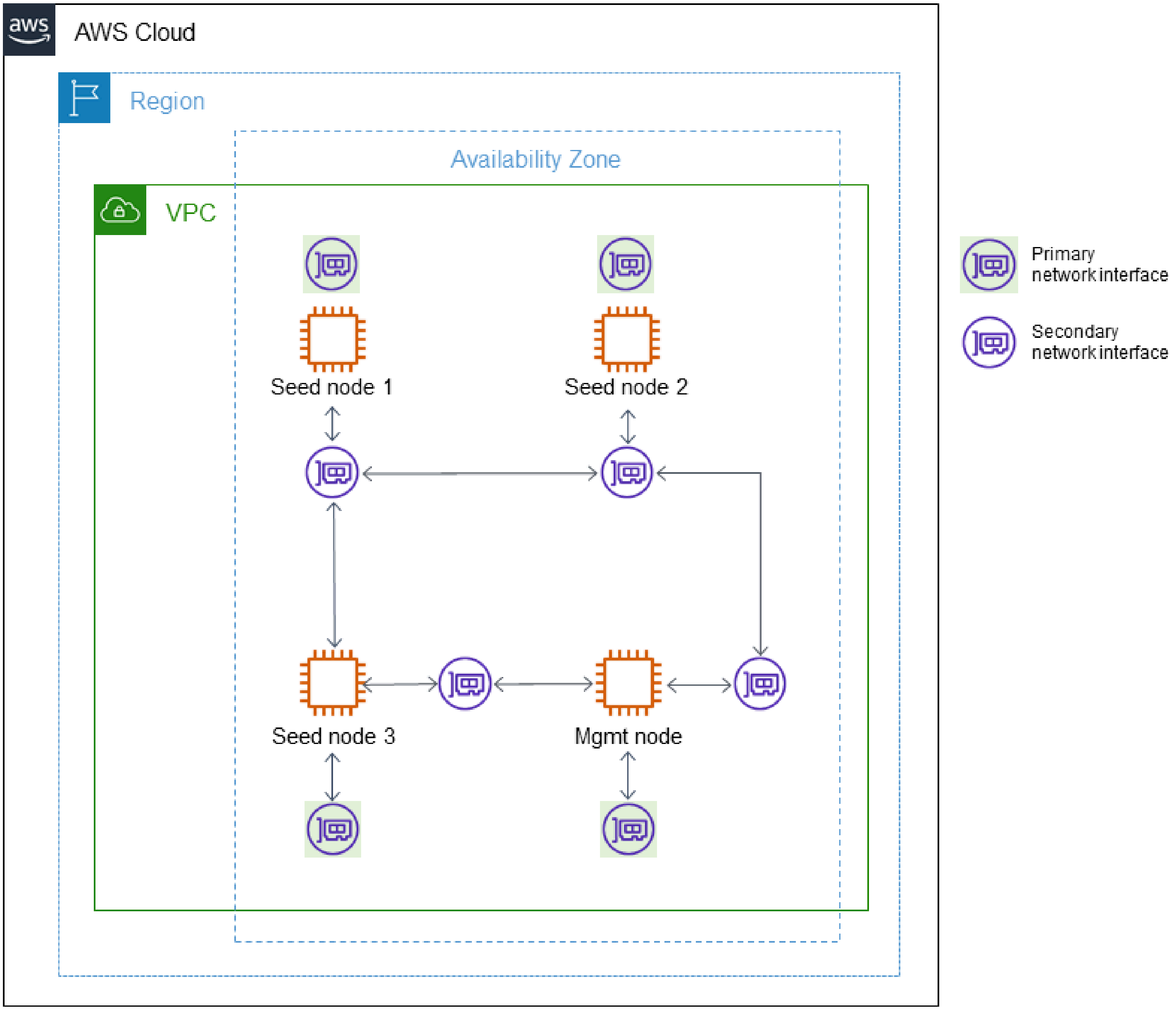
Source architecture

The source could be a Cassandra cluster on an on-premises virtual machine (VM) or on EC2 instances in the AWS Cloud. The following diagram illustrates the second scenario. This example includes four cluster nodes: three seed nodes and one management node. In the source architecture, each node has a single network interface attached.



Target architecture

The destination cluster is hosted on EC2 instances with a secondary elastic network interface attached to each node, as illustrated in the following diagram.



Automation and scale

You can also automate attaching a second elastic network interface to an EC2 Auto Scaling group, as described in an AWS Knowledge Center video.

Epics

Configure a Cassandra cluster on Amazon EC2

Task	Description	Skills required
Launch EC2 nodes to host a Cassandra cluster.	On the Amazon EC2 console, launch four EC2 instances for your Ubuntu nodes in your AWS account. Three (seed) nodes are used for the Cassandra cluster, and the fourth node acts as a cluster management node where you will install DataStax Enterprise (DSE) OpsCenter . For instructions, see the Amazon EC2 documentation.	Cloud engineer
Confirm node communica tions.	Make sure that the four nodes can communicate with one another over the database and cluster management ports.	Network engineer
Install DSE OpsCenter on the management node.	Install DSE OpsCenter 6.1 from the Debian package on the management node. For instructions, see the DataStax documentation.	DBA
Create a secondary network interface.	Cassandra generates a universal unique identifie r (UUID) for each node based on the IP address of the EC2 instance for that node. This UUID is used for distributing virtual nodes	Cloud Engineer

Task	Description	Skills required
	(vnodes) on the ring. When Cassandra is deployed on EC2 instances, IP addresses are assigned automatically to the instances as they are created. In the event of a planned or unplanned outage, the IP address for the new EC2 instance changes, the data distribution changes, and the entire ring has to be rebalanced. This is not desirable. To preserve the assigned IP address, use a [secondary elastic network interface]() with a fixed IP address. 1. On the [Amazon EC2 console](), choose **Network Interfaces**, **Create network interface**. 2. For **Subnet**, select the subnet that you created the EC2 instance in. 3. For **Private IPv4 address**, choose **Auto-assign**. 4. For **Security groups**, select a security group, and then choose **Create network interface**. For more information about creating a network interface	

Task	Description	Skills required
	, see the Amazon EC2 documentation.	
Attach the secondary network interface to cluster nodes.	1. On the Amazon EC2 console, choose **Instances**. 2. Select the checkbox for the EC2 instance you created earlier. 3. Choose **Actions**, **Networking**, **Attach network interface**. 4. Select the network interface you created in the previous step, and then choose **Attach**. For more information about attaching a network interface, see the Amazon EC2 documentation.	Cloud engineer

Task	Description	Skills required
Add routes in Amazon EC2 to address asymmetric routing.	When you attach the second network interface, the network will very likely perform asymmetric routing. To avoid this, you can add routes for the new network interfaces. For an in-depth explanati on and remediation of asymmetric routing, see the [AWS Knowledge Center video] or [Overcoming Asymmetri c Routing on Multi-Hom e Servers] (article in *Linux Journal* by Patrick McManus, April 5, 2004).	Network engineer
Update DNS entries to point to the secondary network interface IP.	Point the fully qualified domain name (FQDN) of the node to the IP of the secondary network interface.	Network engineer
Install and configure the Cassandra cluster by using DSE OpsCenter.	When the cluster nodes are ready with the secondary network interfaces, you can install and configure the Cassandra cluster.	DBA

Recover cluster from node failure

Task	Description	Skills required
Create an AMI for the cluster seed node.	Make a backup of the nodes so you can restore them with	Backup administrator

Task	Description	Skills required
	database binaries in case of node failure. For instructi ons, see Create an AMI in the Amazon EC2 documentation.	
Recover from node failure.	Replace the failed node with a new EC2 instance launched from the AMI, and attach the secondary network interface of the failed node.	Backup administrator
Verify that the Cassandra cluster is healthy.	When the replacement node is up, verify cluster health in DSE OpsCenter.	DBA

Related resources

- Installing DSE OpsCenter 6.1 from the Debian package (DataStax documentation)
- How to make a secondary network interface work in an Ubuntu EC2 instance (AWS Knowledge Center video)
- Best Practices for Running Apache Cassandra on Amazon EC2 (AWS blog post)

Extend VRFs to AWS by using AWS Transit Gateway Connect

Adam Till, Yashar Araghi, Vikas Dewangan, and Mohideen HajaMohideen, Amazon Web Services

Summary

Virtual routing and forwarding (VRF) is a feature of traditional networks. It uses isolated logical routing domains, in the form of route tables, to separate network traffic within the same physical infrastructure. You can configure AWS Transit Gateway to support VRF isolation when you connect your on-premises network to AWS. This pattern uses a sample architecture to connect on-premises VRFs to different transit gateway route tables.

This pattern uses transit virtual interfaces (VIFs) in AWS Direct Connect and transit gateway Connect attachments to extend the VRFs. A transit VIF is used to access one or more Amazon VPC transit gateways that are associated with Direct Connect gateways. A transit gateway Connect attachment connects a transit gateway with a third-party virtual appliance that is running in a VPC. A transit gateway Connect attachment supports the Generic Routing Encapsulation (GRE) tunnel protocol for high performance, and it supports Border Gateway Protocol (BGP) for dynamic routing.

The approach described in this pattern has the following benefits:

- Using Transit Gateway Connect, you can advertise up to 1,000 routes to the Transit Gateway Connect peer and receive up to 5,000 routes from it. Using the Direct Connect transit VIF feature without Transit Gateway Connect is limited to 20 prefixes per transit gateway.
- You can maintain the traffic isolation and use Transit Gateway Connect to provide hosted services on AWS, regardless of the IP address schemas your customers are using.
- The VRF traffic doesn't need to traverse a public virtual interface. This makes it easier to adhere to compliance and security requirements in many organizations.
- Each GRE tunnel supports up to 5 Gbps, and you can have up to four GRE tunnels per transit gateway Connect attachment. This is faster than many other connection types, such as AWS Site-to-Site VPN connections that support up to 1.25 Gbps.

Prerequisites and limitations

Prerequisites

- The required AWS accounts have been created (see the architecture for details)

- Permissions to assume an AWS Identity and Access Management (IAM) role in each account.
- The IAM roles in each account must have permissions to provision AWS Transit Gateway and AWS Direct Connect resources. For more information, see Authentication and access control for your transit gateways and see Identity and access management for Direct Connect.
- The Direct Connect connections have successfully been created. For more information, see Create a connection using the Connection wizard.

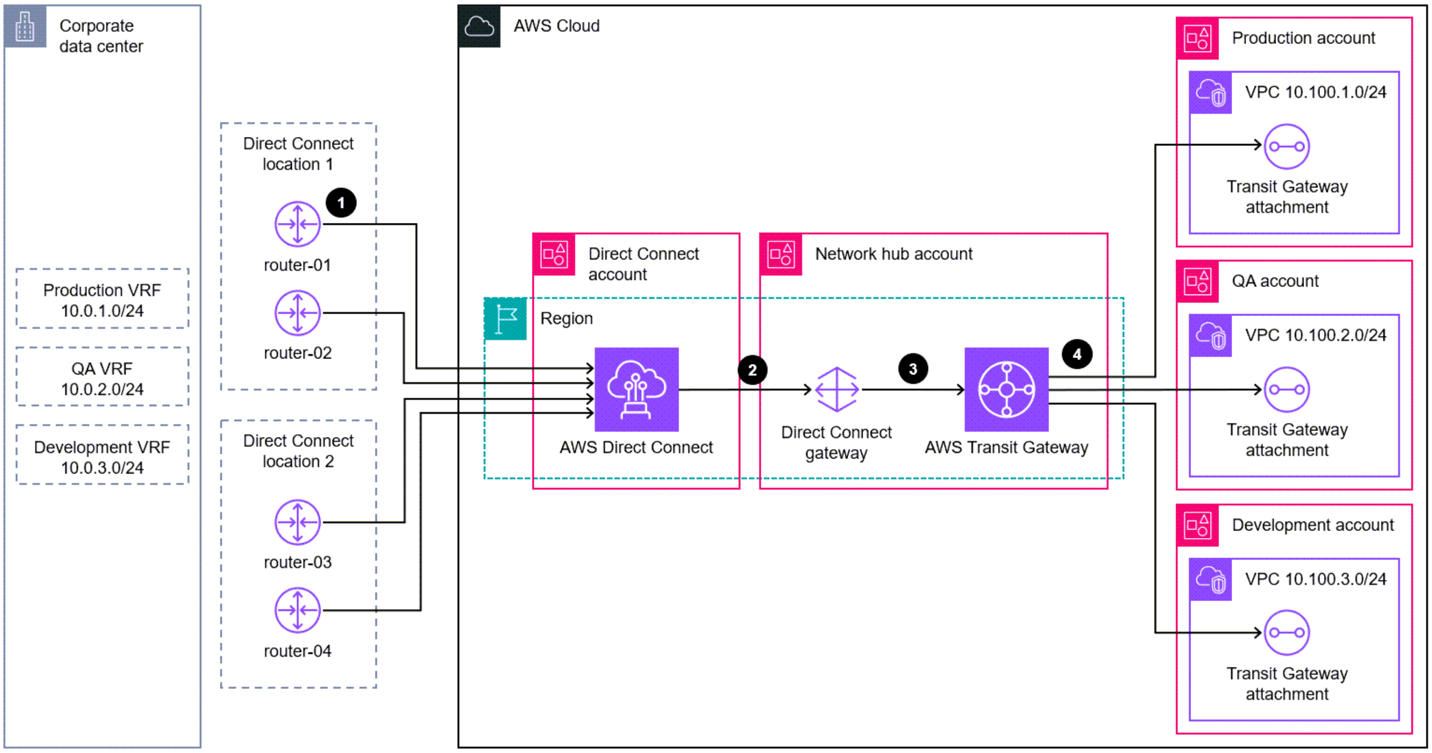
Limitations

- There are limits for transit gateway attachments to the VPCs in the production, QA, and development accounts. For more information, see Transit gateway attachments to a VPC.
- There are limits for creating and using Direct Connect gateways. For more information, see AWS Direct Connect quotas.

Architecture

Target architecture

The following sample architecture provides a reusable solution to deploy transit VIFs with transit gateway Connect attachments. This architecture provides resilience by using multiple Direct Connect locations. For more information, see Maximum resiliency in the Direct Connect documentation. The on-premises network has production, QA, and development VRFs that are extended to AWS and isolated by using dedicated route tables.

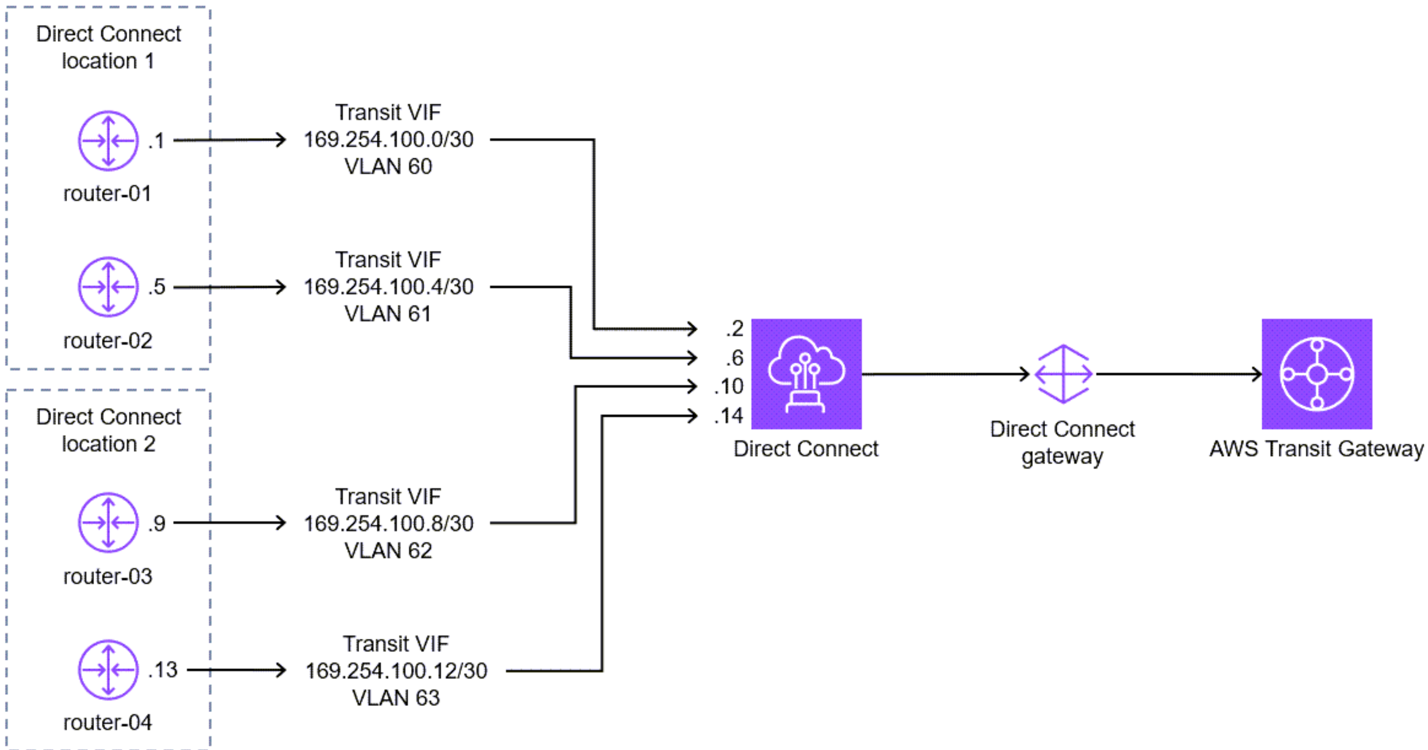


In the AWS environment, two accounts are dedicated to extending the VRFs: a *Direct Connect account* and a *network hub account*. The Direct Connect account contains the connection and the transit VIFs for each router. You create the transit VIFs from the Direct Connect account but deploy them to the network hub account so that you can associate them with the Direct Connect gateway in the network hub account. The network hub account contains the Direct Connect gateway and transit gateway. The AWS resources are connected as follows:

1. Transit VIFs connect the routers in the Direct Connect locations with AWS Direct Connect in the Direct Connect account.
2. A transit VIF connects Direct Connect with the Direct Connect gateway in the network hub account.
3. A transit gateway association connects the Direct Connect gateway with the transit gateway in the network hub account.
4. Transit gateway Connect attachments connect the transit gateway with the VPCs in the production, QA, and development accounts.

Transit VIF architecture

The following diagram shows the configuration details for the transit VIFs. This sample architecture uses a VLAN for the tunnel source, but you could also use a loopback.

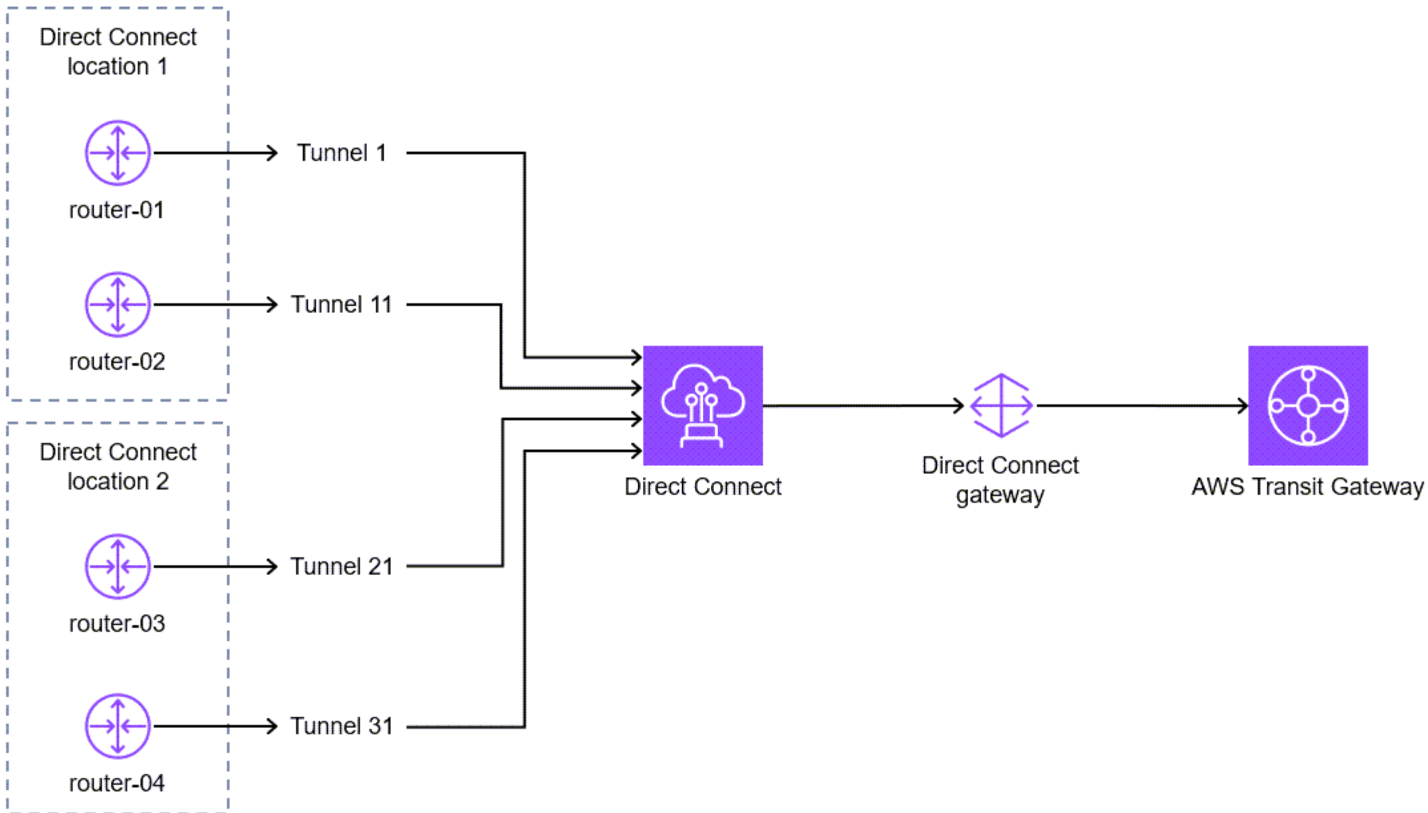


The following are the configuration details, such as autonomous system numbers (ASNs), for the transit VIFs.

Resource	Item	Detail
router-01	ASN	65534
router-02	ASN	65534
router-03	ASN	65534
router-04	ASN	65534
Direct Connect gateway	ASN	64601
Transit gateway	ASN	64600
	CIDR block	10.100.254.0/24

Transit gateway Connect architecture

The following diagram and tables describe how to configure a single VRF through a transit gateway Connect attachment. For additional VRFs, assign unique tunnel IDs, transit gateway GRE IP addresses, and BGP inside CIDR blocks. The peer GRE IP address matches the router peer IP address from the transit VIF.



The following table contains router configuration details.

Router	Tunnel	IP address	Source	Destination
router-01	Tunnel 1	169.254.101.17	VLAN 60 169.254.100.1	10.100.254.1
router-02	Tunnel 11	169.254.101.81	VLAN 61 169.254.100.5	10.100.254.11
router-03	Tunnel 21	169.254.1 01.145	VLAN 62	10.100.254.21

			169.254.100.9	
router-04	Tunnel 31	169.254.101.209	VLAN 63 169.254.100.13	10.100.254.31

The following table contains transit gateway configuration details.

Tunnel	Transit gateway GRE IP address	Peer GRE IP address	BGP inside CIDR blocks
Tunnel 1	10.100.254.1	VLAN 60 169.254.100.1	169.254.101.16/29
Tunnel 11	10.100.254.11	VLAN 61 169.254.100.5	169.254.101.80/29
Tunnel 21	10.100.254.21	VLAN 62 169.254.100.9	169.254.101.144/29
Tunnel 31	10.100.254.31	VLAN 63 169.254.100.13	169.254.101.208/29

Deployment

The Epics section describes how to deploy a sample configuration for a single VRF across multiple customer routers. After steps 1–5 are complete, you can create new transit gateway Connect attachments by using steps 6–7 for every new VRF that you're extending into AWS:

1. Create the transit gateway.
2. Create a Transit Gateway route table for each VRF.
3. Create the transit virtual interfaces.
4. Create the Direct Connect gateway.

5. Create the Direct Connect gateway virtual interface and gateway associations with allowed prefixes.
6. Create the transit gateway Connect attachment.
7. Create the Transit Gateway Connect peers.
8. Associate the transit gateway Connect attachment with the route table.
9. Advertise routes to the routers.

Tools

AWS services

- AWS Direct Connect links your internal network to a Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- AWS Transit Gateway is a central hub that connects virtual private clouds (VPCs) and on-premises networks.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Plan the architecture

Task	Description	Skills required
Create custom architecture diagrams.	1. In the Attachments section, download the diagram template. 2. Open the attached diagram in Microsoft Office PowerPoint. 3. On the **Architecture overview** slide, customize the architecture diagram	Cloud architect, Network administrator

Task	Description	Skills required
	for your environment. Identify the on-premises VRFs that need to be extended into your AWS environment. 4. On the **Transit VIF** slide, customize the architecture diagram. Identify the AS numbers of the routers, the Direct Connect gateway, and the transit gateway. Identify the IP addresses at each end of the transit VIF. 5. On the **Transit Gateway Connect** slide, customize an architecture diagram for each VRF. Identify all required IP addresses necessary to configure the routers and the Transit Gateway Connect peers.	

Create the Transit Gateway resources

Task	Description	Skills required
Create the transit gateway.	1. Sign in to the network hub account. 2. Follow the instructions in [Create a transit gateway](). Note the following for this pattern:	Network administrator, Cloud architect

Task	Description	Skills required
	• For **Amazon side Autonomous System Number (ASN)**, enter a unique ASN. For the purposes of this example, the ASN is `64600`. • Select **DNS support**. • For this sample architecture, **VPN ECMP support**, **Default route table association**, **Default route table prorogation**, and **Multicast support** are not required. • For **Transit gateway CIDR blocks**, enter the IPv4 CIDR blocks for your transit gateway. For the purposes of this example, the CIDR block is `10.100.254.0/24` .	

Task	Description	Skills required
Create the transit gateway route table.	Follow the instructions in [Create a transit gateway route table](). Note the following for this pattern: • For **Name tag**, provide a name for the transit gateway route table. We recommend using a name that corresponds to the VRF, such as `routetable-dev-vrf` . • For **Transit gateway ID**, choose the transit gateway that you created previously.	Cloud architect, Network administrator

Create the transit virtual interfaces

Task	Description	Skills required
Create the transit virtual interfaces.	1. Sign in to the Direct Connect account. 2. Follow the instructions in [Create a transit virtual interface to the Direct Connect gateway](). Note the following for this pattern: • For **Virtual interface name**, enter a name for the transit VIF. We recommend using a name that corresponds to the router, such as	Cloud architect, Network administrator

Task	Description	Skills required

	`transit-vif-router 01` .
	• For **Connection**, select the router, such as `router-01` .
	• For **Virtual interface owner**, enter the account ID of the network hub account. For instructions, see [View your AWS account ID](#).
	• For **Direct Connect gateway**, do not make any selection. You attach the Direct Connect gateway in a subsequent step.
	• For **VLAN**, enter the VLAN of the router, such as `60`.
	• For **BGP ASN**, enter the ASN of the router, such as `65534`.
	• Under **Additional** Settings, do the following:
	  • Choose **IPv4**.
	  • For **Your router peer ip**, enter the router peer IP address, such as `169.254.100.1` .
	  • For **Amazon router peer ip**. enter the	

Task	Description	Skills required
	Amazon router peer IP, such as `169.254.100.2` . • For **BGP authentication key**, a password is required. If this is left blank, AWS creates a key that is only accessible in this account. 3. Repeat these instructions to create all transit VIFs for the VRF.	

Create the Direct Connect resources

Task	Description	Skills required
Create a Direct Connect gateway.	1. Sign in to the network hub account. 2. Follow the instructions in [Creating a Direct Connect gateway](). Note the following for this pattern: • For **Amazon side ASN**, enter the ASN of the Direct Connect gateway, such as `64601`. • Do not choose a virtual private gateway.	Cloud architect, Network administrator
Attach the Direct Connect gateway to the transit VIFs.	1. In the network hub account, open the AWS Direct Connect console	Cloud architect, Network administrator

Task	Description	Skills required
	at https://console.aws.amazon.com/directconnect/v2/. 2. In the navigation pane, choose **Virtual Interfaces**. 3. Select a new transit VIF, and then choose **Accept**. 4. Choose the Direct Connect gateway you created. 5. Repeat these instructions for each transit VIF.	
Create the Direct Connect gateway associations with allowed prefixes.	In the network hub account, follow the instructions in To associate a transit gateway. Note the following for this pattern: • For **Gateways**, choose the transit gateway you created previously. • For **Allowed prefixes**, enter the CIDR block assigned to the transit gateway, such as `10.100.254.0/24` . Creating this association automatically creates a Transit Gateway attachment that has a Direct Connect Gateway resource type. This attachment does not need to be associated with a transit gateway route table.	Cloud architect, Network administrator

Task	Description	Skills required
Create the transit gateway Connect attachment.	1. In the network hub account, open the Amazon VPC console at https://console.aws.amazon.com/vpc/. 2. In the navigation pane, choose **Transit gateway attachments**. 3. Choose **Create transit gateway attachment**. 4. For **Name tag**, enter a name for the attachmen t. We recommend using a name that corresponds to the VRF, such as `PROD-VRF`. 5. For **Transit gateway ID**, choose the transit gateway you created previously. 6. For **Attachment type**, choose **Connect**. 7. For **Transport attachmen t ID**, choose the Direct Connect gateway you created previously. 8. Choose **Create transit gateway attachment**. 9. Repeat this step for each VRF that you are extending .	Cloud architect, Network administrator

Task	Description	Skills required
Create the Transit Gateway Connect peers.	1. In the network hub account, follow the instructions in [Create a Transit Gateway Connect peer (GRE tunnel)](). Note the following for this pattern: • For **Name tag**, enter a name for the Transit Gateway Connect peer. We recommending using a name that corresponds with the router, such as `connectpeer-router 01` . • For **Transit gateway GRE address**, enter the assigned IP address from the transit gateway CIDR block, such as `10.100.254.1` . • For **Peer GRE address**, enter the IP address assigned to the VLAN created on the router for the transit VIF, such as `169.254.1 00.1` . Provided that AWS can reach the IP address, you can use any interface, such as VLAN or Loopback, for the peer GRE address. • For **BGP Inside CIDR Blocks (IPv4)**, enter	

Task	Description	Skills required
	the BGP inside CIDR block IP address, such as `169.254.101.16/29` . • For **Peer ASN**, enter the ASN of the router, such as `65534`. 2. Repeat these instructions to create a GRE tunnel for each router.	

Advertise routes to the routers

Task	Description	Skills required
Advertise the routes.	Associate the new transit gateway Connect attachmen t with the route table you created previously for this VRF. For example, associate the production transit gateway Connect attachment with the `Production-VRF` route table. Create a static route for the prefix that is advertised to the routers. 1. Sign in to the network hub account. 2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.	Network administrator, Cloud architect

Task	Description	Skills required
	3. In the navigation pane, under **Transit Gateways**, choose **Transit gateway route tables**. 4. Select the `Production-VRF` route table. 5. On the **Actions** menu, choose **Create static route**. 6. For **CIDR**, enter the CIDR block for the advertise d route to the transit gateway attachment in the target VPC, such as `10.100.1.0/24` . 7. For **Choose Attachmen t**, choose the relevant transit gateway Connect attachment. 8. Choose **Create static route**.	

Related resources

AWS documentation

- Direct Connect documentation
  - Working with Direct Connect gateways
  - Transit gateway associations
  - AWS Direct Connect virtual interfaces
- Transit Gateway documentation
  - Working with transit gateways
  - Transit gateway attachments to a Direct Connect gateway
  - Transit gateway Connect attachments and Transit Gateway Connect peers

- Create a transit gateway Connect attachment

AWS blog posts

- Segmenting hybrid networks with AWS Transit Gateway connect
- Using AWS Transit Gateway connect to extend VRFs and increase IP prefix advertisement

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Get Amazon SNS notifications when the key state of an AWS KMS key changes

Shubham Harsora, Aromal Raj Jayarajan, and Navdeep Pareek, Amazon Web Services

Summary

The data and metadata associated with an AWS Key Management Service (AWS KMS) key is lost when that key is deleted. The deletion is irreversible and you can't recover lost data (including encrypted data). You can prevent data loss by setting up a notification system to alert you of status changes to [key states](#) of your AWS KMS keys.

This pattern shows you how to monitor status changes to AWS KMS keys by using Amazon EventBridge and Amazon Simple Notification Service (Amazon SNS) to issue automated notifications whenever the key state of an AWS KMS key changes to `Disabled` or `PendingDeletion`. For example, if a user tries to disable or delete an AWS KMS key, you will receive an email notification with details about the attempted status change. You can also use this pattern to schedule the deletion of AWS KMS keys.

Prerequisites and limitations

Prerequisites

- An active AWS account with an AWS Identity and Access Management (IAM) user
- An [AWS KMS key](#)

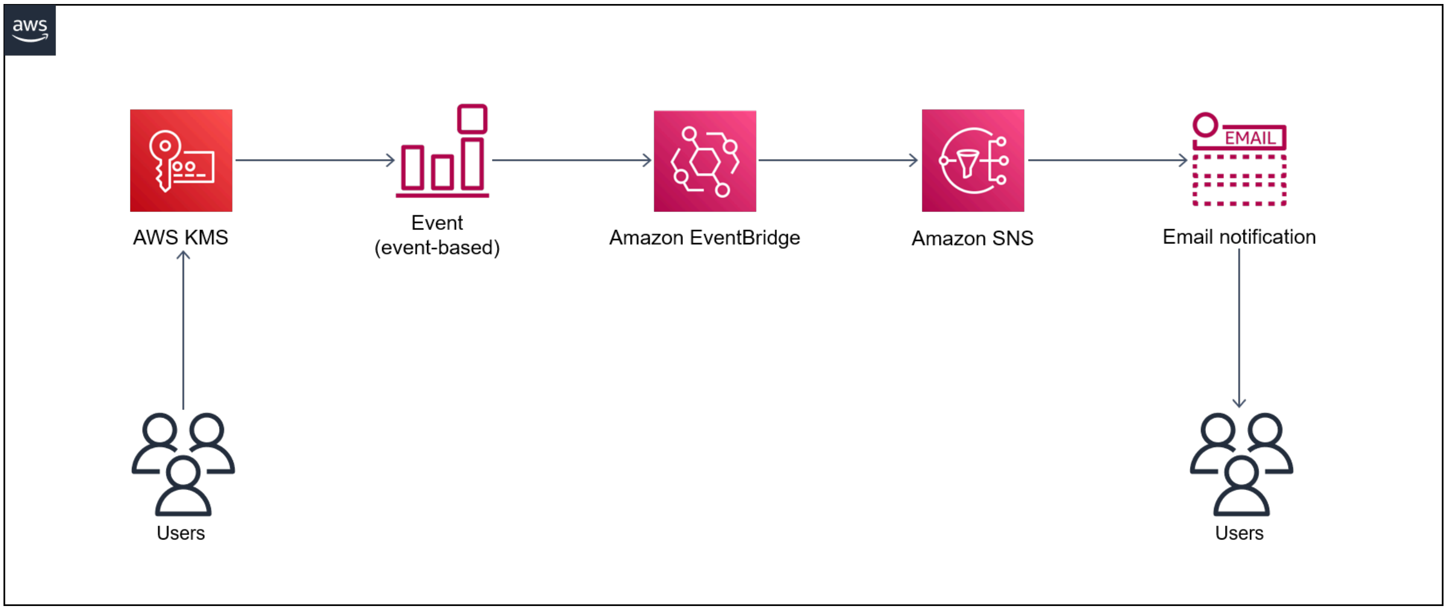
Architecture

Technology stack

- Amazon EventBridge
- AWS Key Management Service (AWS KMS)
- Amazon Simple Notification Service (Amazon SNS)

Target architecture

The following diagram shows an architecture for building an automated monitoring and notification process for detecting any changes to the state of an AWS KMS key.



The diagram shows the following workflow:

1. A user disables or schedules the deletion of an AWS KMS key.
2. An EventBridge rule evaluates the scheduled `Disabled` or `PendingDeletion` event.
3. The EventBridge rule invokes the Amazon SNS topic.
4. Amazon SNS sends an email notification message to the users.

> **Note**
>
> You can customize the email message to meet your organization's needs. We recommend including information about the entities where the AWS KMS key is used. This can help users understand the impact of deleting the AWS KMS key. You can also schedule a reminder email notification that's sent one or two days before the AWS KMS key is deleted.

Automation and scale

The AWS CloudFormation stack deploys all the necessary resources and services for this pattern to work. You can implement the pattern independently in a single account, or by using [AWS CloudFormation StackSets] for multiple independent accounts or [organizational units] in AWS Organizations.

Tools

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions. The CloudFormation template for this pattern describes all the AWS resources that you want, and CloudFormation provisions and configures those resources for you.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications and AWS services, and it routes that data to targets such as AWS Lambda. EventBridge simplifies the process of building event-driven architectures.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

Code

The code for this pattern is available in the GitHub Monitor AWS KMS keys disable and scheduled deletion repository.

Epics

Deploy the CloudFormation template

Task	Description	Skills required
Clone the repository.	Clone the GitHub Monitor AWS KMS keys disable and scheduled deletion repository to your local machine by running the following command: `git clone https://github.com/aws-samp`	AWS administrator, Cloud architect

Task	Description	Skills required
	`les/aws-kms-deletion-notification`	
Update the template's parameters.	In a code editor, open the `Alerting-KMS-Events.yaml` CloudFormation template that you cloned from the repository, and then update the following parameters: • For `DestinationEmailAddress` , enter an active email address that you plan to use for receiving the SNS notification. • For `SNSTopicName` , enter a name for your SNS topic.	AWS administrator, Cloud architect

Task	Description	Skills required
Deploy the CloudFormation template.	1. Sign in to the AWS Management Console and open the [CloudFormation console](). 2. In the navigation pane, choose **Create stack**, and then choose **With new resources (standard)**. 3. On the **Identify resources** page, choose **Next**. 4. On the **Specify template** page, for **Template source**, select **Upload a template file**. 5. Choose **Choose file**, select the `Alerting-KMS-Events.yaml` file from your cloned GitHub repository, and then choose **Next**. 6. For **Stack name**, enter your stack name. 7. Choose **Submit**.	AWS administrator, Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription email.	After the CloudFormation template successfully deploys, Amazon SNS sends a subscription confirmation message to the email address	AWS administrator, Cloud architect

Task	Description	Skills required
	that you provided in the CloudFormation template. To receive notifications, you must confirm this email subscription. For more information, see Confirm the subscription in the Amazon SNS Developer Guide.	

Test the subscription notification

Task	Description	Skills required
Disable AWS KMS keys.	1. Sign in to the AWS Management Console and open the AWS KMS console. 2. To change the Region, choose the name of the currently displayed Region, and then choose the Region to which you want to switch. 3. In the navigation pane, choose **Customer managed keys**. 4. Select the check box for the AWS KMS key that you want to enable or disable. 5. To disable the AWS KMS key, choose **Key actions**, and then choose **Disable**.	AWS administrator

Task	Description	Skills required
Validate the subscription.	Confirm that you received the Amazon SNS notification email.	AWS administrator

Clean up resources

Task	Description	Skills required
Delete the CloudFormation stack.	1. Sign in to the AWS Management Console and open the [CloudFormation console](). 2. In the navigation pane, choose **Stacks**. 3. Select that stack that you created previously, and then choose **Delete**.	AWS administrator

Related resources

- AWS CloudFormation (AWS documentation)
- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Building event-driven architectures on AWS (AWS Workshop Studio documentation)
- AWS Key Management Service Best Practices (AWS Whitepaper)
- Security best practices for AWS Key Management Service (AWS KMS Developer Guide)

Additional information

Amazon SNS provides in-transit encryption by default. To align with security best practices, you can also enable server-side encryption for Amazon SNS by using an AWS KMS customer managed key.

Preserve routable IP space in multi-account VPC designs for non-workload subnets

Adam Spicer, Amazon Web Services

Summary

Amazon Web Services (AWS) has published best practices that recommend using dedicated subnets in a virtual private cloud (VPC) for both transit gateway attachments and Gateway Load Balancer endpoints (to support AWS Network Firewall or third-party appliances). These subnets are used to contain elastic network interfaces for these services. If you use both AWS Transit Gateway and a Gateway Load Balancer, two subnets are created in each Availability Zone for the VPC. Because of the way VPCs are designed, these extra subnets can't be smaller than a /28 mask and can consume precious routable IP space that could otherwise be used for routable workloads. This pattern demonstrates how you can use a secondary, non-routable Classless Inter-Domain Routing (CIDR) range for these dedicated subnets to help preserve routable IP space.

Prerequisites and limitations

Prerequisites

- Multi-VPC strategy for routable IP space
- A non-routable CIDR range for the services you're using (transit gateway attachments and Gateway Load Balancer or Network Firewall endpoints)

Architecture

Target architecture

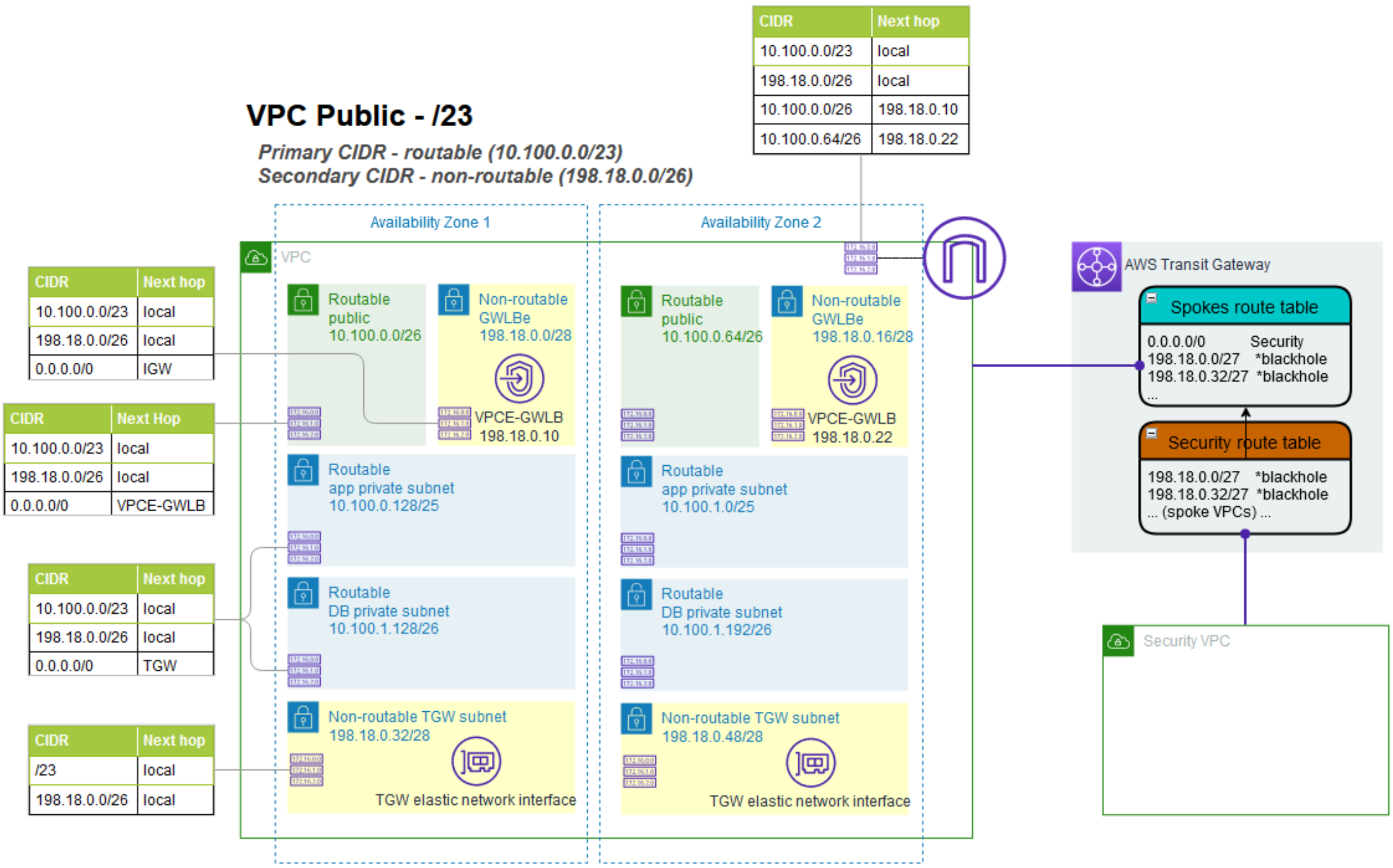
This pattern includes two reference architectures: one architecture has subnets for transit gateway (TGW) attachments and a Gateway Load Balancer endpoint (GWLBe), and the second architecture has subnets for TGW attachments only.

Architecture 1 ‒ TGW-attached VPC with ingress routing to an appliance

The following diagram represents a reference architecture for a VPC that spans two Availability Zones. On ingress, the VPC uses an ingress routing pattern to direct traffic destined for the public subnet to a bump-in-the-wire appliance for firewall inspection. A TGW attachment supports egress from the private subnets to a separate VPC.

This pattern uses a non-routable CIDR range for the TGW attachment subnet and the GWLBe subnet. In the TGW routing table, this non-routable CIDR is configured with a blackhole (static) route by using a set of more specific routes. If the routes were to get propagated to the TGW routing table, these more specific blackhole routes would apply.

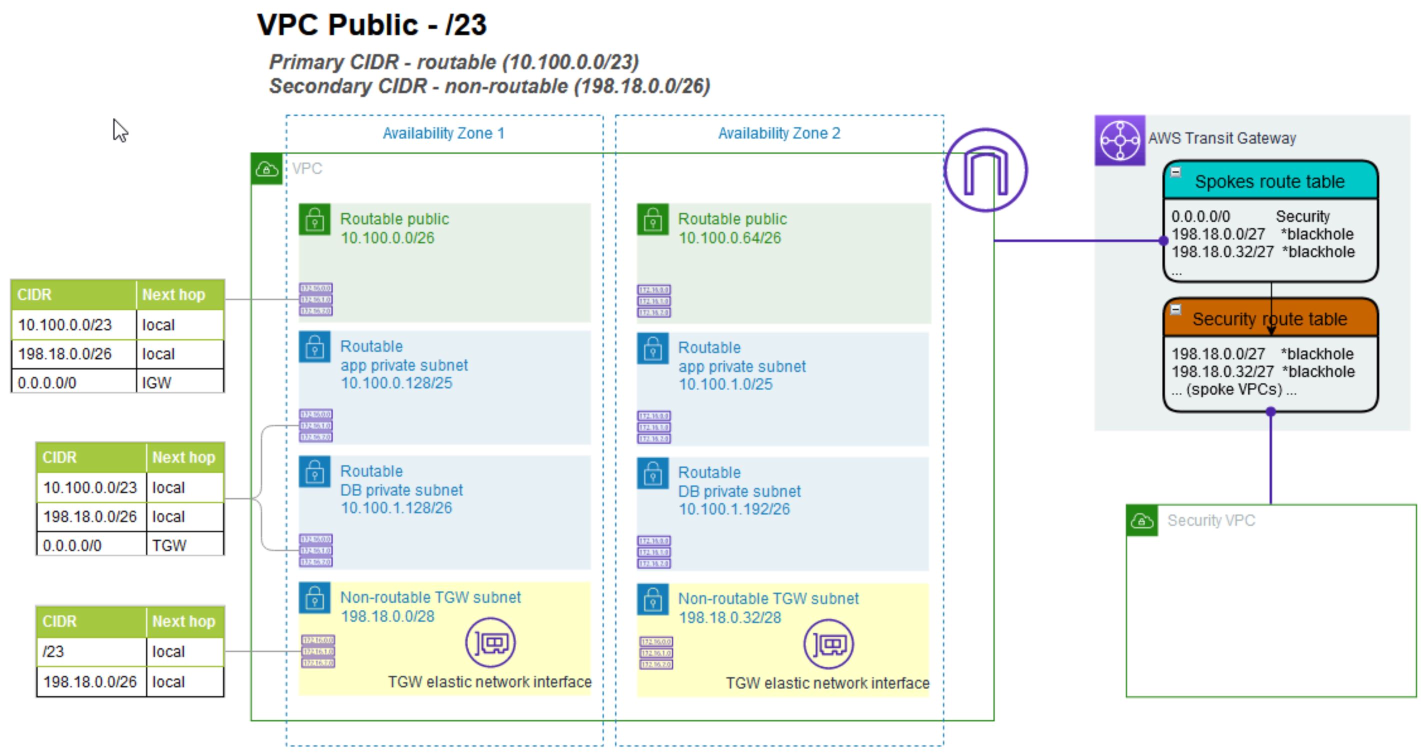
In this example, the /23 routable CIDR is divided up and fully allocated to routable subnets.



Architecture 2 – TGW-attached VPC

The following diagram represents another reference architecture for a VPC that spans two Availability Zones. A TGW attachment supports outbound traffic (egress) from the private subnets to a separate VPC. It uses a non-routable CIDR range only for the TGW attachments subnet. In the TGW routing table, this non-routable CIDR is configured with a blackhole route by using a set of more specific routes. If the routes were to get propagated to the TGW routing table, these more specific blackhole routes would apply.

In this example, the /23 routable CIDR is divided up and fully allocated to routable subnets.



Tools

AWS services and resources

- [Amazon Virtual Private Cloud (Amazon VPC)](#) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS. In this pattern, VPC secondary CIDRs are used to preserve routable IP space in workload CIDRs.
- [Internet gateway ingress routing](#) (edge associations) can be used along with Gateway Load Balancer endpoints for dedicated non-routable subnets.
- [AWS Transit Gateway](#) is a central hub that connects VPCs and on-premises networks. In this pattern, VPCs are centrally attached to a transit gateway, and the transit gateway attachments are in a dedicated non-routable subnet.
- [Gateway Load Balancers](#) help you deploy, scale, and manage virtual appliances, such as firewalls, intrusion detection and prevention systems, and deep packet inspection systems. The gateway serves as a single entry and exit point for all traffic. In this pattern, endpoints for a Gateway Load Balancer can be used in a dedicated non-routable subnet.

- [AWS Network Firewall](#) is a stateful, managed, network firewall and intrusion detection and prevention service for VPCs in the AWS Cloud. In this pattern, endpoints for an firewall can be used in a dedicated non-routable subnet.

Code repository

A runbook and AWS CloudFormation templates for this pattern are available in the GitHub Non-Routable Secondary CIDR Patterns repository. You can use the sample files to set up a working lab in your environment.

Best practices

AWS Transit Gateway

- Use a separate subnet for each transit gateway VPC attachment.
- Allocate a /28 subnet from the secondary non-routable CIDR range for the transit gateway attachment subnets.
- In each transit gateway routing table, add a static, more specific route for the non-routable CIDR range as a blackhole.

Gateway Load Balancer and ingress routing

- Use ingress routing to direct traffic from the internet to the Gateway Load Balancer endpoints.
- Use a separate subnet for each Gateway Load Balancer endpoint.
- Allocate a /28 subnet from the secondary non-routable CIDR range for the Gateway Load Balancer endpoint subnets.

Epics

Create VPCs

Task	Description	Skills required
Determine non-routable CIDR range.	Determine a non-routable CIDR range that will be used for the transit gateway attachment subnet and	Cloud architect

Task	Description	Skills required
	(optionally) for any Gateway Load Balancer or Network Firewall endpoint subnets. This CIDR range will be used as the secondary CIDR for the VPC. It must **not be routable** from the VPC's primary CIDR range or the larger network.	
Determine routable CIDR ranges for VPCs.	Determine a set of routable CIDR ranges that will be used for your VPCs. This CIDR range will be used as the primary CIDR for your VPCs.	Cloud architect
Create VPCs.	Create your VPCs and attach them to the transit gateway. Each VPC should have a primary CIDR range that is routable and a secondary CIDR range that is non-routa ble, based on the ranges you determined in the previous two steps.	Cloud architect

Configure Transit Gateway blackhole routes

Task	Description	Skills required
Create more specific non-routable CIDRs as blackholes.	Each transit gateway routing table needs to have a set of blackhole routes created for the non-routable CIDRs. These are configured to ensure that any traffic from	Cloud architect

Task	Description	Skills required
	the secondary VPC CIDR remains non-routable and doesn't leak into the larger network. These routes should be more specific than the non-routable CIDR that is set as the secondary CIDR on the VPC. For example, if the secondary non-routa ble CIDR is 100.64.0.0/26, the blackhole routes in the transit gateway routing table should be 100.64.0.0/27 and 100.64.0.32/27.	

Related resources

- Best practices for deploying Gateway Load Balancer
- Distributed Inspection Architectures with Gateway Load Balancer
- Networking Immersion Day – Internet to VPC Firewall Lab
- Transit gateway design best practices

Additional information

The non-routable secondary CIDR range can also be useful when working with larger scaled container deployments that require a large set of IP addresses. You can use this pattern with a private NAT Gateway to use a non-routable subnet to host your container deployments. For more information, see the blog post How to solve Private IP exhaustion with Private NAT Solution.

Provision a Terraform product in AWS Service Catalog by using a code repository

Dr. Rahul Sharad Gaikwad and Tamilselvan P, Amazon Web Services

Summary

AWS Service Catalog supports self-service provisioning with governance for your HashiCorp Terraform configurations. If you use Terraform, you can use Service Catalog as the single tool to organize, govern, and distribute your Terraform configurations within AWS at scale. You can access Service Catalog key features, including cataloging of standardized and pre-approved infrastructure as code (IaC) templates, access control, cloud resources provisioning with least privilege access, versioning, sharing to thousands of AWS accounts, and tagging. End users, such as engineers, database administrators, and data scientists, see a list of products and versions they have access to, and they can deploy them through a single action.

This pattern helps you deploy AWS resources by using Terraform code. The Terraform code in the GitHub repository is accessed through Service Catalog. Using this approach, you integrate the products with your existing Terraform workflows. Administrators can create Service Catalog portfolios and add AWS Launch Wizard products to them by using Terraform.

The following are the benefits of this solution:

- Because of the rollback feature in Service Catalog, if any issues occur during deployment, you can revert the product to a previous version.
- You can easily identify the differences between product versions. This helps you resolve issues during deployment.
- You can configure a repository connection in Service Catalog, such as to GitHub or GitLab. You can make product changes directly through the repository.

For information about the overall benefits of AWS Service Catalog, see What is Service Catalog.

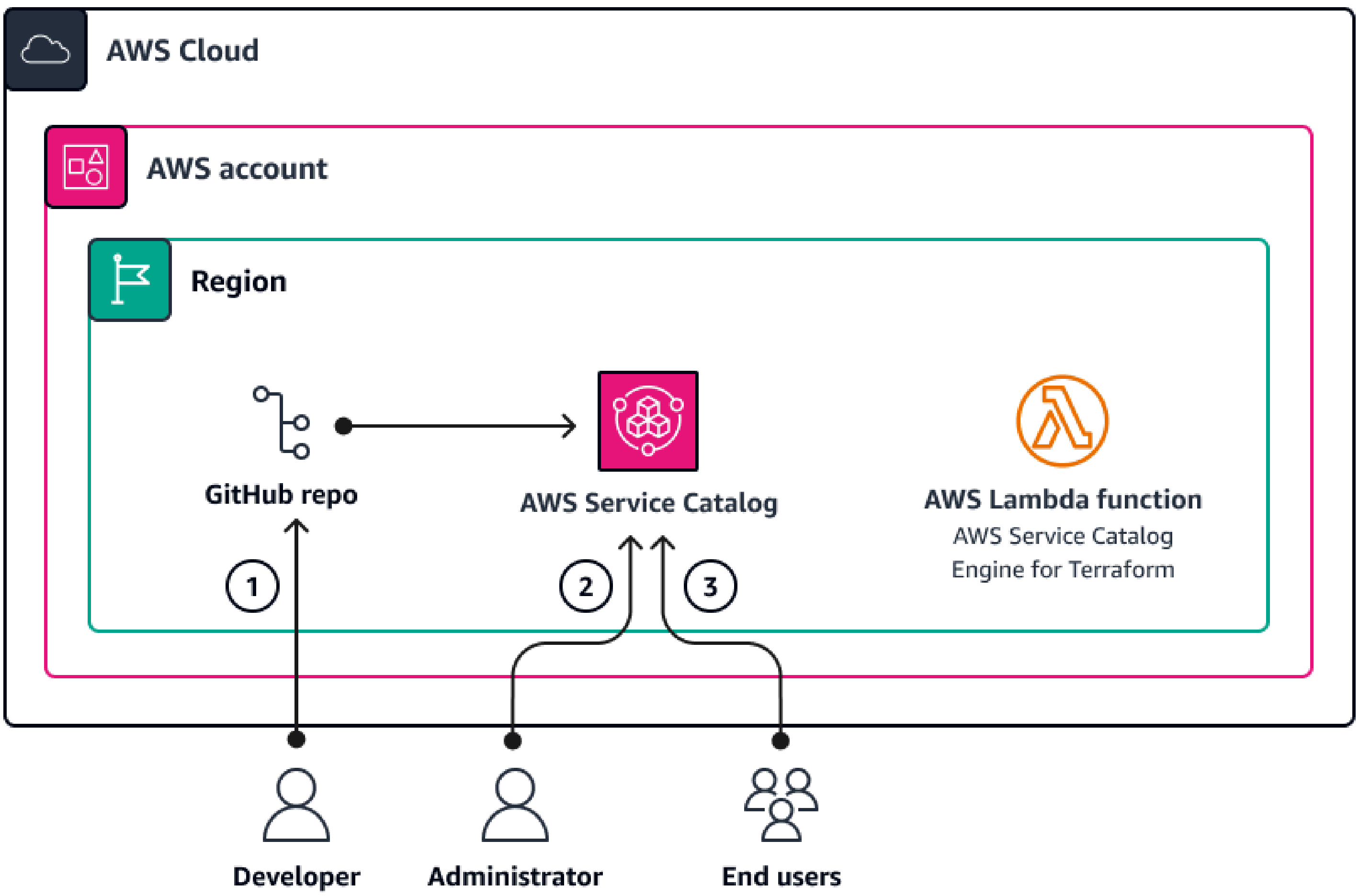
Prerequisites and limitations

Prerequisites

- An active AWS account.
- A GitHub, BitBucket, or other repository that contains Terraform configuration files in ZIP format.

- AWS Serverless Application Model Command Line Interface (AWS SAM CLI), installed.
- AWS Command Line Interface (AWS CLI), installed and configured.
- Go, installed.
- Python version 3.9 , installed. AWS SAM CLI requires this version of Python.
- Permissions to write and run AWS Lambda functions and permissions to access and manage Service Catalog products and portfolios.

Architecture



The diagram shows the following workflow:

1. When a Terraform configuration is ready, a developer creates a .zip file that contains all of the Terraform code. The developer uploads the .zip file into the code repository that is connected to Service Catalog.

2. An administrator associates the Terraform product to a portfolio in Service Catalog. The administrator also creates a launch constraint that allows end users to provision the product.
3. In Service Catalog, end users launch AWS resources by using the Terraform configuration. They can choose which product version to deploy.

Tools

AWS services

- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Service Catalog helps you centrally manage catalogs of IT services that are approved for AWS. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.

Other services

- Go is an open source programming language that Google supports.
- Python is a general-purpose computer programming language.

Code repository

If you require sample Terraform configurations that you can deploy through Service Catalog, you can use the configurations in the GitHub Amazon Macie Organization Setup Using Terraform repository. Use of the code samples in this repository is not required.

Best practices

- Instead of providing the values for variables in the Terraform configuration file (`terraform.tfvars`), configure variable values when launching product through Service Catalog.
- Grant access to the portfolio only to specific users or administrators.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the AWS Identity and Access Management (IAM) documentation.

Epics

Set up your local workstation

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>(Optional) Install Docker.</td><td>If you want to run the AWS Lambda functions in your development environment, install Docker. For instructions, see Install Docker Engine in the Docker documentation.</td><td>DevOps engineer</td></tr>
<tr><td>Install the AWS Service Catalog Engine for Terraform.</td><td>1. Enter the following command to clone the AWS Service Catalog Engine for Terraform repository.
<pre>git clone https://github.com/aws-samples/service-catalog-engine-for-terraform-os.git</pre>2. Navigate to the the root directory of the cloned repository.
3. Enter the following command. This installs the engine.
<pre>run ./bin/bash/deploy-tre.sh -r</pre>The AWS Region set in your default profile is not used during the automated</td><td>DevOps engineer, AWS administrator</td></tr>
</table>

Task	Description	Skills required
	installation. Instead, you provide the Region when you run this command.	

Connect the GitHub repository

Task	Description	Skills required
Create a connection to the GitHub repository.	1. Sign in to the AWS Management Console, and then open the **Developer Tools** console. You can access the Developer Tools console by choosing a service such as AWS CodePipeline or AWS CodeDeploy. 2. In the left navigation pane, choose **Settings**, and then choose **Connections**. 3. Choose **Create connection**. 4. Select the repository where you maintain the Terraform source code. For example, you can choose **Bitbucket**, **GitHub**, or **GitHub Enterprise Server**. 5. Enter a name for the connection, and then choose **Connect**. 6. When you are prompted, authenticate the repositor y.	AWS administrator

Task	Description	Skills required
	After authentication is complete, the connection is created and the status changes to **active**.	

Create a Terraform product in Service Catalog

Task	Description	Skills required
Create the Service Catalog product.	1. Open the [AWS Service Catalog console](https://console.aws.amazon.com/servicecatalog/). 2. Navigate to the **Administration** section, and then choose **Product list**. 3. Choose **Create product**. 4. On the **Create product** page in the **Product details** section, choose the **External** product type. Service Catalog uses this product type to support Terraform Community Edition products. 5. Enter a name and owner for the Service Catalog product. 6. Select **Specify your code repository using a CodeStar provider**. 7. Enter the following information for your repository:	AWS administrator

Task	Description	Skills required
	Connect to your provider using AWS CodeConnections – Select the connection you created previously. **Repository –** Select the repository. **Branch –** Select the branch. **Template file path –** Choose the path where the code template file is stored. The file name should end with `tar.gz`. 8. Under **Version name and description**, provide information about the product version. 9. Choose **Create product**.	

Task	Description	Skills required
Create a portfolio.	1. Open the [AWS Service Catalog console](). 2. Navigate to the **Administr ation** section, and then choose, choose **Portfolios**. 3. Choose **Create portfolio**. 4. Enter the following values: • **Portfolio name** – `Sample terraform` • **Portfolio description** – `Sample portfolio for Terraform configurations` • **Owner** – Your contact information, such as an email address 5. Choose **Create**.	AWS administrator

Task	Description	Skills required
Add the Terraform product to the portfolio.	1. Open the [AWS Service Catalog console](). 2. Navigate to the **Administr ation** section, and then choose **Product list**. 3. Select the Terraform product that you created previously. 4. Choose **Actions**, and then choose **Add product to portfolio**. 5. Choose the `Sample terraform` portfolio. 6. Choose **Add product to portfolio**.	AWS administrator

Task	Description	Skills required
Create the access policy.	1. Open the [AWS Identity and Access Management (IAM) console](). 2. In the navigation pane, choose **Policies**. 3. In the content pane, choose **Create policy**. 4. Choose the **JSON** option. 5. Enter the sample JSON policy in *Access policy* in the [Additional information]() section of this pattern. 6. Choose **Next**. 7. On the **Review and create** page, in the **Policy name** box, enter `TerraformResourceCreationAndArtifactAccessPolicy` . 8. Choose **Create policy**.	AWS administrator

Task	Description	Skills required
Create a custom trust policy.	1. Open the IAM console. 2. In the navigation pane, choose **Roles**. 3. Choose **Create role**. 4. Under **Trusted entity type**, choose **Custom trust policy**. 5. In the JSON policy editor, enter the sample JSON policy in *Trust policy* in the Additional information section of this pattern. 6. Choose **Next**. 7. Under **Permissions policies**, choose the `TerraformResourceCreationAndArtifactAccessPolicy` that you previously created. 8. Choose **Next**. 9. Under **Role details**, in the **Role name** box, enter `SCLaunch-product` . **Important** The role name must begin with `SCLaunch`. 10. Choose **Create role**.	AWS administrator

Task	Description	Skills required
Add a launch constraint to the Service Catalog product.	1. Sign in to the AWS Management Console as a user with administrative permissions. 2. Open the [AWS Service Catalog console](https://console.aws.amazon.com/servicecatalog/). 3. In the navigation pane, choose **Portfolios**. 4. Choose the portfolio that you created previously. 5. On the **Portfolio details** page, choose the **Constraints** tab, and then choose **Create constraint**. 6. For **Product**, select the Terraform product you created previously. 7. Under **Launch constraint**, for **Method**, choose **Enter role name**. 8. In the **Role name** box, enter `SCLaunch-product` . 9. Choose **Create**.	AWS administrator

Task	Description	Skills required
Grant access to the product.	1. Open the [AWS Service Catalog console](). 2. In the navigation pane, choose **Portfolios**. 3. Choose the portfolio that you created previously. 4. Choose the **Access** tab, and then choose **Grant access**. 5. Choose the **Roles** tab, and then select the role that should have access to deploy this product. 6. Choose **Grant Access**.	AWS administrator
Launch the product.	1. Sign in to the AWS Management Console as a user with permissions to deploy the Service Catalog product. 2. Open the [AWS Service Catalog console](). 3. In the navigation pane, choose **Products**. 4. Choose the produce you created previously, and then choose **Launch product**. 5. Enter a product name and define any required parameters. 6. Choose **Launch product**.	DevOps engineer

Verify the deployment

Task	Description	Skills required
Validate the deployment.	There are two AWS Step Functions state machines for the Service Catalog provisioning workflow: • `ManageProvisionedProductStateMachine` – Service Catalog invokes this state machine when provisioning a new Terraform product and when updating an existing Terraform provisioned product. • `TerminateProvisionedProductStateMachine` – Service Catalog invokes this state machine when terminating an existing Terraform provisioned product. You check the logs for the `ManageProvisionedProductStateMachine` state machine to confirm that the product was provisioned. 1. Sign in to the AWS Management Console, and then open the [AWS Step Functions console](https://console.aws.amazon.com/states/home).	DevOps engineer

Task	Description	Skills required
	2. In the left navigation pane, choose **State machines**. 3. Choose `ManageProvisionedProductStateMachine` . 4. In the **Executions** list, enter the provisioned product ID to locate the execution. **Note** The state file backend bucket names start with `sc-terraform-engine-state-`. 5. Validate that all the required resources have been created in the account.	

Clean up infrastructure

Task	Description	Skills required
Delete provisioned products.	1. Sign in to the AWS Management Console as a user with permissions to deploy the Service Catalog product. 2. Open the [AWS Service Catalog console]().	DevOps engineer

Task	Description	Skills required
	3. In the left navigation, choose **Provisioned products**. 4. Select the product you created. 5. In the **Actions** list, choose **Terminate**. 6. In the confirmation text box, enter `terminate`, and then choose **Terminate provisioned product**. 7. Repeat these steps to terminate all provisioned products.	

Task	Description	Skills required
Remove the AWS Service Catalog Engine for Terraform.	1. Sign in to the AWS Management Console as a user with administrative permissions. 2. Open the [Amazon Simple Storage Service (Amazon S3) console](https://console.aws.amazon.com/s3/). 3. In the navigation pane, choose **Buckets**. 4. Select the `sc-terraform-engine-logging-XXXX` bucket. 5. Choose **Empty**. 6. Repeat steps 4–5 for the following buckets: • `sc-terraform-engine-state-XXXX` • `terraform-engine-bootstrap-XXXX` 7. Open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/), and then validate you're in the correct AWS Region. 8. In the left navigation, choose **Stacks**. 9. Select `SAM-TRE`, and then choose **Delete**. Wait until the stack has been deleted. 10. Select `Bootstrap-TRE`, and then choose **Delete**. Wait until the stack has been deleted.	AWS administrator

Related resources

AWS documentation

- Getting started with a Terraform product

Terraform documentation

- Terraform installation
- Terraform backend configuration
- Terraform AWS Provider documentation

Additional information

Access policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/servicecatalog:provisioning": "true"
                }
            }
        },
        {
            "Action": [
                "s3:CreateBucket*",
                "s3:DeleteBucket*",
                "s3:Get*",
                "s3:List*",
                "s3:PutBucketTagging"
            ],
            "Resource": "arn:aws:s3:::*",
            "Effect": "Allow"
        },
```

```
        {
            "Action": [
                "resource-groups:CreateGroup",
                "resource-groups:ListGroupResources",
                "resource-groups:DeleteGroup",
                "resource-groups:Tag"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "tag:GetResources",
                "tag:GetTagKeys",
                "tag:GetTagValues",
                "tag:TagResources",
                "tag:UntagResources"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

Trust policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GivePermissionsToServiceCatalog",
            "Effect": "Allow",
            "Principal": {
                "Service": "servicecatalog.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::account_id:root"
            },
            "Action": "sts:AssumeRole",
```

```
            "Condition": {
                "StringLike": {
                    "aws:PrincipalArn": [
                        "arn:aws:iam::accounti_id:role/TerraformEngine/
TerraformExecutionRole*",
                        "arn:aws:iam::accounti_id:role/TerraformEngine/
ServiceCatalogExternalParameterParserRole*",
                        "arn:aws:iam::accounti_id:role/TerraformEngine/
ServiceCatalogTerraformOSParameterParserRole*"
                    ]
                }
            }
        }
    ]
}
```

Register multiple AWS accounts with a single email address by using Amazon SES

Joe Wozniak and Shubhangi Vishwakarma, Amazon Web Services

Summary

This pattern describes how you can decouple real email addresses from the email address that's associated with an AWS account. AWS accounts require a unique email address to be provided at the time of account creation. In some organizations, the team that manages AWS accounts must take on the burden of managing many unique email addresses with their messaging team. This can be difficult for large organizations that manage many AWS accounts. Additionally, if your email system doesn't allow *plus addressing* or *sub-addressing* as defined in Sieve Email Filtering: Subaddress Extension (RFC 5233)—by adding a plus sign (+) and an identifier to the end of the local part of the email address, such as `admin+123456789123@example.com`—this pattern can help overcome this limitation.

This pattern provides a unique email address vending solution that enables AWS account owners to associate one email address with multiple AWS accounts. The real email addresses of AWS account owners are then associated with these generated email addresses in a table. The solution handles all incoming email for the unique email accounts, looks up the owner of each account, and then forwards any received messages to the owner.

Prerequisites and limitations

Prerequisites

- Administrative access to an AWS account.
- Access to a development environment.
- (Optional) Familiarity with AWS Cloud Development Kit (AWS CDK) workflows and the Python programming language will help you troubleshoot any issues or make modifications.

Limitations

- Overall vended email address length of 64 characters. For details, see CreateAccount in the *AWS Organizations API reference*.

Product versions

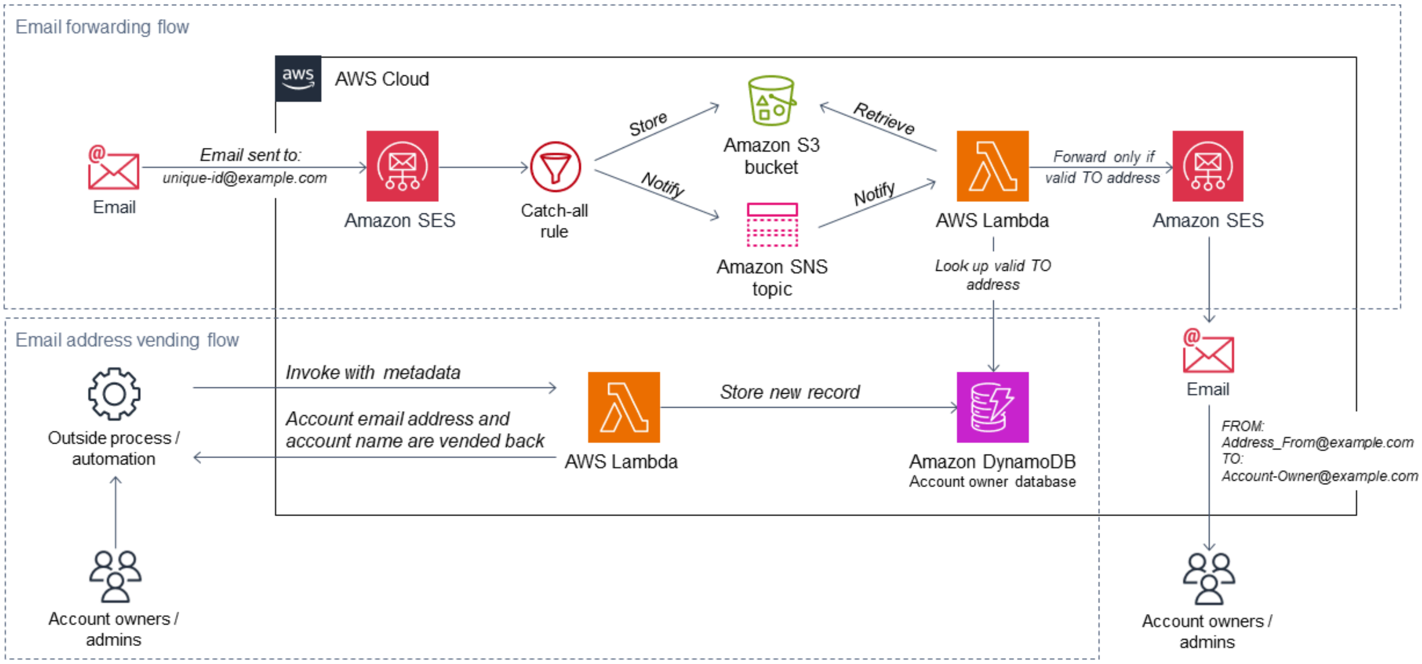
- Node.js version 22.x or later
- Python 3.13 or later
- Python packages **pip** and **virtualenv**
- AWS CDK CLI version 2.1019.2 or later
- Docker 20.10.x or later

Architecture

Target technology stack

- AWS CloudFormation stack
- AWS Lambda functions
- Amazon Simple Email Service (Amazon SES) rule and rule set
- AWS Identity and Access Management (IAM) roles and policies
- Amazon Simple Storage Service (Amazon S3) bucket and bucket policy
- AWS Key Management Service (AWS KMS) key and key policy
- Amazon Simple Notification Service (Amazon SNS) topic and topic policy
- Amazon DynamoDB table

Target architecture



This diagram shows two flows:

- **Email address vending flow:** In the diagram, the email address vending flow (lower section) begins typically with an account vending solution or outside automation, or is invoked manually. In the request, a Lambda function is called with a payload that contains the needed metadata. The function uses this information to generate a unique account name and email address, stores it in a DynamoDB database, and returns the values to the caller. These values can then be used to create a new AWS account (typically by using AWS Organizations).
- **Email forwarding flow:** This flow is illustrated in the upper section of the previous diagram. When an AWS account is created by using the account email generated from the email address vending flow, AWS sends various emails, such as account registration confirmation and periodic notifications, to that email address. By following the steps in this pattern, you configure your AWS account with Amazon SES to receive emails for the entire domain. This solution configures forwarding rules that allow Lambda to process all incoming emails, check to see if the TO address is in the DynamoDB table, and forward the message to the account owner's email address instead. Using this process gives account owners the ability to associate multiple accounts with one email address.

Automation and scale

This pattern uses the AWS CDK to fully automate the deployment. The solution uses AWS managed services that will (or can be configured to) scale automatically to meet your needs. The Lambda functions might require additional configuration to meet your scaling needs. For more information, see Understanding Lambda function scaling in the Lambda documentation.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Email Service (Amazon SES) helps you send and receive emails by using your own email addresses and domains.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Tools needed for deployment

- Development environment with the AWS CLI and IAM access to your AWS account. For details, see the links in the Related resources section.
- On your development system, install the following:
  - Git command line tool, available from the Git downloads website.

- The AWS CLI to configure access credentials for the AWS CDK. For more information, see the AWS CLI documentation.
- Python version 3.13 or later, available from the Python downloads website.
- UV for Python package management. For installation instructions, see the UV installation guide.
- Node.js version 22.x or later. For installation instructions, see the Node.js documentation.
- AWS CDK CLI version 2.1019.2 or later. For installation instructions, see the AWS CDK documentation.
- Docker version 20.10.x or later. For installation instructions, see the Docker documentation.

Code

The code for this pattern is available in the GitHub AWS account factory email repository.

Epics

Allocate a target deployment environment

Task	Description	Skills required
Identify or create an AWS account.	Identify an existing or new AWS account to which you have full administrative access, to deploy the email solution.	AWS administrator, Cloud administrator
Set up a deployment environment.	Configure an easy to use deployment environment and set up dependencies by following these steps: 1. Set up your development environment with the tools listed in the Tools section. 2. Clone the GitHub AWS account factory email repository code base	AWS DevOps, App developer

Task	Description	Skills required

	to your development environment by using the command:
	`git clone https://github.com/aws-samples/aws-account-factory-email`
	3. In the `requirements.in` file (in the root of the repository), update the line that starts with `aws-cdk-lib==` to match the latest version of the AWS CDK. To identify what the latest version is, run `npm view aws-cdk-lib` and look for the keyword "latest:".
	4. If you made an update to the version in `requirements.in`, run the following command to update the `requirements.txt` file by using UV:
	`uv pip compile requirements.in -o requirements.txt --python-version 3.13`	

Set up a verified domain

Task	Description	Skills required
Identify and allocate a domain.	The email forwarding functionality requires a dedicated domain. Identify and allocate a domain or subdomain that you can verify with Amazon SES. This domain should be available to receive incoming email within the AWS account where the email forwarding solution is deployed. Domain requirements: • The domain should be a standard domain or subdomain. • The domain should be externally DNS resolvable, because it will be used to receive emails from outside the organization.	Cloud administrator, Network administrator, DNS administr ator
Verify the domain.	Verify that the identified domain can be used to accept incoming email. Complete the instructions in [Verifying your domain for Amazon SES email receiving](#) in the Amazon SES documentation. This will require coordination with the person or team who is	App developer, AWS DevOps

Task	Description	Skills required
	responsible for the domain's DNS records.	
Set up MX records.	Set up your domain with MX records that point to the Amazon SES endpoints in your AWS account and Region. For more informati on, see [Publishing an MX record for Amazon SES email receiving](#) in the Amazon SES documentation.	Cloud administrator, Network administrator, DNS administr ator

Deploy the email vending and forwarding solution

Task	Description	Skills required
Modify the default values in `cdk.json`.	Edit some of the default values in the `cdk.json` file (in the root of the repositor y) so that the solution will operate correctly after it is deployed. 1. Modify the `SES_DOMAIN_NAME` value to match the domain name you verified previously. 2. Modify the `ADDRESS_FROM` value to include the same domain that is in `SES_DOMAIN_NAME`. The local part of the address should be determined by your cloud team. This	App developer, AWS DevOps

Task	Description	Skills required
	address becomes the `FROM` address for every email that's forwarded through the solution. 3. Modify the `ADDRESS_ADMIN` value to match the email address that any non-matching incoming messages will be forwarded to. This value must be a valid and operating email address.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy the email vending and forwarding solution.</td><td>
1. Create a Python virtual environment:
<pre>uv venv --python 3.13</pre>
2. Activate the Python virtual environment:
<pre>source .venv/bin/
activate</pre>
Or, on the Windows platform, use:
<pre>% .venv\Scripts\acti
vate.bat</pre>
3. Install all Python requirements with no errors:
<pre>uv pip install -r
 requirements.txt</pre>
4. Synthesize the CloudFormation template:
<pre>cdk synth</pre>
Confirm that there are no errors and that the full CloudFormation template contains the expected output.
5. (Optional) If you are deploying the AWS CDK
</td><td>App developer, AWS DevOps</td></tr>
</table>

Task	Description	Skills required

	code into the current AWS account or Region for the first time, bootstrap the environment. For more information, see [AWS CDK Bootstrapping](#) in the AWS CDK documentation.
	`cdk bootstrap aws://AWS-ACCOUNT-NUMBER/REGION`
	Replace `AWS-ACCOUNT-NUMBER` and `REGION` with actual values.
	6. Deploy the solution:
	`cdk bootstrap`
	`cdk deploy`
	The commands should complete without errors.	

Task	Description	Skills required
Verify that the solution has been deployed.	Verify that the solution deployed successfully before you begin testing: 1. Open the [AWS CloudFormation console](#) and look for a CloudFormation stack that contains the name `AwsMailFwdStack` . 2. Confirm that this `AwsMailFwdStack` stack has the following resources : • Lambda functions • Amazon SES rule and rule set • IAM roles and policies • Amazon S3 bucket and bucket policy • AWS KMS key and key policy • Amazon SNS topic and topic policy • DynamoDB table	App developer, AWS DevOps

Verify that email vending and forwarding operate as expected

Task	Description	Skills required
Verify that the API is working.	In this step, you submit test data to the solution's API and confirm that the solution produces the expected output	App developer, AWS DevOps

Task	Description	Skills required
	and that backend operation s have been performed as expected. Manually run the **Vend Email** Lambda function by using test input. (For an example, see the sample_vend_reques t.json file.) For `OwnerAddr ess` , use a valid email address. The API should return an account name and account email with values as expected.	

Task	Description	Skills required
Verify that email is being forwarded.	In this step, you send a test email through the system and verify that the email is forwarded to the expected recipient. 1. Obtain the account email from the last step. 2. Send an email to this address with a test subject and body text. 3. Confirm that you received the email at the account owner's email address. 4. Confirm that the email you received has a `FROM` address that matches the `ADDRESS_FROM` setting in `cdk.json`. 5. Confirm that the received email's subject and body are the same as the original sent message.	App developer, AWS DevOps

Troubleshooting

Issue	Solution
The system doesn’t forward email as expected.	Verify that your setup is correct: 1. You should have completed the [Amazon SES verification process]() for your domain.

Issue	Solution
	2. Your domain should be set up properly with MX records pointing to the Amazon SES endpoints in your AWS account and Region. For more information, see [Publishing an MX record for Amazon SES email receiving]() in the Amazon SES documentation. After you verify your domain setup, follow these steps: 1. Open the [Amazon CloudWatch console]() for the account and Region where you deployed the solution, and navigate to CloudWatch log groups in the navigation pane. 2. Search the list of log groups for `SesMailForwardLogGroup` . 3. Investigate the logs in this group to see if any errors are generated during the email vending and forwarding process.

Issue	Solution
When you try to deploy the AWS CDK stack, you receive an error similar to: "Template format error: Unrecognized resource types"	In most instances, this error message means that the Region you're targeting doesn't have all the available AWS services. If you're using an Amazon EC2 instance to deploy the solution, you might be targeting a Region that is different from the Region where the instance is running. **Note** By default, the AWS CDK deploys to the Region and account that you configured in the AWS CLI. Possible solutions: 1. Investigate if all the services needed for this solution (see the Target technology stack section earlier in this pattern) are in the AWS Region you are targeting by reviewing AWS Services by Region. 2. If you are using an EC2 instance and targeting a Region that's different from the Region where your instance is running, make sure to set the `AWS_DEFAULT_REGION` environment variable, or set a Region with the AWS CLI before you deploy the solution. For more information, see the Configuring environment variables for the AWS CLI in the AWS CLI documentation. Alternatively, you can modify the `app.py` file in the root of the repository to include a hard-coded account ID and Region

Issue	Solution
	by following the instructions in the [AWS CDK documentation for environments]().
When you deploy the solution, you receive the error message: "Deployment failed: Error: AwsMailFwdStack: SSM parameter /cdk-bootstrap/hnb659fds/version not found. Has the environment been bootstrapped? Please run 'cdk bootstrap'"	If you have never deployed any AWS CDK resources to the AWS account and Region you're targeting, you will have to first run the `cdk bootstrap` command as the error indicates. If you continue to receive this error after you run the bootstrapping command, you might be trying to deploy the solution to a Region that's different from the Region where your development environment is running. To solve this problem, set the `AWS_DEFAULT_REGION` environment variable or set a Region with the AWS CLI before you deploy the solution. Alternatively, you can modify the `app.py` file in the root of the repository to include a hard-coded account ID and Region by following the instructions in the AWS CDK documentation for environments.

Related resources

- For help installing the AWS CLI, see Installing or updating to the latest version of the AWS CLI.
- For help setting up the AWS CLI with IAM access credentials, see Configuring settings for the AWS CLI.
- For help with the AWS CDK, see Getting started with the AWS CDK.

Additional information

Costs

When you deploy this solution, the AWS account holder might incur costs that are associated with the use of the following services. It is important for you to understand how these services are billed so you are aware of any potential charges. For pricing information, see the following pages:

- Amazon SES pricing
- Amazon S3 pricing
- AWS KMS pricing
- AWS Lambda pricing
- Amazon DynamoDB pricing

Set up DNS resolution for hybrid networks in a single-account AWS environment

Abdullahi Olaoye, Amazon Web Services

Summary

This pattern describes how to set up a fully hybrid Domain Name System (DNS) architecture that enables end-to-end DNS resolution of on-premise resources, AWS resources, and internet DNS queries, without administrative overhead. The pattern describes how to set up Amazon Route 53 Resolver forwarding rules that determine where a DNS query that originates from AWS should be sent, based on the domain name. DNS queries for on-premises resources are forwarded to on-premises DNS resolvers. DNS queries for AWS resources and internet DNS queries are resolved by Route 53 Resolver.

This pattern covers hybrid DNS resolution in an AWS single-account environment. For information about setting up outbound DNS queries in an AWS multi-account environment, see the pattern Set up DNS resolution for hybrid networks in a multi-account AWS environment.

Prerequisites and limitations

Prerequisites

- An AWS account
- A virtual private cloud (VPC) in your AWS account
- A network connection between the on-premises environment and your VPC, through AWS Virtual Private Network (AWS VPN) or AWS Direct Connect
- IP addresses of your on-premises DNS resolvers (reachable from your VPC)
- Domain/subdomain name to forward to on-premises resolvers (for example, onprem.mydc.com)
- Domain/subdomain name for the AWS private hosted zone (for example, myvpc.cloud.com)

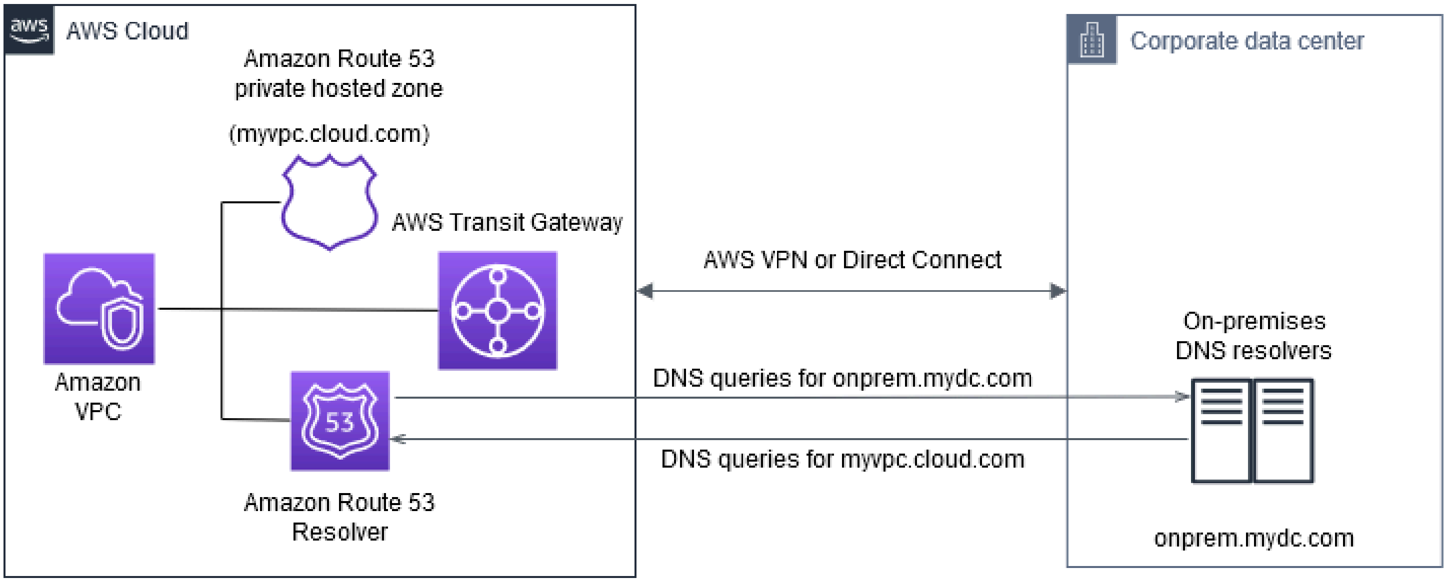
Architecture

Target technology stack

- Amazon Route 53 private hosted zone
- Amazon Route 53 Resolver
- Amazon VPC

- AWS VPN or Direct Connect

Target architecture



Tools

- Amazon Route 53 Resolver makes hybrid cloud easier for enterprise customers by enabling seamless DNS query resolution across your entire hybrid cloud. You can create DNS endpoints and conditional forwarding rules to resolve DNS namespaces between your on-premises data center and your VPCs.
- Amazon Route 53 private hosted zone is a container that holds information about how you want Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs that you create with the Amazon VPC service.

Epics

Configure a private hosted zone

Task	Description	Skills required
Create a Route 53 private hosted zone for an AWS	This zone holds the DNS records for AWS resources that should be resolved from	Network admin, System admin

Task	Description	Skills required
reserved domain name such as myvpc.cloud.com.	the on-premises environment. For instructions, see Creating a private hosted zone in the Route 53 documentation.	
Associate the private hosted zone with your VPC.	To enable resources in your VPC to resolve DNS records in this private hosted zone, you must associate your VPC with the hosted zone. For instructions, see Creating a private hosted zone in the Route 53 documentation.	Network admin, System admin

Set up Route 53 Resolver endpoints

Task	Description	Skills required
Create an inbound endpoint.	Route 53 Resolver uses the inbound endpoint to receive DNS queries from on-premises DNS resolvers. For instructions, see Forwarding inbound DNS queries to your VPCs in the Route 53 documentation. Make a note of the inbound endpoint IP address.	Network admin, System admin
Create an outbound endpoint.	Route 53 Resolver uses the outbound endpoint to send DNS queries to on-premises DNS resolvers. For instructions, see Forwarding outbound DNS queries to your network in the Route 53	Network admin, System admin

Task	Description	Skills required
	documentation. Make a note of the output endpoint ID.	

Set up a forwarding rule and associate it with your VPC

Task	Description	Skills required
Create a forwarding rule for the on-premises domain.	This rule will instruct Route 53 Resolver to forward any DNS queries for on-premises domains (such as onprem.mydc.com) to on-premises DNS resolvers. To create this rule, you will need the IP addresses of the on-premises DNS resolvers and the outbound endpoint ID for Route 53 Resolver. For instructions, see Managing forwarding rules in the Route 53 documentation.	Network admin, System admin
Associate the forwarding rule with your VPC.	For the forwarding rule to take effect, you must associate the rule with your VPC. Route 53 Resolver then takes the rule into consideration when resolving a domain. For instructions, see Managing forwarding rules in the Route 53 documentation.	Network admin, System admin

Configure on-premises DNS resolvers

Task	Description	Skills required
Configure conditional forwarding in the on-premise DNS resolvers.	For DNS queries to be sent to the Route 53 private hosted zone from the on-premises environment, you must configure condition al forwarding in the on-premises DNS resolvers. This instructs the DNS resolvers to forward all DNS queries for the AWS domain (for example, for myvpc.clo ud.com) to the inbound endpoint IP address for Route 53 Resolver.	Network admin, System admin

Test end-to-end DNS resolution

Task	Description	Skills required
Test DNS resolution from AWS to the on-premises environment.	From a server in the VPC, perform a DNS query for an on-premises domain (such as server1.onprem.mydc.com).	Network admin, System admin
Test DNS resolution from the on-premises environment to AWS.	From an on-premises server, perform DNS resolution for an AWS domain (such as server1.myvpc.cloud.com).	Network admin, System admin

Related resources

- Centralized DNS management of hybrid cloud with Amazon Route 53 and AWS Transit Gateway (AWS Networking & Content Delivery blog)
- Simplify DNS management in a multi-account environment with Route 53 Resolver (AWS Security blog)
- Working with private hosted zones (Route 53 documentation)
- Getting started with Route 53 Resolver (Route 53 documentation)

Set up UiPath RPA bots automatically on Amazon EC2 by using AWS CloudFormation

Dr. Rahul Sharad Gaikwad and Tamilselvan P, Amazon Web Services

Summary

This pattern explains how you can deploy robotic process automation (RPA) bots on Amazon Elastic Compute Cloud (Amazon EC2) instances. It uses an EC2 Image Builder pipeline to create a custom Amazon Machine Image (AMI). An AMI is a preconfigured virtual machine (VM) image that contains the operating system (OS) and preinstalled software to deploy EC2 instances. This pattern uses AWS CloudFormation templates to install UiPath Studio Community edition on the custom AMI. UiPath is an RPA tool that helps you set up robots to automate your tasks.

As part of this solution, EC2 Windows instances are launched by using the base AMI, and the UiPath Studio application is installed on the instances. The pattern uses the Microsoft System Preparation (Sysprep) tool to duplicate the customized Windows installation. After that, it removes the host information and creates a final AMI from the instance. You can then launch the instances on demand by using the final AMI with your own naming conventions and monitoring setup.

Note: This pattern doesn't provide any information about using RPA bots. For that information, see the UiPath documentation. You can also use this pattern to set up other RPA bot applicati ons by customizing the installation steps based on your requirements.

This pattern provides the following automations and benefits:

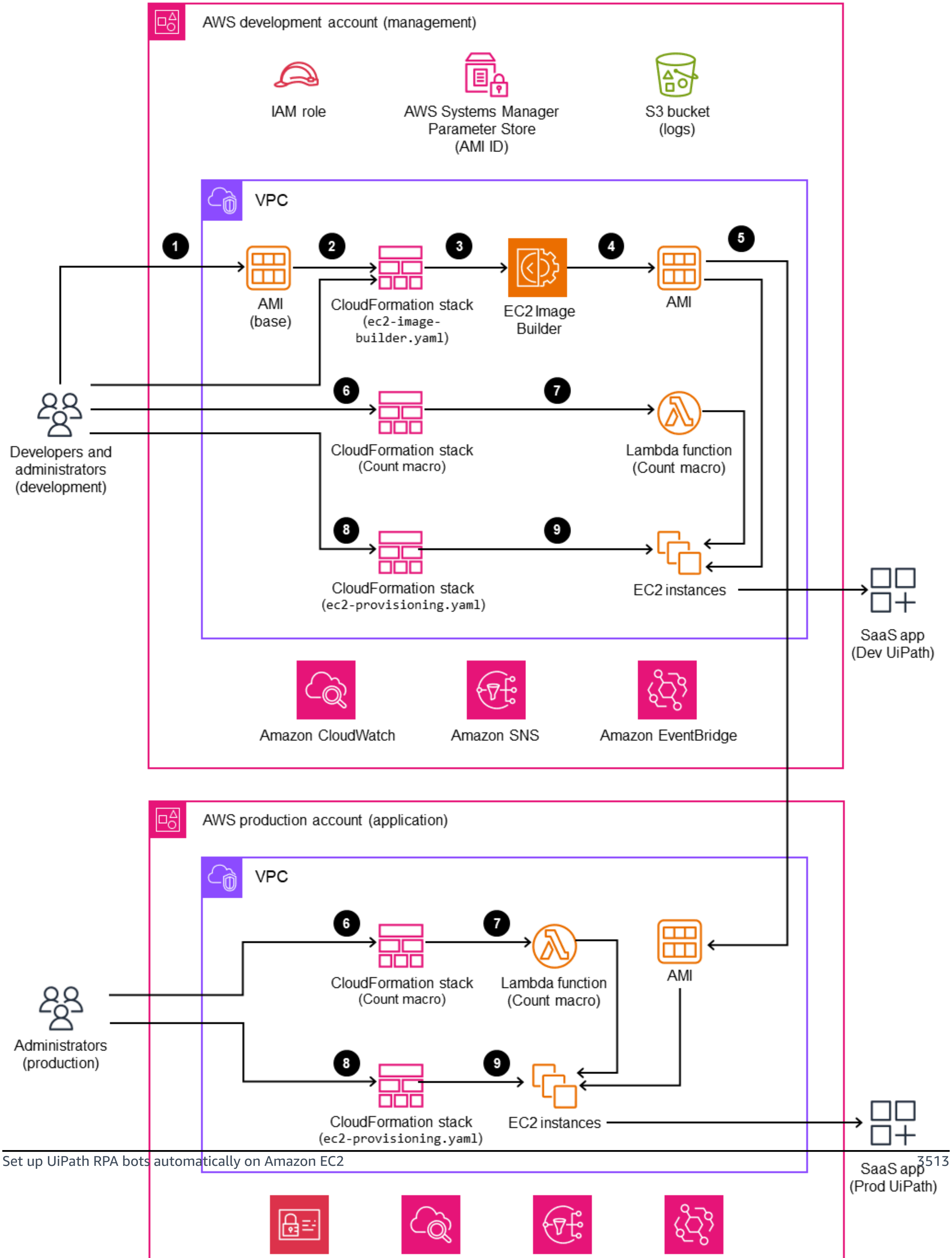
- Application deployment and sharing: You can build Amazon EC2 AMIs for application deployment and share them across multiple accounts through an EC2 Image Builder pipeline, which uses AWS CloudFormation templates as infrastructure as code (IaC) scripts.
- Amazon EC2 provisioning and scaling: CloudFormation IaC templates provide custom computer name sequences and Active Directory join automation.
- Observability and monitoring: The pattern sets up Amazon CloudWatch dashboards to help you monitor Amazon EC2 metrics (such as CPU and disk usage).
- RPA benefits for your business: RPA improves accuracy because robots can perform assigned tasks automatically and consistently. RPA also increases speed and productivity because it removes operations that don't add value and handles repetitious activities.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Identity and Access Management (IAM) permissions for deploying CloudFormation templates
- IAM policies to set up cross-account AMI distribution with EC2 Image Builder

Architecture



1. The administrator provides the base Windows AMI in the `ec2-image-builder.yaml` file and deploys the stack in the CloudFormation console.
2. The CloudFormation stack deploys the EC2 Image Builder pipeline, which includes the following resources:
   - `Ec2ImageInfraConfiguration`
   - `Ec2ImageComponent`
   - `Ec2ImageRecipe`
   - `Ec2AMI`
3. The EC2 Image Builder pipeline launches a temporary Windows EC2 instance by using the base AMI and installs the required components (in this case, UiPath Studio).
4. The EC2 Image Builder removes all the host information and creates an AMI from Windows Server.
5. You update the `ec2-provisioning yaml` file with the custom AMI and launch a number of EC2 instances based on your requirements.
6. You deploy the Count macro by using a CloudFormation template. This macro provides a **Count** property for CloudFormation resources so you can specify multiple resources of the same type easily.
7. You update the name of the macro in the CloudFormation `ec2-provisioning.yaml` file and deploy the stack.
8. The administrator updates the `ec2-provisioning.yaml` file based on requirements and launches the stack.
9. The template deploys EC2 instances with the UiPath Studio application.

Tools

AWS services

- AWS CloudFormation helps you model and manage infrastructure resources in an automated and secure manner.
- Amazon CloudWatch helps you observe and monitor resources and applications on AWS, on premises, and on other clouds.
- Amazon Elastic Compute Cloud (Amazon EC2) provides secure and resizable compute capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.

- EC2 Image Builder simplifies the building, testing, and deployment of virtual machines and container images for use on AWS or on premises.
- Amazon EventBridge helps you build event-driven applications at scale across AWS, existing systems, or software as a service (SaaS) applications.
- AWS Identity and Access Management (IAM) helps you securely control access to AWS resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can call Lambda functions from over 200 AWS services and SaaS applications, and pay only for what you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data..
- AWS Systems Manager Agent (SSM Agent) helps Systems Manager update, manage, and configure EC2 instances, edge devices, on-premises servers, and virtual machines (VMs).

Code repositories

The code for this pattern is available in the GitHub UiPath RPA bot setup using CloudFormation repository. The pattern also uses a macro that's available from the AWS CloudFormation Macros repository.

Best practices

- AWS releases new Windows AMIs each month. These contain the latest OS patches, drivers, and launch agents. We recommend that you use the latest AMI when you launch new instances or when you build your own custom images.
- Apply all available Windows or Linux security patches during image builds.

Epics

Deploy an image pipeline for the base image

Task	Description	Skills required
Set up an EC2 Image Builder pipeline.	1. Clone the UiPath RPA bot setup using CloudFormation repository, or download the `ec2-image-builder.yaml` template from the repository. 2. Sign in to the AWS Management Console, and open the AWS CloudFormation console. 3. Choose **Create stack**. 4. In the **Specify template** section, choose **Upload a template file**. 5. Locate and upload the `ec2-image-builder.yaml` template from your computer, and then choose **Next**. 6. Provide input parameters for your stack or accept the default values. Choose **Next**. **Note** The number and values of parameters might	AWS DevOps

Task	Description	Skills required
	vary depending on your input values. 7. Optionally, configure stack options, and then choose **Next**. 8. Review your stack details. 9. At the end of the screen, select the check box to acknowledge capabilities, and then choose **Submit**. 10. Monitor the stack's progress. When the status is `CREATE_COMPLETE` , the deployment is ready.	

Task	Description	Skills required
View EC2 Image Builder settings.	The EC2 Image Builder settings include infrastru cture configuration, distribut ion settings, and security scanning settings. To view the settings: 1. Open the EC2 Image Builder console. 2. From the navigation pane, navigate to various Image Builder settings. **Note** As a best practice, you should make any updates to EC2 Image Builder through the CloudFormation template only.	AWS DevOps

Task	Description	Skills required
View the image pipeline.	To view the deployed image pipeline: 1. On the EC2 Image Builder console, choose **Image pipelines** from the navigation pane. 2. Select the image pipeline you created. 3. View the configuration details of the output images, image recipe, infrastructure configura tion, distribution settings, Amazon EventBridge rules, and tags.	AWS DevOps

Task	Description	Skills required
View Image Builder logs.	EC2 Image Builder logs are aggregated in CloudWatch log groups. To view the logs in CloudWatch: 1. Open the [CloudWatch console](https://console.aws.amazon.com/cloudwatch/). 2. In the navigation pane, choose **Logs**, **Log groups**. 3. Choose the log group name. EC2 Image Builder logs are aggregated in the log group `/aws/imagebuilder/XXX` . 4. Check the latest logs in the respective log stream for any errors encountered when running the image pipeline. EC2 Image Builder logs are also stored in an S3 bucket. To view the logs in the bucket: 1. Open the [Amazon S3 console](https://console.aws.amazon.com/s3/). 2. In the **Buckets** list, choose the bucket name. The logs are aggregated in the S3 bucket `<stack-name>-XXXXXX` .	AWS DevOps

Task	Description	Skills required
Upload the UiPath file to an S3 bucket.	1. Download the `.msi` file for UiPath Studio from the location https://download.uipath.com/UiPathStudioCommunity.msi. 2. Upload the file to an S3 bucket. 3. Update the bucket name and file key in the `ec2-image-builder.yaml` template, in the user data section, line number 310.	AWS DevOps

Deploy and test the Count macro

Task	Description	Skills required
Deploy the Count macro.	1. Clone or download the Count CloudFormation macro. 2. Navigate to the `Count` folder. 3. You will need an S3 bucket to store the CloudFormation artifacts. If you don't have an S3 bucket already, create one with the name `aws s3 mb s3://<bucket name> .` 4. Package the Count macro template. The template uses the AWS Serverless Application Model (SAM),	DevOps engineer

Task	Description	Skills required

so it must be transformed before you can deploy it.

```
aws cloudformation
 package \
    --template-file
 template.yaml \
    --s3-bucket <your
 bucket name here> \
    --output-
template-file
 packaged.yaml
```

For example:

```
aws cloudformation
 package \
    --template-file
 template.yaml \
    --s3-bucket
 count-macro-ec2 \
    --output-
template-file
 packaged.yaml
```

5. Deploy the packaged template to create a CloudFormation stack.

```
aws cloudformation
 deploy \
    --stack-name
 Count-macro \
    --template-file
 packaged.yaml \
    --capabilities
 CAPABILITY_IAM
```

Task	Description	Skills required
	If you want to use the console, follow the instructions in the previous epic or in the [CloudFormation documentation]().	
Test the Count macro.	To test the macro's capabilities, try launching the example template that's provided with the macro. `aws cloudformation deploy \ --stack-name Count-test \ --template-file test.yaml \ --capabilities CAPABILITY_IAM`	DevOps engineer

Deploy the CloudFormation stack to provision instances with the custom image

Task	Description	Skills required
Deploy the Amazon EC2 provisioning template.	To deploy EC2 Image Pipeline by using CloudFormation: 1. Download the `ec2-provisioning.yaml` template from the [GitHub repository](), or locate it on your computer if you cloned the repository. 2. Open the [CloudFormation console]().	AWS DevOps

Task	Description	Skills required
	3. Repeat the steps from the first epic (or follow the instructions in the [CloudFormation documentation](#)) to deploy `ec2-provisioning.yaml` .	
View Amazon EC2 settings.	Amazon EC2 settings include security, networking, storage, status checks, monitoring, and tags configurations. To view these configurations: 1. Open the [Amazon EC2 console](#). 2. In the navigation pane, choose **Instances**, and then select the EC2 instance that was created by the Amazon EC2 provisioning template. 3. In the instance summary, select the tabs to view the corresponding Amazon EC2 settings.	AWS DevOps

Task	Description	Skills required
View the CloudWatch dashboard.	1. Open the [CloudWatch console](). 2. In the navigation pane, choose **Dashboards**. 3. Choose the dashboard that has your stack name. **Note** After you provision the stack, it takes time to populate the dashboard with metrics. The dashboard provides these metrics: `CPUUtilization` , `DiskUtilization` , `MemoryUtilization` , `NetworkIn` , `NetworkOut` , `StatusCheckFailed` .	AWS DevOps
View custom metrics for memory and disk usage.	1. On the [CloudWatch console](), choose **Dashboards**. 2. In the navigation pane, choose **Metrics**, **All metrics**. 3. Choose **Custom namespaces**, **CWAgent**.	AWS DevOps

Task	Description	Skills required
View alarms for memory and disk usage.	1. On the CloudWatch console, in the navigation pane, choose **Dashboards**. 2. Choose **All alarms**.	AWS DevOps
Verify the snapshot lifecyle rule.	1. Open the Amazon EC2 console. 2. In the navigation pane, choose **Lifecycle Manager**. 3. Verify the settings for the AMI lifecycle.	AWS DevOps

Delete the environment (optional)

Task	Description	Skills required
Delete the stacks.	When your PoC or pilot project is complete, we recommend that you delete the stacks you created to make sure that you aren't charged for these resources. 1. Open the AWS CloudFormation console. 2. In the navigation pane, choose **Stacks**, and then select one or both stacks you created earlier that you want to delete. The stack must be currently running. 3. In the stack details pane, choose **Delete**.	AWS DevOps

Task	Description	Skills required
	4. When prompted, choose **Delete stack**. ⚠ **Important** The stack deletion operation can't be stopped after it begins. The stack proceeds to the `DELETE_IN_PROGRESS` state. If the deletion fails, the stack will be in the `DELETE_FAILED` state. For solutions, see Delete stack fails in the AWS CloudFormation troubleshooting documentation. For information about protecting stacks from being accidentally deleted, see Protecting a stack from being deleted in the AWS CloudFormation documentation.	

Troubleshooting

Issue	Solution
When you deploy the Amazon EC2 provision ing template, you get the error: *Received malformed response from transform 123xxxx::Count*.	This is a known issue. (See the custom solution and PR in the AWS CloudFormation macros repository.) To fix this issue, open the AWS Lambda console and update `index.py` with the content from the GitHub repository.

Related resources

GitHub repositories

- UiPath RPA bot setup using CloudFormation
- Count CloudFormation Macro

AWS references

- Creating a stack on the AWS CloudFormation console (CloudFormation documentation)
- Troubleshooting CloudFormation (CloudFormation documentation)
- Monitor memory and disk metrics forAmazon EC2 instances (Amazon EC2 documentation)
- How can I use the CloudWatch agent to view metrics for Performance Monitor on a Windows server? (AWS re:Post article)

Additional references

- UiPath documentation
- Setting the Hostname in a SysPreped AMI (blog post by Brian Beach)
- How do I make Cloudformation reprocess a template using a macro when parameters change? (Stack Overflow)

Set up a highly available PeopleSoft architecture on AWS

Ramanathan Muralidhar, Amazon Web Services

Summary

When you migrate your PeopleSoft workloads to AWS, resiliency is an important objective. It ensures that your PeopleSoft application is always highly available and able to recover from failures quickly.

This pattern provides an architecture for your PeopleSoft applications on AWS to ensure high availability (HA) at the network, application, and database tiers. It uses an Amazon Relational Database Service (Amazon RDS) for Oracle or Amazon RDS for SQL Server database for the database tier. This architecture also includes AWS services such as Amazon Route 53, Amazon Elastic Compute Cloud (Amazon EC2) Linux instances, Amazon Elastic Block Storage (Amazon EBS), Amazon Elastic File System (Amazon EFS), and an Application Load Balancer, and is scalable.

Oracle PeopleSoft provides a suite of tools and applications for workforce management and other business operations.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A PeopleSoft environment with the necessary licenses for setting it up on AWS
- A virtual private cloud (VPC) set up in your AWS account with the following resources:
  - At least two Availability Zones
  - One public subnet and three private subnets in each Availability Zone
  - A NAT gateway and an internet gateway
  - Route tables for each subnet to route the traffic
  - Network access control lists (network ACLs) and security groups defined to help ensure the security of the PeopleSoft application in accordance with your organization’s standards

Limitations

- This pattern provides a high availability (HA) solution. It doesn't support disaster recovery (DR) scenarios. In the rare occurrence that the entire AWS Region for the HA implementation goes down, the application will become unavailable.

Product versions

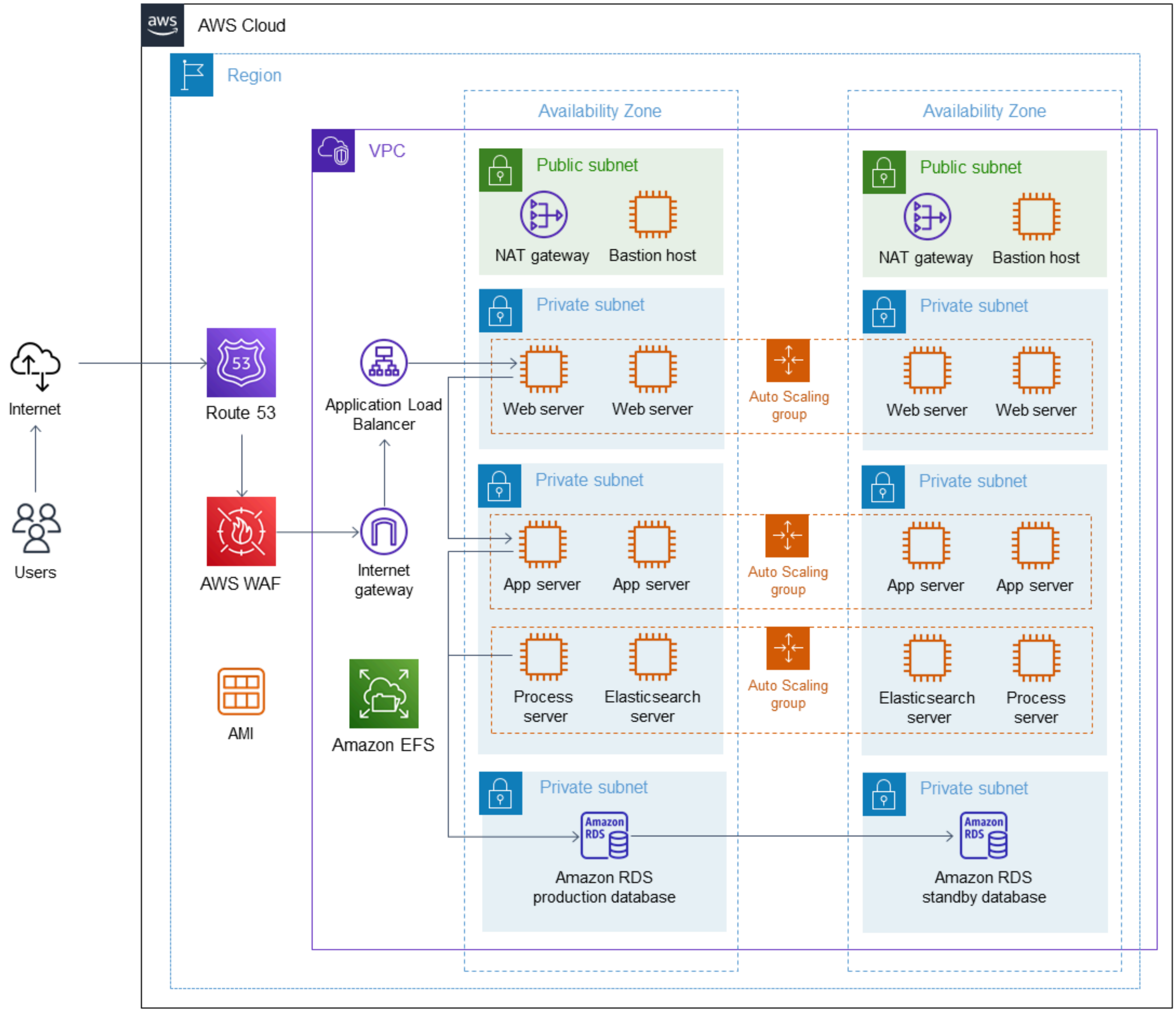
- PeopleSoft applications running PeopleTools 8.52 and later

Architecture

Target architecture

Downtime or outage of your PeopleSoft production application impacts the availability of the application and causes major disruptions to your business.

We recommend that you design your PeopleSoft production application so that it is always highly available. You can achieve this by eliminating single points of failure, adding reliable crossover or failover points, and detecting failures. The following diagram illustrates an HA architecture for PeopleSoft on AWS.



This architecture deployment uses Amazon RDS for Oracle as the PeopleSoft database, and EC2 instances that are running on Red Hat Enterprise Linux (RHEL). You can also use Amazon RDS for SQL Server as the Peoplesoft database.

This architecture contains the following components:

- Amazon Route 53 is used as the Domain Name Server (DNS) for routing requests from the internet to the PeopleSoft application.

- AWS WAF helps you protect against common web exploits and bots that can affect availability, compromise security, or consume excessive resources. AWS Shield Advanced (not illustrated) provides much broader protection.
- An Application Load Balancer load-balances HTTP and HTTPS traffic with advanced request routing targeted at the web servers.
- The web servers, application servers, process scheduler servers, and Elasticsearch servers that support the PeopleSoft application run in multiple Availability Zones and use Amazon EC2 Auto Scaling.
- The database used by the PeopleSoft application runs on Amazon RDS in a Multi-AZ configuration.
- The file share used by the PeopleSoft application is configured on Amazon EFS and is used to access files across instances.
- Amazon Machine Images (AMIs) are used by Amazon EC2 Auto Scaling to ensure that PeopleSoft components are cloned quickly when needed.
- The NAT gateways connect instances in a private subnet to services outside your VPC, and ensure that external services cannot initiate a connection with those instances.
- The internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet.
- The bastion hosts in the public subnet provide access to the servers in the private subnet from an external network, such as the internet or on-premises network. The bastion hosts provide controlled and secure access to the servers in the private subnets.

Architecture details

The PeopleSoft database is housed in an Amazon RDS for Oracle (or Amazon RDS for SQL Server) database in a Multi-AZ configuration. The Amazon RDS Multi-AZ feature replicates database updates across two Availability Zones to increase durability and availability. Amazon RDS automatically fails over to the standby database for planned maintenance and unplanned disruptions.

The PeopleSoft web and middle tier are installed on EC2 instances. These instances are spread across multiple Availability Zones and tied by an Auto Scaling group. This ensures that these components are always highly available. A minimum number of required instances are maintained to ensure that the application is always available and can scale when required.

We recommend that you use a current generation EC2 instance type for the OEM EC2 instances. Current generation instance types, such as instances built on the AWS Nitro System, support hardware virtual machines (HVMs). The HVM AMIs are required to take advantage of enhanced networking, and they also offer increased security. The EC2 instances that are part of each Auto Scaling group use their own AMI when replacing or scaling up instances. We recommend that you select EC2 instance types based on the load you want your PeopleSoft application to handle and the minimum values recommended by Oracle for your PeopleSoft application and PeopleTools release. For more information about hardware and software requirements, see the Oracle support website.

The PeopleSoft web and middle tier share an Amazon EFS mount to share reports, data files, and (if needed) the `PS_HOME` directory. Amazon EFS is configured with mount targets in each Availability Zone for performance and cost reasons.

An Application Load Balancer is provisioned to support the traffic that accesses the PeopleSoft application and load-balances the traffic among the web servers across different Availability Zones. An Application Load Balancer is a network device that provides HA in at least two Availability Zones. The web servers distribute the traffic to different application servers by using a load balancing configuration. Load balancing among the web server and application server ensures that load is distributed evenly across the instances, and helps avoid bottlenecks and service disruptions due to overloaded instances.

Amazon Route 53 is used as the DNS service to route traffic to the Application Load Balancer from the internet. Route 53 is a highly available and scalable DNS web service.

HA details

- Databases: The Multi-AZ feature of Amazon RDS operates two databases in multiple Availability Zones with synchronous replication. This creates a highly available environment with automatic failover. Amazon RDS has failover event detection and initiates automated failover when these events occur. You can also initiate manual failover through the Amazon RDS API. For a detailed explanation, see the blog post Amazon RDS Under The Hood: Multi-AZ. The failover is seamless and the application automatically reconnects to the database when it happens. However, any process scheduler jobs during the failover generate errors and have to be resubmitted.
- PeopleSoft application servers: The application servers are spread across multiple Availability Zones and have an Auto Scaling group defined for them. If an instance fails, the Auto Scaling group immediately replaces it with a healthy instance that's cloned from the AMI of the application server template. Specifically, *jolt pooling* is enabled, so when an application server

instance goes down, the sessions automatically fail over to another application server, and the Auto Scaling group automatically spins up another instance, brings up the application server, and registers it in the Amazon EFS mount. The newly created application server is automatically added to the web servers by using the `PSSTRSETUP.SH` script in the web servers. This ensures that the application server is always highly available and recovers from failure quickly.

- Process schedulers: The process schedulers servers are spread across multiple Availability Zones and have an Auto Scaling group defined for them. If an instance fails, the Auto Scaling group immediately replaces it with a healthy instance that's cloned from the AMI of the process scheduler server template. Specifically, when a process scheduler instance goes down, the Auto Scaling group automatically spins up another instance and brings up the process scheduler. Any jobs that were running when the instance failed must be resubmitted. This ensures that the process scheduler is always highly available and recovers from failure quickly.
- Elasticsearch servers: The Elasticsearch servers have an Auto Scaling group defined for them. If an instance fails, the Auto Scaling group immediately replaces it with a healthy instance that's cloned from the AMI of the Elasticsearch server template. Specifically, when an Elasticsearch instance goes down, the Application Load Balancer that serves requests to it detects the failure and stops sending traffic to it. The Auto Scaling group automatically spins up another instance and brings up the Elasticsearch instance. When the Elasticsearch instance is back up, the Application Load Balancer detects that it's healthy and starts sending requests to it again. This ensures that the Elasticsearch server is always highly available and recovers from failure quickly.
- Web servers: The web servers have an Auto Scaling group defined for them. If an instance fails, the Auto Scaling group immediately replaces it with a healthy instance that's cloned from the AMI of the web server template. Specifically, when a web server instance goes down, the Application Load Balancer that serves requests to it detects the failure and stops sending traffic to it. The Auto Scaling group automatically spins up another instance and brings up the web server instance. When the web server instance is back up, the Application Load Balancer detects that it's healthy and starts sending requests to it again. This ensures that the web server is always highly available and recovers from failure quickly.

Tools

AWS services

- Application Load Balancers distribute incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones.

- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Route 53 is a highly available and scalable DNS web service.

Best practices

Operational best practices

- When you run PeopleSoft on AWS, use Route 53 to route the traffic from the internet and locally. Use the failover option to reroute traffic to the disaster recovery (DR) site if the primary DB instance isn't available.
- Always use an Application Load Balancer in front of the PeopleSoft environment. This ensures that traffic is load-balanced to the web servers in a secure fashion.
- In the Application Load Balancer target group settings, make sure that stickiness is turned on with a load balancer generated cookie.

> **Note**
>
> You might need to use an application-based cookie if you use external single sign-on (SSO). This ensures that connections are consistent across the web servers and application servers.

- For a PeopleSoft production application, the Application Load Balancer idle timeout must match what is set in the web profile you use. This prevents user sessions from expiring at the load balancer layer.
- For a PeopleSoft production application, set the application server recycle count to a value that minimizes memory leaks.
- If you're using an Amazon RDS database for your PeopleSoft production application, as described in this pattern, run it in Multi-AZ format for high availability.

- If your database is running on an EC2 instance for your PeopleSoft production application, make sure that a standby database is running on another Availability Zone for high availability.
- For DR, make sure that your Amazon RDS database or EC2 instance has a standby configured in a separate AWS Region from the production database. This ensures that in event of a disaster in the Region, you can switch the application over to another Region.
- For DR, use Amazon Elastic Disaster Recovery to set up application-level components in a separate Region from production components. This ensures that in the event of a disaster in the Region, you can switch the application over to another Region.
- Use Amazon EFS (for moderate I/O requirements) or Amazon FSx (for high I/O requirements) to store your PeopleSoft reports, attachments, and data files. This ensures that the content is stored in one central location and can be accessed from anywhere within the infrastructure.
- Use Amazon CloudWatch (basic and detailed) to monitor the AWS Cloud resources that your PeopleSoft application is using in near real time. This ensures that you are alerted of issues instantly and can address them quickly before they affect the availability of the environment.
- If you're using an Amazon RDS database as the PeopleSoft database, use Enhanced Monitoring. This feature provides access to over 50 metrics, including CPU, memory, file system I/O, and disk I/O.
- Use AWS CloudTrail to monitor API calls on the AWS resources that your PeopleSoft application is using. This helps you perform security analysis, resource change tracking, and compliance auditing.

Security best practices

- To protect your PeopleSoft application from common exploits such as SQL injection or cross-site scripting (XSS), use AWS WAF. Consider using AWS Shield Advanced for tailored detection and mitigation services.
- Add a rule to the Application Load Balancer to redirect traffic from HTTP to HTTPS automatically to help secure your PeopleSoft application.
- Set up a separate security group for the Application Load Balancer. This security group should allow only HTTPS/HTTP inbound traffic and no outbound traffic. This ensures that only intended traffic is allowed and helps secure your application.
- Use private subnets for the application servers, web servers, and database, and use NAT gateways for outbound internet traffic. This ensures that the servers that support the application aren't reachable publicly, while providing public access only to the servers that need it.

- Use different VPCs to run your PeopleSoft production and non-production environments. Use AWS Transit Gateway, VPC peering, network ACLs, and security groups to control the traffic flow between the VPCs and, if necessary, your on-premises data center.
- Follow the principle of least privilege. Grant access to the AWS resources used by the PeopleSoft application only to users who absolutely need it. Grant only the minimum privileges required to perform a task. For more information, see the security pillar of the AWS Well-Architected Framework.
- Wherever possible, use AWS Systems Manager to access the EC2 instances that the PeopleSoft application uses.

Reliability best practices

- When you use an Application Load Balancer, register a single target for each enabled Availability Zone. This makes the load balancer most effective.
- We recommend that you have three distinct URLs for each PeopleSoft production environment: one URL to access the application, one to serve the integration broker, and one to view reports. If possible, each URL should have its own dedicated web servers and application servers. This design helps make your PeopleSoft application more secure, because each URL has a distinct functionality and controlled access. It also minimizes the scope of impact if the underlying services fail.
- We recommend that you configure health checks on the load balancer target groups for your PeopleSoft application. The health checks should be performed on the web servers instead of the EC2 instances running those servers. This ensures that if the web server crashes or the EC2 instance that hosts the web server goes down, the Application Load Balancer reflects that information accurately.
- For a PeopleSoft production application, we recommend that you spread the web servers across at least three Availability Zones. This ensures that the PeopleSoft application is always highly available even if one of the Availability Zones goes down.
- For a PeopleSoft production application, enable jolt pooling (`joltPooling=true`). This ensures that your application fails over to another application server if a server is down for patching purposes or because of a VM failure.
- For a PeopleSoft production application, set `DynamicConfigReload` to 1. This setting is supported in PeopleTools version 8.52 and later. It adds new application servers to the web server dynamically, without restarting the servers.

- To minimize downtime when you apply PeopleTools patches, use the blue/green deployment method for your Auto Scaling group launch configurations for the web and application servers. For more information, see the Overview of deployment options on AWS whitepaper.
- Use AWS Backup to back up your PeopleSoft application on AWS. AWS Backup is a cost-effective, fully managed, policy-based service that simplifies data protection at scale.

Performance best practices

- Terminate the SSL at the Application Load Balancer for optimal performance of the PeopleSoft environment, unless your business requires encrypted traffic throughout the environment.
- Create interface VPC endpoints for AWS services like such as Amazon Simple Notification Service (Amazon SNS) and CloudWatch so that traffic is always internal. This is cost-effective and helps keep your application secure.

Cost optimization best practices

- Tag all the resources used by your PeopleSoft environment, and enable cost allocation tags. These tags help you view and manage your resource costs.
- For a PeopleSoft production application, set up Auto Scaling groups for the web servers and the application servers. This maintains a minimal number of web and application servers to support your application. You can use Auto Scaling group policies to scale the the servers up and down as required.
- Use billing alarms to get alerts when costs exceed a budget threshold that you specify.

Sustainability best practices

- Use infrastructure as code (IaC) to maintain your PeopleSoft environments. This helps you build consistent environments and maintain change control.

Epics

Migrate your PeopleSoft database to Amazon RDS

Task	Description	Skills required
Create a DB subnet group.	On the Amazon RDS console, in the navigation pane, choose **Subnet groups**, and then create an Amazon RDS DB subnet group with subnets in multiple Availabil ity Zones. This is required for the Amazon RDS database to run in a Multi-AZ configura tion.	Cloud administrator
Create the Amazon RDS database.	Create an Amazon RDS database in an Availability Zone of the AWS Region you selected for the PeopleSof t HA environment. When you create the Amazon RDS database, make sure to select the Multi-AZ option (**Create a standby instance**) and the database subnet group you created in the previous step. For more information, see the Amazon RDS documentation.	Cloud administrator, Oracle database administrator
Migrate your PeopleSoft database to Amazon RDS.	Migrate your existing PeopleSoft database into the Amazon RDS database by using AWS Database Migration Service (AWS DMS). For more information, see the AWS DMS documentation and	Cloud administrator, PeopleSoft DBA

Task	Description	Skills required
	the AWS blog post [Migrating Oracle databases with near-zero downtime using AWS DMS]().	

Set up your Amazon EFS file system

Task	Description	Skills required
Create a file system.	On the [Amazon EFS console](), create a file system and mount targets for each Availability Zone. For instructions, see the [Amazon EFS documentation](). When the file system has been created, note its DNS name. You will use this information when you mount the file system.	Cloud administrator

Set up your PeopleSoft application and file system

Task	Description	Skills required
Launch an EC2 instance.	Launch an EC2 instance for your PeopleSoft application. For instructions, see the [Amazon EC2 documentation](). • For **Name**, enter `APP_TEMPLATE` . • For **OS images**, choose **Red Hat**.	Cloud administrator, PeopleSoft administrator

Task	Description	Skills required
	• For **Instance type**, choose the instance type that's appropriate for your PeopleSoft application. For more information, see *Architecture details* in the [Architecture](#) section.	
Install PeopleSoft on the instance.	Install your PeopleSoft application and PeopleToo ls on the EC2 instance you created. For instructions, see the [Oracle documentation](#).	Cloud administrator, PeopleSoft administrator
Create the application server.	Create the application server for the AMI template and make sure that it connects successfully to the Amazon RDS database.	Cloud administrator, PeopleSoft administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Mount the Amazon EFS file system.</td><td>Log in to the EC2 instance as the root user and run the following commands to mount the Amazon EFS file system to a folder called PSFTMNT on the server.
<pre><code>sudo su –
mkdir /psftmnt
cat /etc/fstab</code></pre>
Append the following line to the <code>/etc/fstab</code> file. Use the DNS name you noted when you created the file system.
<pre><code>fs-09e064308f11453
88.efs.us-east-1.a
mazonaws.com:/ /
psftmnt nfs4 nfsvers=4
.1,rsize=1048576,w
size=1048576,hard,
timeo=600,retrans=
2,noresvport,_netdev
 0 0
mount -a</code></pre></td><td>Cloud administrator, PeopleSoft administrator</td></tr>
<tr><td>Check permissions.</td><td>Make sure that the PSFTMNT folder has the proper permissions so that the PeopleSoft user can access it properly.</td><td>Cloud administrator, PeopleSoft administrator</td></tr>
</table>

Task	Description	Skills required
Create additional instances.	Repeat the previous steps in this epic to create template instances for the process scheduler, web server, and Elasticsearch server. Name these instances `PRCS_TEMPLATE`, `WEB_TEMPLATE`, and `SRCH_TEMPLATE`. For the web server, set `joltPooling=true` and `DynamicConfigReload=1`.	Cloud administrator, PeopleSoft administrator

Create scripts to set up servers

Task	Description	Skills required

Create a script to install the application server.	In the Amazon EC2 `APP_TEMPLATE` instance, as the PeopleSoft user, create the following script. Name it `appstart.sh` and place it in the PS_HOME directory. You will use this script to bring up the application server and also record the server name on the Amazon EFS mount.
	
	`#!/bin/ksh`
	`. /usr/homes/hcmdemo/.profile.`
	`psadmin -c configure -d HCMDEMO`
	`psadmin -c parallelboot -d HCMDEMO`
		PeopleSoft administrator

Task	Description	Skills required
	`touch /psftmnt/`echo $HOSTNAME``	

Create a script to install the process scheduler server.	In the Amazon EC2 `PRCS_TEMPLATE` instance, as the PeopleSoft user, create the following script. Name it `prcsstart.sh` and place it in the `PS_HOME` directory. You will use this script to bring up the process scheduler server.
	
	`#!/bin/ksh`
	`. /usr/homes/hcmdemo/.profile`
	`/* The following line ensures that the process scheduler always has a unique name during replacement or scaling activity. */`
	`sed -i "s/.*PrcsServerName.*/`hostname -I | awk -F. '{print "PrcsServerName=PSUNX"$3$4}'`/" $HOME/appserv/prcs/*/psprcs.cfg`
	`psadmin -p configure -d HCMDEMO`
	`psadmin -p start -d HCMDEMO`
		PeopleSoft administrator

Task	Description	Skills required

Create a script to install the Elasticsearch server.	In the Amazon EC2 `SRCH_TEMPLATE` instance, as the Elasticsearch user, create the following script. Name it `srchstart.sh` and place it in the `HOME` directory.
	
	`#!/bin/ksh`
	`/* The following line ensures that the correct IP is indicated in the elasticsearch.yaml file. */`
	`sed -i "s/.*network.host.*/`hostname -I | awk '{print "host:"$0}'`/" $ES_HOME_DIR/config/elasticsearch.yaml`
	`nohup $ES_HOME_DIR/bin/elasticsearch &`
		PeopleSoft administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a script to install the web server.</td><td>

In the Amazon EC2 `WEB_TEMPLATE` instance, as the web server user, create the following scripts in the `HOME` directory.

`renip.sh`: This script ensures that the web server has the correct IP when cloned from the AMI.

```
#!/bin/ksh
hn=`hostname`
/* On the following line, change the IP with the hostname with the hostname of the web template. */
for text_file in `find * -type f -exec grep -l '<hostname-of-the-web-template>' {} \;`
do
sed -e 's/<hostname-of-the-web-template>/'$hn'/g' $text_file > temp
mv -f temp $text_file
done
```

`psstrsetup.sh` : This script ensures that the web server uses the correct application server IPs that are currently running. It tries to connect to each application server on the jolt port and adds it to the configuration file.

</td><td>PeopleSoft administrator</td></tr>
</table>

Task	Description	Skills required
	<pre>#!/bin/ksh c2="" for ctr in `ls -1 / psftmnt/*.internal` do c1=`echo $ctr | awk -F  "/" '{print $3}'` /* In the following lines, 9000 is the jolt port. Change it if necessary. */ if nc -z $c1 9000 2> / dev/null; then if [[$c2 = ""]]; then c2="psserver="`echo  $c1`":9000" else c2=`echo $c2`","`echo  $c1`":9000" fi fi done</pre> `webstart.sh` : This script runs the two previous scripts and starts the web servers. <pre>#!/bin/ksh /* Change the path in the following if necessary. */ cd /usr/homes/hcmdemo ./renip.sh ./psstrsetup.sh webserv/peoplesoft/ bin/startPIA.sh</pre>	

Task	Description	Skills required
Add a crontab entry.	In the Amazon EC2 `WEB_TEMPLATE` instance, as the web server user, add the following line to **crontab**. Change the time and path to reflect the values you need. This entry ensures that your web server always has the correct application server entries in the `configuration.properties` file. `* * * * * /usr/homes/hcmdemo/psstrsetup.sh`	PeopleSoft administrator

Create AMIs and Auto Scaling group templates

Task	Description	Skills required
Create an AMI for the application server template.	On the Amazon EC2 console, create an AMI image of the Amazon EC2 `APP_TEMPLATE` instance. Name the AMI `PSAPPSRV-SCG-VER1` . For instructions, see the [Amazon EC2 documentation]().	Cloud administrator, PeopleSoft administrator
Create AMIs for the other servers.	Repeat the previous step to create AMIs for the process scheduler, Elasticsearch server, and web server.	Cloud administrator, PeopleSoft administrator

Task	Description	Skills required

Create a launch template for the application server Auto Scaling group.	Create a launch template for the application server Auto Scaling group. Name the template `PSAPPSRV_TEMPLATE.` In the template, choose the AMI you created for the `APP_TEMPLATE` instance. For instructions, see the [Amazon EC2 documentation]().
	• In the launch template, select the instance type based on your requirements.
	• In the **User data** field of the **Advanced details** section, add the following entries. Make sure that the path and user information are correct. You created the `appstart.sh` script in a previous step.
	`#! /bin/ksh`
	`su -c "/usr/homes/hcmdemo/appstart.sh" - hcmdemo`
		Cloud administrator, PeopleSoft administrator

Task	Description	Skills required

Create a launch template for the process scheduler server Auto Scaling group.	Repeat the previous step to create a launch template for the process scheduler server Auto Scaling group. Name the template `PSPRCS_TEMPLATE` . In the template, choose the AMI you created for the process scheduler.
	
	• In the **User data** field of the **Advanced details** section, add the following entries. Make sure that the path and user information are correct. You created the `prcsstart.sh` script in a previous step.
	
	`#! /bin/ksh`
	`su -c "/usr/homes/hcmdemo/prcsstart.sh" - hcmdemo`
		Cloud administrator, PeopleSoft administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a launch template for the Elasticsearch server Auto Scaling group.</td><td>Repeat the previous steps to create a launch template for the Elasticsearch server Auto Scaling group. Name the template <code>SRCH_TEMPLATE</code> . In the template, choose the AMI you created for the search server.
In the User data field of the Advanced details section, add the following entries. Make sure that the path and user information are correct. You created the <code>srchstart.sh</code> script in a previous step.<pre><code>#! /bin/ksh
su -c "/usr/homes/essearch/srchstart.sh" - essearch</code></pre></td><td>Cloud administrator, PeopleSoft administrator</td></tr>
</table>

Task	Description	Skills required

Create a launch template for the web server Auto Scaling group.	Repeat the previous steps to create a launch template for the web server Auto Scaling group. Name the template `WEB_TEMPLATE` . In the template, choose the AMI you created for the web server.
	• In the **User data** field of the **Advanced details** section, add the following entries. Make sure that the path and user information are correct. You created the `webstart.sh` script in a previous step.
	`#! /bin/ksh`
	`su -c "/usr/homes/hcmdemo/webstart.sh" - hcmdemo`
		Cloud administrator, PeopleSoft administrator

Create Auto Scaling groups

Task	Description	Skills required
Create an Auto Scaling group for the application server.	On the Amazon EC2 console, create an Auto Scaling group called PSAPPSRV_ASG for the application server by using the `PSAPPSRV_TEMPLATE` template. For instructions, see the [Amazon EC2 documentation]().	Cloud administrator, PeopleSoft administrator

Task	Description	Skills required
	• On the **Choose instance launch options** page, select the correct VPC and then select multiple subnets from different Availability Zones. • On the **Configure advanced options** page, do not select a load balancer. • On the **Configure group size and scaling policies** page, choose settings depending on how much load you want to architect your system for and whether you want to use a scaling policy. We recommend that you set the desired and minimum capacity to 2 at a minimum so that at least one instance is available to service the traffic at any point in time. For more information about Auto Scaling policies, see the [Amazon EC2 documentation]().	
Create Auto Scaling groups for the other servers.	Repeat the previous step to create Auto Scaling groups for the process scheduler, Elasticsearch server, and web server.	Cloud administrator, PeopleSoft administrator

Create and configure target groups

Task	Description	Skills required
Create a target group for the web server.	On the Amazon EC2 console, create a target group for the web server. For instructions, see the [Elastic Load Balancing documentation](). Set the port to the port that the web server is listening on.	Cloud administrator
Configure health checks.	Confirm that the health checks have the correct values to reflect your business requirements. For more information, see the [Elastic Load Balancing documentation]().	Cloud administrator
Create a target group for the Elasticsearch server.	Repeat the previous steps to create a target group called `PSFTSRCH` for the Elasticsearch server, and set the correct Elasticsearch port.	Cloud administrator
Add target groups to Auto Scaling groups.	Open the web server Auto Scaling group called `PSPIA_ASG` that you created earlier. On the **Load balancing** tab, choose **Edit** and then add the `PSFTWEB` target group to the Auto Scaling group. Repeat this step for the Elasticsearch Auto Scaling group `PSSRCH_ASG` to add	Cloud administrator

Task	Description	Skills required
	the target group PSFTSRCH you created earlier.	
Set session stickiness.	In the target group PSFTWEB, choose the **Attributes** tab, choose **Edit**, and set the session stickiness. For stickiness type, choose **Load balancer generated cookie**, and set the duration to 1. For more information, see the Elastic Load Balancing documentation. Repeat this step for the target group PSFTSRCH.	Cloud administrator

Create and configure application load balancers

Task	Description	Skills required
Create a load balancer for the web servers.	Create an Application Load Balancer named PSFTLB to load-balance traffic to the web servers. For instructions, see the Elastic Load Balancing documentation. • Provide the load balancer name. • For **Scheme**, choose **Internet-facing**. • In the **Network mapping** section, select the correct VPC and at least two public	Cloud administrator

Task	Description	Skills required
	subnets from different Availability Zones. • In the **Listeners and routing** section, select the target group PSFTWEB and specify the correct protocol and port number.	
Create a load balancer for the Elasticsearch servers.	Create an Application Load Balancer named PSFTSCH to load-balance traffic to the Elasticsearch servers. • Provide the load balancer name. • For **Scheme**, choose **Internal**. • In the **Network mapping** section, select the correct VPC and private subnets. • In the **Listeners and routing** section, select the target group PSFTSRCH and specify the correct protocol and port number.	Cloud administrator
Configure Route 53.	On the [Amazon Route 53 console](), create a record in the hosted zone that will service the PeopleSoft applicati on. For instructions, see the [Amazon Route 53 documenta tion](). This ensures that all the traffic passes through the PSFTLB load balancer.	Cloud administrator

Related resources

- Oracle PeopleSoft website
- AWS documentation

Set up disaster recovery for Oracle JD Edwards EnterpriseOne with AWS Elastic Disaster Recovery

Thanigaivel Thirumalai, Amazon Web Services

Summary

Disasters that are triggered by natural catastrophes, application failures, or disruption of services harm revenue and cause downtime for corporate applications. To reduce the repercussions of such events, planning for disaster recovery (DR) is critical for firms that adopt JD Edwards EnterpriseOne enterprise resource planning (ERP) systems and other mission-critical and business-critical software.

This pattern explains how businesses can use AWS Elastic Disaster Recovery as a DR option for their JD Edwards EnterpriseOne applications. It also outlines the steps for using Elastic Disaster Recovery failover and failback to construct a cross-Region DR strategy for databases hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance in the AWS Cloud.

> **Note**
>
> This pattern requires the primary and secondary Regions for the cross-Region DR implementation to be hosted on AWS.

Oracle JD Edwards EnterpriseOne is an integrated ERP software solution for midsize to large companies in a wide range of industries.

AWS Elastic Disaster Recovery minimizes downtime and data loss with fast, reliable recovery of on-premises and cloud-based applications by using affordable storage, minimal compute, and point-in-time recovery.

AWS provides four core DR architecture patterns. This document focuses on setup, configuration, and optimization by using the pilot light strategy. This strategy helps you create a lower-cost DR environment where you initially provision a replication server for replicating data from the source database, and you provision the actual database server only when you start a DR drill and recovery. This strategy removes the expense of maintaining a database server in the DR Region. Instead, you pay for a smaller EC2 instance that serves as a replication server.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A JD Edwards EnterpriseOne application running on Oracle Database or Microsoft SQL Server with a supported database in a running state on a managed EC2 instance. This application should include all JD Edwards EnterpriseOne base components (Enterprise Server, HTML Server, and Database Server) installed in one AWS Region.
- An AWS Identity and Access Management (IAM) role to set up the Elastic Disaster Recovery service.
- The network for running Elastic Disaster Recovery configured according to the required connectivity settings.

Limitations

- You can use this pattern to replicate all tiers, unless the database is hosted on Amazon Relational Database Service (Amazon RDS), in which case we recommend that you use the cross-Region copy functionality of Amazon RDS.
- Elastic Disaster Recovery isn't compatible with CloudEndure Disaster Recovery, but you can upgrade from CloudEndure Disaster Recovery. For more information, see the FAQ in the Elastic Disaster Recovery documentation.
- Amazon Elastic Block Store (Amazon EBS) limits the rate at which you can take snapshots. You can replicate a maximum number of 300 servers in a single AWS account by using Elastic Disaster Recovery. To replicate more servers, you can use multiple AWS accounts or multiple target AWS Regions. (You will have to set up Elastic Disaster Recovery separately for each account and Region.) For more information, see Best practices in the Elastic Disaster Recovery documentation.
- The source workloads (the JD Edwards EnterpriseOne application and database) must be hosted on EC2 instances. This pattern doesn't support workloads that are on premises or in other cloud environments.
- This pattern focuses on the JD Edwards EnterpriseOne components. A full DR and business continuity plan (BCP) should include other core services, including:
  - Networking (virtual private cloud, subnets, and security groups)
  - Active Directory
  - Amazon WorkSpaces

- Elastic Load Balancing
- A managed database service such as Amazon Relational Database Service (Amazon RDS)

For additional information about prerequisites, configurations, and limitations, see the Elastic Disaster Recovery documentation.

Product versions

- Oracle JD Edwards EnterpriseOne (Oracle and SQL Server supported versions based on Oracle Minimum Technical Requirements)

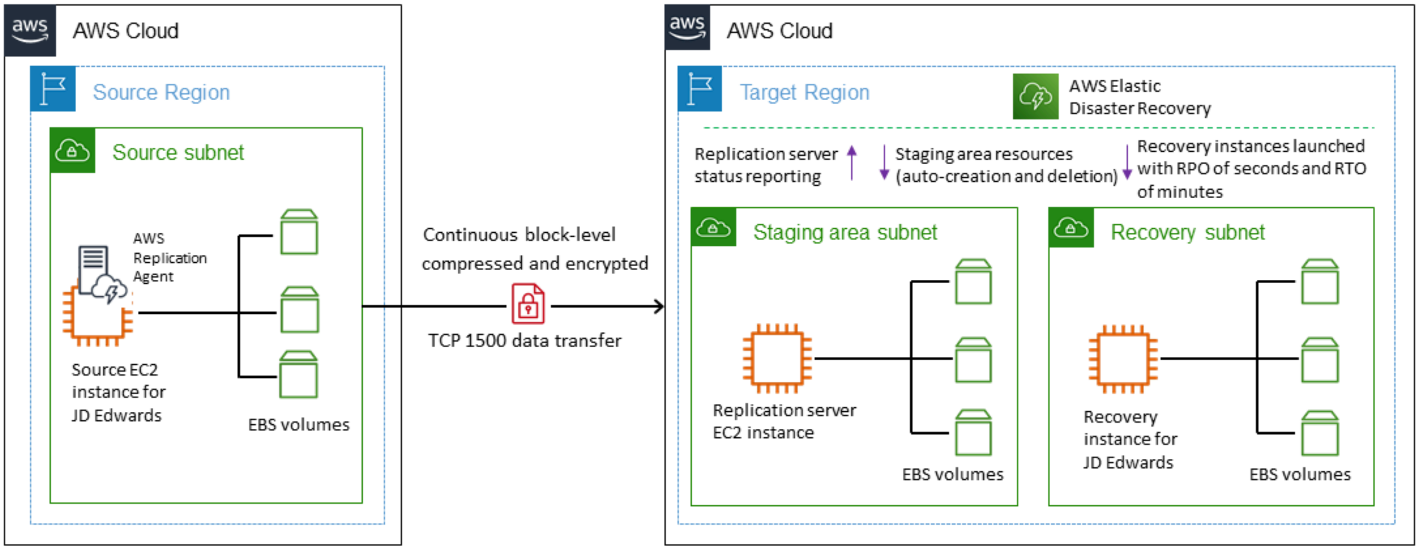
Architecture

Target technology stack

- A single Region and single virtual private cloud (VPC) for production and non-production, and a second Region for DR
- Single Availability Zones to ensure low latency between servers
- An Application Load Balancer that distributes network traffic to improve the scalability and availability of your applications across multiple Availability Zones
- Amazon Route 53 to provide Domain Name System (DNS) configuration
- Amazon WorkSpaces to provide users with a desktop experience in the cloud
- Amazon Simple Storage Service (Amazon S3) for storing backups, files, and objects
- Amazon CloudWatch for application logging, monitoring, and alarms
- Amazon Elastic Disaster Recovery for disaster recovery

Target architecture

The following diagram shows the cross-Region disaster recovery architecture for JD Edwards EnterpriseOne using Elastic Disaster Recovery.



Procedure

Here is a high-level review of the process. For details, see the *Epics* section.

- Elastic Disaster Recovery replication begins with an initial sync. During the initial sync, the AWS Replication Agent replicates all the data from the source disks to the appropriate resource in the staging area subnet.
- Continuous replication continues indefinitely after the initial sync is complete.
- You review the launch parameters, which include service-specific configurations and an Amazon EC2 launch template, after the agent has been installed and replication has started. When the source server is indicated as being ready for recovery, you can start instances.
- When Elastic Disaster Recovery issues a series of API calls to begin the launch operation, the recovery instance is immediately launched on AWS according to your launch settings. The service automatically spins up a conversion server during startup.
- The new instance is spun up on AWS after the conversion is complete and is ready for use. The source server state at the time of launch is represented by the volumes associated with the launched instance. The conversion process involves changes to the drivers, network, and operating system license to ensure that the instance boots natively on AWS.
- After the launch, the newly created volumes are no longer kept in sync with the source servers. The AWS Replication Agent continues to routinely replicate changes made to your source servers to the staging area volumes, but the launched instances do not reflect those changes.
- When you start a new drill or recovery instance, the data is always reflected in the most recent state that has been replicated from the source server to the staging area subnet.

- When the source server is marked as being prepared for recovery, you can start instances.

> **Note**
>
> The process works both ways: for failover from a primary AWS Region to a DR Region, and to fail back to the primary site, when it has been recovered. You can prepare for failback by reversing the direction of data replication from the target machine back to the source machine in a fully orchestrated way.

The benefits of this process described in this pattern include:

- Flexibility: Replication servers scale out and scale in based on dataset and replication time, so you can perform DR tests without disrupting source workloads or replication.
- Reliability: The replication is robust, non-disruptive, and continuous.
- Automation: This solution provides a unified, automated process for test, recovery, and failback.
- Cost optimization: You can replicate only the needed volumes and pay for them, and pay for compute resources at the DR site only when those resources are activated. You can use a cost-optimized replication instance (we recommend that you use a compute-optimized instance type) for multiple sources or a single source with a large EBS volume.

Automation and scale

When you perform disaster recovery at scale, the JD Edwards EnterpriseOne servers will have dependencies on other servers in the environment. For example:

- JD Edwards EnterpriseOne application servers that connect to a JD Edwards EnterpriseOne supported database on boot have dependencies on that database.
- JD Edwards EnterpriseOne servers that require authentication and need to connect to a domain controller on boot to start services have dependencies on the domain controller.

For this reason, we recommend that you automate failover tasks. For example, you can use AWS Lambda or AWS Step Functions to automate the JD Edwards EnterpriseOne startup scripts and load balancer changes to automate the end-to-end failover process. For more information, see the blog post [Creating a scalable disaster recovery plan with AWS Elastic Disaster Recovery](#).

Tools

AWS services

- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with EC2 instances.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Elastic Disaster Recovery minimizes downtime and data loss with fast, reliable recovery of on-premises and cloud-based applications using affordable storage, minimal compute, and point-in-time recovery.
- Amazon Virtual Private Cloud (Amazon VPC) gives you full control over your virtual networking environment, including resource placement, connectivity, and security.

Best practices

General best practices

- Have a written plan of what to do in the event of a real recovery event.
- After you set up Elastic Disaster Recovery correctly, create an AWS CloudFormation template that can create the configuration on demand, should the need arise. Determine the order in which servers and applications should be launched, and record this in the recovery plan.
- Perform a regular drill (standard Amazon EC2 rates apply).
- Monitor the health of the ongoing replication by using the Elastic Disaster Recovery console or programmatically.
- Protect the point-in-time snapshots and confirm before terminating the instances.
- Create a IAM role for AWS Replication Agent installation.
- Enable termination protection for recovery instances in a real DR scenario.
- Do not use the **Disconnect from AWS** action in the Elastic Disaster Recovery console for servers that you launched recovery instances for, even in the case of a real recovery event. Performing a disconnect terminates all replication resources related to these source servers, including your point-in-time (PIT) recovery points.
- Change the PIT policy to change the number of days for snapshot retention.
- Edit the launch template in Elastic Disaster Recovery launch settings to set the correct subnet, security group, and instance type for your target server.

- Automate the end-to-end failover process by using Lambda or Step Functions to automate JD Edwards EnterpriseOne startup scripts and load balancer changes.

JD Edwards EnterpriseOne optimization and considerations

- Move **PrintQueue** into the database.
- Move **MediaObjects** into the database.
- Exclude the logs and temp folder from batch and logic servers.
- Exclude the temp folder from Oracle WebLogic.
- Create scripts for startup after the failover.
- Exclude the tempdb for SQL Server.
- Exclude the temp file for Oracle.

Epics

Perform initial tasks and configuration

Task	Description	Skills required
Set up the replication network.	Implement your JD Edwards EnterpriseOne system in the primary AWS Region and identify the AWS Region for DR. Follow the steps in the Replication network requirements section of the Elastic Disaster Recovery documentation to plan and set up your replication and DR network.	AWS administrator
Determine RPO and RTO.	Identify the recovery time objective (RTO) and recovery point objective (RPO) for	Cloud architect, DR architect

Task	Description	Skills required
	your application servers and database.	
Enable replication for Amazon EFS.	If applicable, enable replicati on from the AWS primary to DR Region for shared file systems such as Amazon Elastic File System (Amazon EFS) by using AWS DataSync, **rsync**, or another appropriate tool.	Cloud administrator
Manage DNS in case of DR.	Identify the process to update the Domain Name System (DNS) during the DR drill or actual DR.	Cloud administrator
Create an IAM role for setup.	Follow the instructions in the [Elastic Disaster Recovery initialization and permissions](#) section of the Elastic Disaster Recovery documentation to create an IAM role to initialize and manage the AWS service.	Cloud administrator
Set up VPC peering.	Make sure that the source and target VPCs are peered and accessible to each other. For configuration instructions, see the [Amazon VPC documenta tion](#).	AWS administrator

Configure Elastic Disaster Recovery replication settings

Task	Description	Skills required
Initialize Elastic Disaster Recovery.	Open the [Elastic Disaster Recovery console](), choose the target AWS Region (where you will replicate data and launch recovery instances), and then choose **Set default replication settings**.	AWS administrator
Set up replication servers.	1. In the **Set up replication servers** pane, enter the staging area subnet and replication server instance type. The `t3.small` instance type is selected by default. Configure this setting based on your requirements, and remember to consider instance pricing. For more information, see [Amazon EC2 pricing](). 2. In the **Service access** section, choose **View details** to review the service linked role and additional policies created during service initialization. 3. Choose **Next**.	AWS administrator
Configure volumes and security groups.	1. In the **Volumes and security groups** pane, select the EBS volume type for the replication	AWS administrator

Task	Description	Skills required
	server and set Amazon EBS encryption to **Default.** 2. Select **Always use AWS Elastic Disaster Recovery security group** so that Elastic Disaster Recovery automatically attaches and monitors the default security group. 3. Choose **Next**.	

Task	Description	Skills required
Configure additional settings.	1. In the **Additional settings** pane, configure data routing and throttling, PIT policy, and tags. • Data routing and throttling control how data flows from the external server to the replication servers. Choose **Use private IP for data replication**. Otherwise, replication servers will be automatic ally assigned a public IP, and data will flow over the public internet. • In the **Point in time (PIT) policy** section, configure a retention policy that determine s the duration after which snapshots are not required. The default retention period is seven days. • In the **Tags** section, add custom tags to resources created by Elastic Disaster Recovery in your AWS account. 2. Choose **Next**, review the settings in the next pane, and then choose **Create**	AWS administrator

Task	Description	Skills required
	default to create the default template.	

Install the AWS Replication Agent

Task	Description	Skills required
Create an IAM role.	Create an IAM role that contains the `AWSElasticDisasterRecoveryAgentInstallationPolicy` policy. In the **Select AWS access type** section, enable programmatic access. Note the access key ID and secret access key. You will need this information during the installation of the AWS Replication Agent.	AWS administrator
Check requirements.	Check and complete the [prerequisites](#) in the Elastic Disaster Recovery documentation for installing the AWS Replication Agent.	AWS administrator
Install the AWS Replication Agent.	Follow the [installation instructions](#) for your operating system and install the AWS Replication Agent. • For Microsoft Windows: Download the setup files and run the .exe file as an administrator. Respond to	AWS administrator

Task	Description	Skills required

	the prompts to complete the installation.
	• For Linux: Copy the following commands (in the order presented) and paste them into your Secure Shell (SSH) session. The first command downloads the installer and the second command runs it.
	`wget -O ./aws-replication-installer-init.py https://aws-elastic-disaster-recovery-us-west-2.s3.amazonaws.com/latest/linux/aws-replication-installer-init.py`
	`sudo python3 aws-replication-installer-init.py`
	Respond to the prompts to complete the installation.
	Note
	Change the URL to reflect your Region.
	
	Repeat these steps for the remaining server.	

Task	Description	Skills required
Monitor the replication.	Return to the Elastic Disaster Recovery **Source servers** pane to monitor the replication status. The initial sync will take some time depending on the size of the data transfer. When the source server is fully synced, the server status will be updated to **Ready**. This means that a replicati on server has been created in the staging area, and the EBS volumes have been replicated from the source server to the staging area.	AWS administrator

Configure launch settings

Task	Description	Skills required
Edit launch settings.	To update the launch settings for the drill and recovery instances, on the [Elastic Disaster Recovery console](https://console.aws.amazon.com/drs/home), select the source server, and then choose **Actions**, **Edit launch settings**. Or you can choose your replicati ng source machines from the **Source servers** page, and then choose the **Launch Settings** tab. This tab has two sections: **General launch**	AWS administrator

Task	Description	Skills required
	settings and **EC2 launch template**.	
Configure general launch settings.	Revise the general launch settings according to your requirements. • **Instance type right sizing**: If you choose **Basic**, Elastic Disaster Recovery bypasses the instance type you selected in the Amazon EC2 launch template and automatically chooses the instance type based on the operating system, CPU, and RAM of the source server. • **Copy private IP**: Choose whether you want Elastic Disaster Recovery to ensure that the private IP used by the drill or recovery instance matches the private IP used by the source server. If you chose **Yes**, make sure that the IP range of the subnet you set in the Amazon EC2 launch template includes the private IP address. For more information, see General launch settings in the Elastic Disaster Recovery documentation.	AWS administrator

Task	Description	Skills required
Configure the Amazon EC2 launch template.	Elastic Disaster Recovery uses Amazon EC2 launch templates to launch drill and recovery instances for each source server. The launch template is created automatically for each source server that you add to Elastic Disaster Recovery after you install the AWS Replication Agent. You must set the Amazon EC2 launch template as the default launch template if you want to use it with Elastic Disaster Recovery. For more information, see EC2 Launch Template in the Elastic Disaster Recovery documentation.	AWS administrator

Initiate DR drill and failover

Task	Description	Skills required
Initiate Drill	1. On the Elastic Disaster Recovery console, open the **Source servers** page and verify that the status of the source server is **Ready**.	AWS administrator

Task	Description	Skills required
	2. Select all the source servers that you want to perform the DR drill for. 3. From the **Initiate recovery job** menu, choose **Initiate drill** and select the appropriate point-in-time snapshot. This starts a recovery job for the selected source servers. You can monitor the status of the job on the **Recovery job history** tab. The launched drill instance also appears on the **Recovery instances** page. **Note** Further changes to the source server will be synced to the replication server, not to the drill instance. 4. Test and verify the DR drill instance. 5. On the **Recovery instances** page, select the drill instance, and then choose **Actions**, **Disconnect from AWS**. This deletes the AWS Replication	

Task	Description	Skills required
	Agent from the recovery instance and removes all resources associated with the recovery instance from Elastic Disaster Recovery. 6. Choose **Delete recovery instances**. This deletes the representation of the instance from the Elastic Disaster Recovery console and completely disassociates the instance from the Elastic Disaster Recovery service. It does not delete the underlying EC2 instance. 7. Terminate the DR drill instance from the Amazon EC2 console. For more information, see [Preparing for failover] in the Elastic Disaster Recovery documentation.	

Task	Description	Skills required
Validate the drill.	In the previous step, you launched new target instances in the DR Region. The target instances are replicas of the source servers based on the snapshot taken when you initiated the launch. In this procedure, you connect to your Amazon EC2 target machines to confirm that they're running as expected. 1. Open the Amazon EC2 console. 2. Choose **Instances (running)**. 3. Select the target instance and note its private IPv4 address. 4. Make sure that you can connect to the EC2 instance and that the JD Edwards EnterpriseOne and related components are replicated as expected.	

Task	Description	Skills required
Initiate a failover.	A failover is the redirection of traffic from a primary system to a secondary system. Elastic Disaster Recovery helps you perform a failover by launching recovery instances on AWS. When the recovery instances have been launched, you redirect the traffic from your primary systems to these instances. 1. On the [Elastic Disaster Recovery console](#), open the **Source servers** page and verify that the **Ready for recovery** column for the source server shows **Ready**, and the **Data replication status** column shows **Healthy**. 2. Select the source server. From the **Initiate recovery job** menu, choose **Initiate recovery**. 3. Select the point-in-time snapshot from which to launch the recovery instance, and then choose **Initiate recovery**. This starts a recovery job. You can monitor the status of the job on the **Recovery instances** page.	AWS administrator

Task	Description	Skills required
	4. Test and verify the recovery instance. If required, adjust the DNS configuration and connect your JD Edwards Enterpris eOne application to the database. 5. You can now disconnec t and decommission the source JD Edwards EnterpriseOne server, because all changes have been written to the new recovery instance. 6. Register the recovery instance as the source server in the DR Region by following the process described in the *Install the AWS Replication Agent* epic. For more information, see [Performing a failover](#) in the Elastic Disaster Recovery documentation.	

Task	Description	Skills required
Initiate a failback.	The process for initiating a failback is similar to the process for initiating failover. 1. Open the [Elastic Disaster Recovery console]() in the primary Region. Navigate to the **Recovery instances** page, select the drill instance, and then choose **Actions**, **Disconnect from AWS**, **Delete recovery instances**. 2. Open the Elastic Disaster Recovery console in the DR Region. Register your new JD Edwards EnterpriseOne server as the source server in the DR Region by installing the AWS Replication Agent. The data will be synced with a new replication server provisioned in the new staging subnet. **Note** When the new JD Edwards EnterpriseOne server is registered as a source server, you might see two source servers in the Elastic Disaster	AWS administrator

Task	Description	Skills required
	Recovery console: one server that was created from the primary EC2 instance, and the new server that was created from the recovery instance. We recommend that you tag the servers correctly to avoid confusion, and preferably add the new server to the launch template. 3. To restart DR replication from the primary Region, disassociate the launched recovery instance from the Elastic Disaster Recovery console in the DR Region, and register the host as a source server in the primary Region. For more information, see Performing a failback in the Elastic Disaster Recovery documentation.	

Task	Description	Skills required
Start JD Edwards EnterpriseOne components.	1. Start the JD Edwards EnterpriseOne database by logging in to the database server. 2. When the database is running, start the JD Edwards EnterpriseOne logic and batch servers. 3. Start WebLogic on the web servers, and start a JAS instance on the JAS servers. 4. Start WebLogic on the provision server and on the server for the SM console. 5. Start SM Agent on the servers. 6. Confirm that the login to JD Edwards EnterpriseOne works correctly. You will need to make incorporate the changes in Route 53 and Application Load Balancer for the JD Edwards EnterpriseOne link to work. You can automate these steps by using Lambda, Step Functions, and Systems Manager (Run Command).	JD Edwards EnterpriseOne CNC

Task	Description	Skills required
	Note Elastic Disaster Recovery performs block-level replication of the source EC2 instance EBS volumes that host the operating system and file systems. Shared file systems that were created by using Amazon EFS aren't part of this replication. You can replicate shared file systems to the DR Region by using AWS DataSync, as noted in the first epic, and then mount these replicated file systems in the DR system.	

Troubleshooting

Issue	Solution
Source server data replication status is **Stalled** and replication lags. If you check details, the data replication status displays **Agent not seen**.	Check to confirm that the stalled source server is running.

Issue	Solution
	Note If the source server goes down, the replication server is automatically terminated. For more information about lag issues, see Replication lag issues in the Elastic Disaster Recovery documentation.
Installation of AWS Replication Agent in source EC2 instance fails in RHEL 8.2 after scanning the disks. `aws_replication_agent_installer.log` reveals that kernel headers are missing.	Before you install the AWS Replication Agent on RHEL 8, CentOS 8, or Oracle Linux 8, run: `sudo yum install elfutils-libelf-devel` For more information, see Linux installation requirements in the Elastic Disaster Recovery documentation.
On the Elastic Disaster Recovery console, you see the source server as **Ready** with a lag and data replication status as **Stalled**. Depending on how long the AWS Replication Agent has been unavailable, the status might indicate high lag, but the issue remains the same.	Use an operating system command to confirm that the AWS Replication Agent is running in the source EC2 instance, or confirm that the instance is running. After you correct any issues, Elastic Disaster Recovery will restart scanning. Wait until all data has been synced and the replication status is **Healthy** before you start a DR drill.

Issue	Solution
Initial replication with high lag. On the Elastic Disaster Recovery console, you can see that the initial sync status is extremely slow for a source server.	Check for the replication lag issues documente d in the Replication lag issues section of the Elastic Disaster Recovery documentation. The replication server might be unable to handle the load because of intrinsic compute operations. In that case, try upgrading the instance type after consulting with the AWS Technical Support team.

Related resources

- AWS Elastic Disaster Recovery User Guide
- Creating a scalable disaster recovery plan with AWS Elastic Disaster Recovery (AWS blog post)
- AWS Elastic Disaster Recovery - A Technical Introduction (AWS Skill Builder course; requires login)
- AWS Elastic Disaster Recovery quick start guide

Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization

Ram Kandaswamy, Amazon Web Services

Summary

Amazon Web Services (AWS) users often look for an efficient way to detect resource configuration mismatches, including drift in AWS CloudFormation stacks, and fix them as soon as possible. This is especially the case when AWS Control Tower is used.

This pattern provides a prescriptive solution that efficiently solves the problem by using consolidated resource configuration changes and acting on those changes to generate results. The solution is designed for scenarios where there are several AWS CloudFormation stacks created in more than one AWS Region, or in more than one account, or a combination of both. The goals of the solution are the following:

- Simplify the drift detection process
- Set up notification and alerting
- Set up consolidated reporting

Prerequisites and limitations

Prerequisites

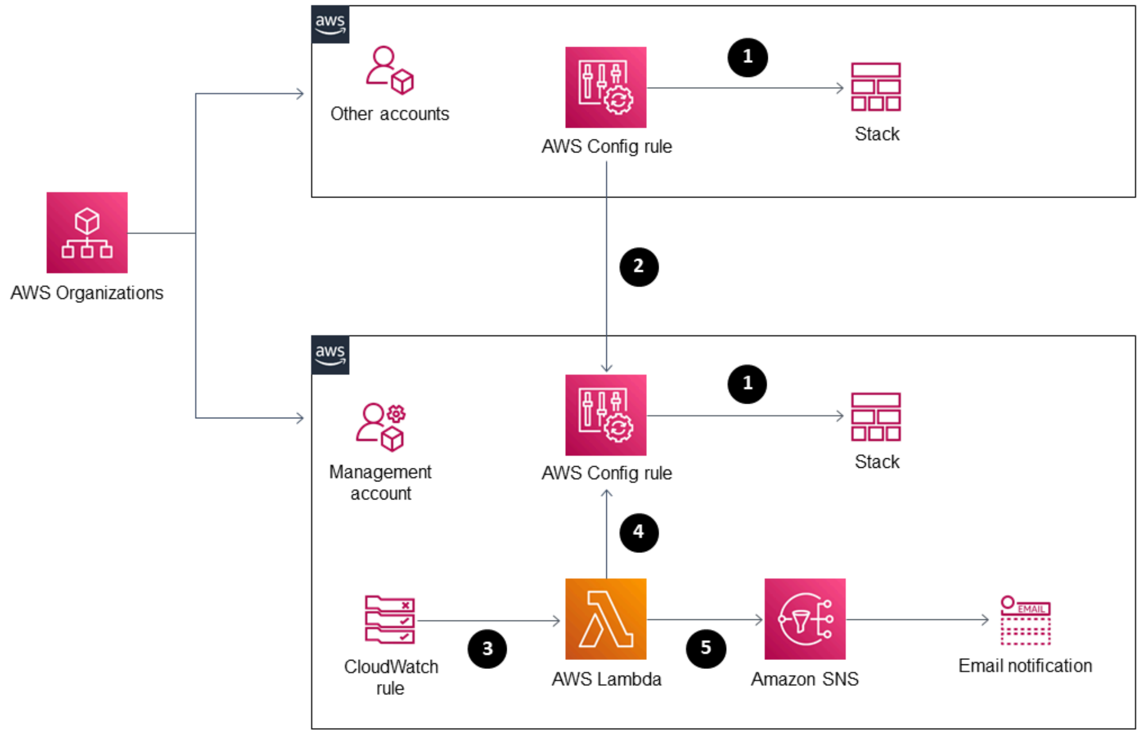
- AWS Config enabled in all the Regions and accounts that must be monitored

Limitations

- The report generated supports only the comma-separated values (CSV) and JSON output formats.

Architecture

The following diagram shows AWS Organizations set up with multiple accounts. AWS Config rules communicate between the accounts.



The workflow includes the following steps:

1. The AWS Config rule detects drift.
2. Drift detection results that are found in other accounts are sent to the management account.
3. The Amazon CloudWatch rule calls an AWS Lambda function.
4. The Lambda function queries the AWS Config rule for aggregated results.
5. The Lambda function notifies Amazon Simple Notification Service (Amazon SNS), which sends email notification of the drift.

Automation and scale

The solution presented here can scale for both additional Regions and accounts.

Tools

AWS services

- AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. This includes how the resources are related to one another and how they were configured in the past so that you can see how the configurations and relationships change over time.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

Epics

Automate drift detection for AWS CloudFormation

Task	Description	Skills required
Create the aggregator.	1. Sign in to the AWS Management Console and open the AWS Config console at https://console.aws.amazon.com/config. 2. Create an aggregator in the management account. 3. Ensure that data replication is turned on so that AWS Config can fetch data from the source accounts. 4. Select all applicable Regions and accounts. You can select accounts based on AWS Organizations. We recommend this approach	Cloud architect

Task	Description	Skills required
	because new accounts in the organization are automatically part of the aggregator.	
Create an AWS managed rule.	Add the `cloudformation-stack-drift-detection-check` AWS managed rule. The rule needs one parameter value: `cloudformationArn` . Enter the IAM role Amazon Resource Name (ARN) that has permissions to detect stack drift. The role must have a trust policy that enables AWS Config to assume the role.	Cloud architect
Create the advanced query section of the aggregator.	To fetch drifted stacks from multiple sources, create the following query: `SELECT resourceId, configuration.driftInformation.stackDriftStatus WHERE resourceType = 'AWS::CloudFormation::Stack' AND configuration.driftInformation.stackDriftStatus IN ('DRIFTED')`	Cloud architect, Developer

Task	Description	Skills required
Automate running the query and publish.	1. Create a Lambda function by using the code that is attached. Lambda will publish the results to an SNS topic that is provided as an environment variable in the Lambda function. 2. To receive alerts, create an email subscription to the SNS topic.	Cloud architect, Developer
Create a CloudWatch rule.	Create a schedule-based CloudWatch rule to call the Lambda function, which is responsible for alerting.	Cloud architect

Related resources

Resources

- What Is AWS Config?
- Multi-account multi-Region data aggregation
- Detecting unmanaged configuration changes to stacks and resources
- IAM: Pass an IAM role to a specific AWS service
- What is Amazon SNS?

Additional information

Considerations

We recommend using the solution presented in this pattern instead of using custom solutions that involve API calls at specific intervals to initiate drift detection on each CloudFormation stack or stack set. Custom solutions that use API calls at specific intervals can lead to a large number API calls and affect performance. Because of the number of API calls, throttling can occur. Another potential issue is a delay in detection if resource changes are identified based on schedule only.

Because stack sets are made of stacks, you can use this solution. Stack instance details are also available as part of the solution.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Successfully import an S3 bucket as an AWS CloudFormation stack

Ram Kandaswamy, Amazon Web Services

Summary

If you use Amazon Web Services (AWS) resources, such as Amazon Simple Storage Service (Amazon S3) buckets, and want to use an infrastructure as code (IaC) approach, then you can import your resources into AWS CloudFormation and manage them as a stack.

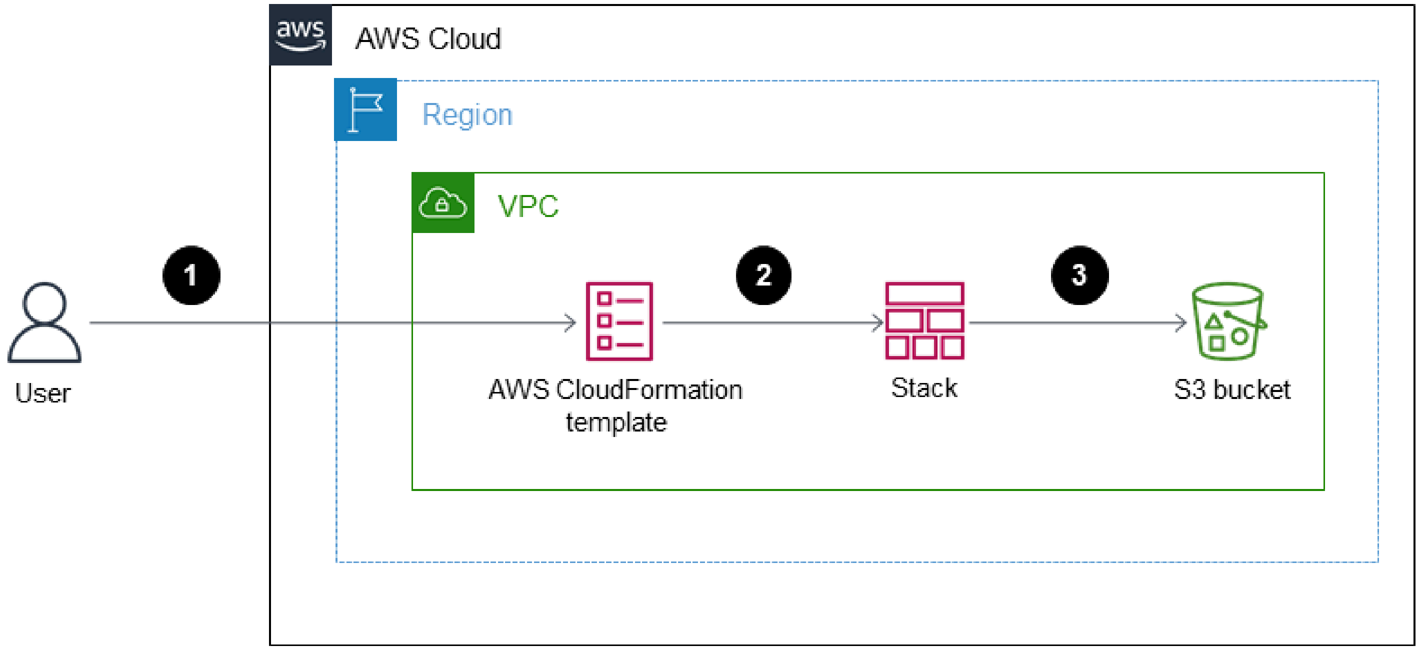
This pattern provides steps to successfully import an S3 bucket as an AWS CloudFormation stack. By using this pattern's approach, you can avoid possible errors that might occur if you import your S3 bucket in a single action.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing S3 bucket and S3 bucket policy. For more information about this, see What S3 bucket policy should I use to comply with the AWS Config rule s3-bucket-ssl-requests-only in the AWS Knowledge Center.
- An existing AWS Key Management Service (AWS KMS) key and its alias. For more information about this, see Working with aliases in the AWS KMS documentation.
- The sample `CloudFormation-template-S3-bucket` AWS CloudFormation template (attached), downloaded to your local computer.

Architecture



The diagram shows the following workflow:

1. The user creates a JSON or YAML-formatted AWS CloudFormation template.
2. The template creates an AWS CloudFormation stack to import the S3 bucket.
3. The AWS CloudFormation stack manages the S3 bucket that you specified in the template.

Technology stack

- AWS CloudFormation
- AWS Identity and Access Management (IAM)
- AWS KMS
- Amazon S3

Tools

- AWS CloudFormation – AWS CloudFormation helps you to create and provision AWS infrastructure deployments predictably and repeatedly.

- [AWS Identity and Access Management (IAM)] – IAM is a web service for securely controlling access to AWS services.
- [AWS KMS] – AWS Key Management Service (AWS KMS) is an encryption and key management service scaled for the cloud.
- [Amazon S3] – Amazon Simple Storage Service (Amazon S3) is storage for the Internet.

Epics

Import an S3 bucket with AWS KMS key-based encryption as an AWS CloudFormation stack

Task	Description	Skills required

Create a template to import the S3 bucket and KMS key.	On your local computer, create a template to import your S3 bucket and KMS key by using the following sample template:
	``
	AWSTemplateFormatVersion: 2010-09-09
	Parameters:
	  bucketName:
	    Type: String
	Resources:
	  S3Bucket:
	    Type: 'AWS::S3::Bucket'
	    DeletionPolicy: Retain
	    Properties:
	``
		AWS DevOps

Task	Description	Skills required

	``
	      BucketName: !Ref
	 bucketName
	
	      BucketEncryption:
	
	        ServerSid
	eEncryptionConfigu
	ration:
	
	          - ServerSid
	eEncryptionByDefault:
	
	              SSEAlgori
	thm: 'aws:kms'
	
	              KMSMaster
	KeyID: !GetAtt
	
	                -
	 KMSS3Encryption
	
	                - Arn
	
	  KMSS3Encryption:
	
	    Type: 'AWS::KMS
	::Key'
	
	    DeletionPolicy:
	 Retain
	
	    Properties:
	
	      Enabled: true
	
	      KeyPolicy: !Sub
	 |-
	
	        {
	
	            "Id": "key-
	consolepolicy-3",
	``	

Task	Description	Skills required
| | ```
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "Enable IAM User Permissions",
 "Effect": "Allow",
 "Principal": {
 "AWS": ["arn:aws:iam::${AWS::AccountId}:root"]
 },
 "Action": "kms:*",
 "Resource": "*"
 }
 }
 ]
 }
``` | |
```

Task	Description	Skills required
	`EnableKey` `Rotation: true`	
Create the stack.	1. Sign in to the AWS Management Console, open the AWS CloudFormation console, choose **View stack**, choose **Create stack**, and then choose **With existing resources (import resources)**. 2. Choose **Upload a template file** and then upload the template file that you created earlier. 3. Enter a name for your stack and configure the remaining options according to your requirements. 4. Choose **Create stack** and wait for the stack's status to change to `IMPORT_COMPLETE` .	AWS DevOps

Task	Description	Skills required

Create the KMS key alias.	1. On the AWS CloudFormation console, choose **Stacks**, choose the name of the stack that you created earlier, choose the **Template** pane, and then choose **View in Designer**.
	2. Add the following snippet to the `Resource` section of your template, and then choose **Create stack** and complete the wizard:
	
	`KMSS3EncryptionAlias:`
	`    Type: 'AWS::KMS::Alias'`
	`    DeletionPolicy: Retain`
	`    Properties:`
	`    AliasName: alias/S3BucketKey`
	`    TargetKeyId: !Ref KMSS3Encryption`
	
	For more information about this, see [AWS CloudFormation stack updates](#) in the AWS CloudFormation documentation.
		AWS DevOps

Task	Description	Skills required

Update the stack to include the S3 bucket policy.	1. On the AWS CloudFormation console, choose **Stacks**, choose the name of the stack that you created earlier, choose the **Template** pane, and then choose **View in Designer**.
	2. Add the following snippet to the `Resource` section of the template, and then choose **Create stack** and complete the wizard:
	<pre>S3BucketPolicy:
	
	  Type: 'AWS::S3::BucketPolicy'
	
	  Properties:
	
	    Bucket: !Ref S3Bucket
	
	    PolicyDocument: !Sub |-
	
	      {
	
	 "Version": "2008-10-17",
	
	                  "Id": "restricthttp",
	
	 "Statement": [</pre>
		AWS DevOps

Task	Description	Skills required

	``
	                    {
	
	  "Sid": "denyhttp",
	
	  "Effect": "Deny",
	
	  "Principal": {
	
	      "AWS": "*"
	
	  },
	
	  "Action": "s3:*",
	
	  "Resource": ["arn:aws
	:s3:::${S3Bucket}"
	,"arn:aws:s3:::${S
	3Bucket}/*"],
	
	  "Condition": {
	
	      "Bool": {
	
	             "aws:Secu
	reTransport": "false"
	
	      }
	
	  }
	``	

Task	Description	Skills required

	``
	}
	]
	}
	``
	Note
	This S3 bucket policy has a deny statement that restricts API calls that are not secure.	

Update the key policy.	1. On the AWS CloudFormation console, choose **Stacks**, choose the name of the stack that you created earlier, choose the **Template** pane, and then choose **View in Designer**. 2. Modify the template's KMS resource to include the key policy that allows administrators to administer the KMS key. 3. Choose **Create stack**, choose **Next**, and then complete the wizard according to your requirements. For more information, see [Key policies in AWS KMS]() in the AWS KMS documentation.	AWS administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add resource-level tags.</td><td>1. On the AWS CloudFormation console, choose Stacks, choose the name of the stack that you created earlier, choose the Template pane, and then choose View in Designer.
2. Add the following snippet to the Amazon S3 resource <code>Properties</code> section of the template, and then choose Create stack and complete the wizard:
<pre>Tags:

  - Key: createdBy

    Value: Cloudformation</pre></td><td>AWS DevOps</td></tr>
</table>

Related resources

- [Bringing existing resources into AWS CloudFormation management](#)
- [AWS re:Invent 2017: Deep dive on AWS CloudFormation](#) (video)

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip](#)

Synchronize data between Amazon EFS file systems in different AWS Regions by using AWS DataSync

Sarat Chandra Pothula and Aditya Ambati, Amazon Web Services

Summary

This solution provides a robust framework for efficient and secure data synchronization between Amazon Elastic File System (Amazon EFS) instances in different AWS Regions. This approach is scalable and provides controlled, cross-Region data replication. This solution can enhance your disaster recovery and data redundancy strategies.

By using the AWS Cloud Development Kit (AWS CDK), this pattern uses as an infrastructure as code (IaC) approach to deploy the solution resources. The AWS CDK application deploys the essential AWS DataSync, Amazon EFS, Amazon Virtual Private Cloud (Amazon VPC), and Amazon Elastic Compute Cloud (Amazon EC2) resources. This IaC provides a repeatable and version-controlled deployment process that is fully aligned with AWS best practices.

Prerequisites and limitations

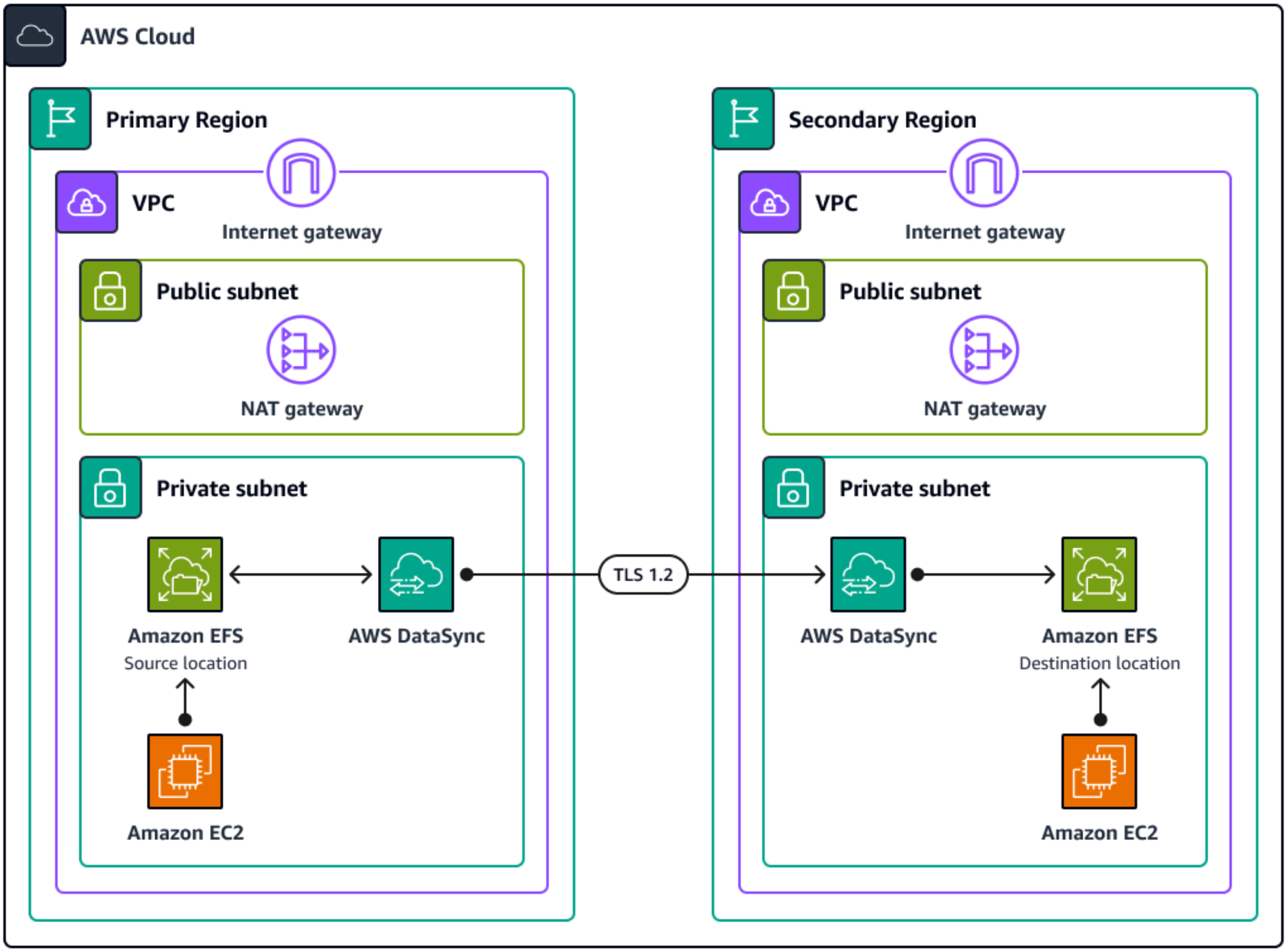
Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2.9.11 or later, installed and configured
- AWS CDK version 2.114.1 or later, installed and bootstrapped
- NodeJS version 20.8.0 or later, installed

Limitations

- The solution inherits limitations from DataSync and Amazon EFS, such as data transfer rates, size limitations, and regional availability. For more information, see AWS DataSync quotas and Amazon EFS quotas.
- This solution supports Amazon EFS only. DataSync supports other AWS services, such as Amazon Simple Storage Service (Amazon S3) and Amazon FSx for Lustre. However, this solution requires modification to synchronize data with these other services.

Architecture



This solution deploys the following AWS CDK stacks:

- **Amazon VPC stack –** This stack sets up virtual private cloud (VPC) resources, including subnets, an internet gateway, and a NAT gateway in both the primary and secondary AWS Regions.
- **Amazon EFS stack –** This stack deploys Amazon EFS file systems into the primary and secondary Regions and connects them to their respective VPCs.
- **Amazon EC2 stack –** This stack launches EC2 instances in the primary and secondary Regions. These instances are configured to mount the Amazon EFS file system, which allows them to access the shared storage.
- **DataSync location stack –** This stack uses a custom construct called `DataSyncLocationConstruct` to create DataSync location resources in the primary and secondary Regions. These resources define endpoints for data synchronization.

- **DataSync task stack –** This stack uses a custom construct called `DataSyncTaskConstruct` to create a DataSync task in the primary Region. This task is configured to synchronize data between the primary and secondary Regions by using the DataSync source and destination locations.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS DataSync is an online data transfer and discovery service that helps you move files or object data to, from, and between AWS storage services.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Code repository

The code for this pattern is available in the GitHub Amazon EFS Cross-Region DataSync Project repository.

Best practices

Follow the best practices described in Best practices for using the AWS CDK in TypeScript to create IaC projects.

Epics

Deploy the AWS CDK app

Task	Description	Skills required
Clone the project repository.	Enter the following command to clone the [Amazon EFS Cross-Region DataSync Project](https://github.com/aws-samples/aws-efs-crossregion-datasync) repository. `git clone https://github.com/aws-samples/aws-efs-crossregion-datasync.git`	AWS DevOps
Install the npm dependencies.	Enter the following command. `npm ci`	AWS DevOps

Choose the primary and secondary Regions.	In the cloned repository, navigate to the `src/infa` directory. In the `Launcher.ts` file, update the `PRIMARY_AWS_REGION` and `SECONDARY_AWS_REGION` values. Use the corresponding Region codes.
	`const primaryRegion = { account: account, region: '<PRIMARY_AWS_REGION>' };`
	`const secondaryRegion = { account: account, region: '<SECONDARY_AWS_REGION>' };`
		AWS DevOps

Task	Description	Skills required
Bootstrap the environment.	Enter the following command to bootstrap the AWS account and AWS Region that you want to use. `cdk bootstrap <aws_account>/<aws_region>` For more information, see [Bootstrapping](#) in the AWS CDK documentation.	AWS DevOps
List the AWS CDK stacks.	Enter the following command to view a list of the AWS CDK stacks in the app. `cdk ls`	AWS DevOps
Synthesize the AWS CDK stacks.	Enter the following command to produce an AWS CloudFormation template for each stack defined in the AWS CDK app. `cdk synth`	AWS DevOps
Deploy the AWS CDK app.	Enter the following command to deploy all of the stacks to your AWS account, without requiring manual approval for any changes. `cdk deploy --all --require-approval never`	AWS DevOps

Validate the deployment

Task	Description	Skills required
Log in to the EC2 instance in the primary Region.	1. Using Session Manager, a capability of AWS Systems Manager, log in to the EC2 instance in the primary Region. For instructions, see [Connect to your Linux instance with AWS Systems Manager Session Manager](). 2. Change directories to the Amazon EFS mount path. `cd /mnt/efs`	AWS DevOps

Create a temporary file.	Enter the following command to create a temporary file in the Amazon EFS mount path.
	 `sudo dd if=/dev/zero \`
	`of=tmptst.dat \`
	`bs=1G \`
	`seek=5 \`
	`count=0`
	
	`ls -lrt tmptst.dat`
		AWS DevOps

Start the DataSync task.	Enter the following command to replicate the temporary file from the primary Region to the secondary Region, where `<ARN-task>` is the Amazon Resource Name (ARN) of your DataSync task.	AWS DevOps

Task	Description	Skills required

	``
	aws datasync start-tas
	k-execution \
	    --task-arn <ARN-
	task>
	``
	
	The command returns the ARN of the task execution in the following format.
	
	`arn:aws:datasync:<region>:<account-ID>:task/task-execution/<exec-ID>`	

Check the status of the data transfer.	Enter the following command to describe the DataSync execution task, where `<ARN-task-execution>` is the ARN of the task execution. ``` aws datasync describe- task-execution \ --task-execution-a rn <ARN-task-executio n> ``` The DataSync task is complete when `PrepareStatus` , `TransferStatus` , and `VerifyStatus` all have the value SUCCESS.	AWS DevOps

Task	Description	Skills required
Log in to the EC2 instance in the secondary Region.	1. Using Session Manager, a capability of AWS Systems Manager, log in to the EC2 instance in the secondary Region. For instructions, see [Connect to your Linux instance with AWS Systems Manager Session Manager](). 2. Change directories to the Amazon EFS mount path. `cd /mnt/efs`	AWS DevOps
Validate the replication.	Enter the following command to verify that the temporary file exists in the Amazon EFS file system. `ls -lrt tmptst.dat`	AWS DevOps

Related resources

AWS documentation

- AWS CDK API Reference
- Configuring AWS DataSync transfers with Amazon EFS
- Troubleshooting issues with AWS DataSync transfers

Other AWS resources

- AWS DataSync FAQs

Test AWS infrastructure by using LocalStack and Terraform Tests

Ivan Girardi and Ioannis Kalyvas, Amazon Web Services

Summary

This pattern helps you locally test infrastructure as code (IaC) for AWS in Terraform without the need to provision infrastructure in your AWS environment. It integrates the Terraform Tests framework with LocalStack. The LocalStack Docker container provides a local development environment that emulates various AWS services. This helps you test and iterate on infrastructure deployments without incurring costs in the AWS Cloud.

This solution provides the following benefits:

- **Cost optimization –** Running tests against LocalStack eliminates the need to use AWS services. This prevents you from incurring costs that are associated with creating, operating, and modifying those AWS resources.
- **Speed and efficiency –** Testing locally is also typically faster than deploying the AWS resources. This rapid feedback loop accelerates development and debugging. Because LocalStack runs locally, you can develop and test your Terraform configuration files without an internet connection. You can debug Terraform configuration files locally and receive immediate feedback, which streamlines the development process.
- **Consistency and reproducibility –** LocalStack provides a consistent environment for testing. This consistency helps make sure that tests yield the same results, regardless of external AWS changes or network issues.
- **Isolation –** Testing with LocalStack prevents you from accidentally affecting live AWS resources or production environments. This isolation makes it safe to experiment and test various configurations.
- **Automation –** Integration with a continuous integration and continuous delivery (CI/CD) pipeline helps you automatically test Terraform configuration files. The pipeline thoroughly tests the IaC before deployment.
- **Flexibility –** You can simulate different AWS Regions, AWS accounts, and service configurations to match your production environments more closely.

Prerequisites and limitations

Prerequisites

- Install Docker
- Enable access to the default Docker socket (`/var/run/docker.sock`). For more information, see the LocalStack documentation.
- Install Docker Compose
- Install Terraform version 1.6.0 or later
- Install Terraform CLI
- Configure the Terraform AWS Provider
- (Optional) Install and configure the AWS Command Line Interface (AWS CLI). For an example of how to use the AWS CLI with LocalStack, see the GitHub Test AWS infrastructure using LocalStack and Terraform Tests repository.

Limitations

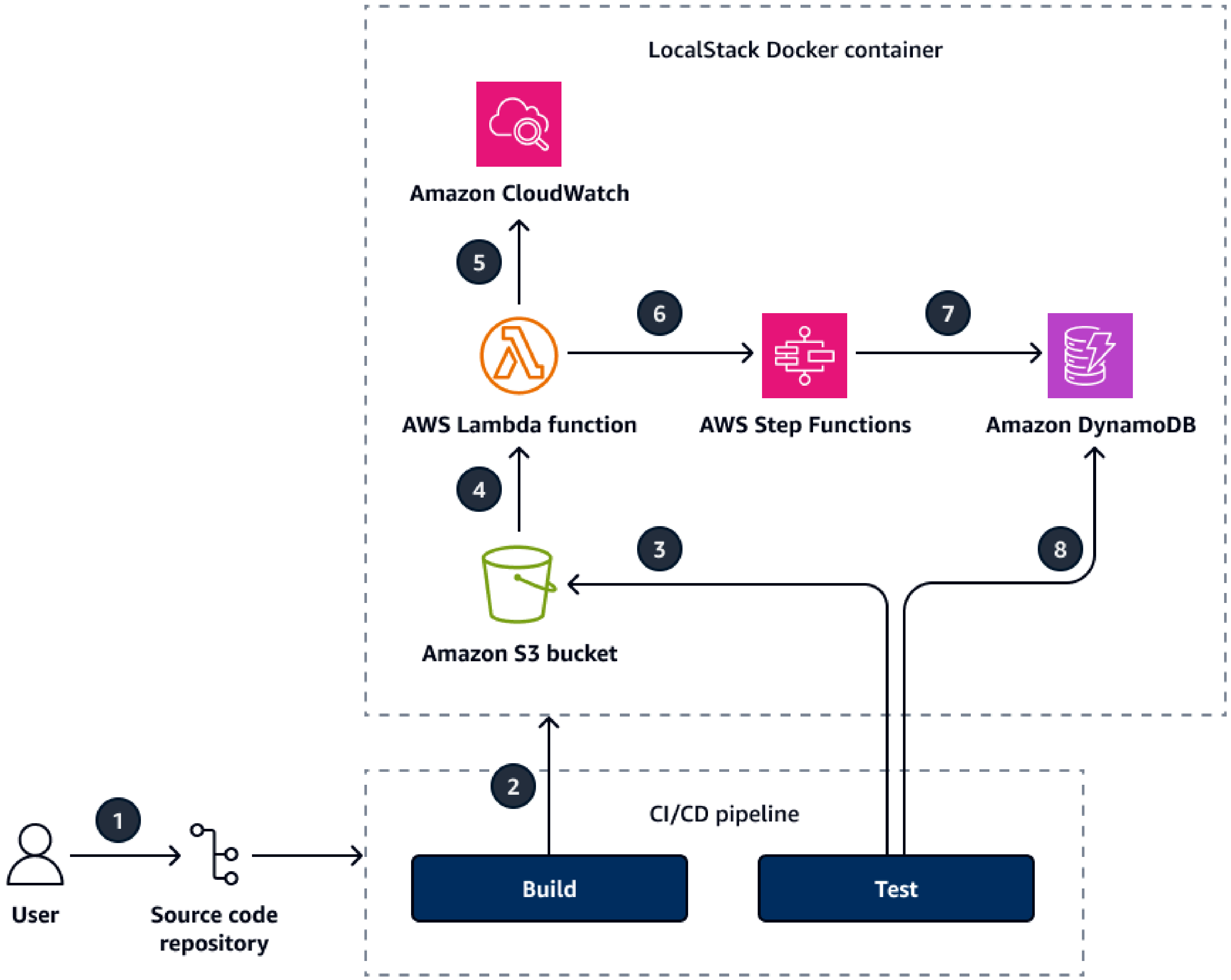
- This pattern provides explicit examples for testing Amazon Simple Storage Service (Amazon S3), AWS Lambda, AWS Step Functions, and Amazon DynamoDB resources. However, you can extend this solution to include additional AWS resources.
- This pattern provides instructions to run Terraform Tests locally, but can you can integrate testing into any CI/CD pipeline.
- This pattern provides instructions for using the LocalStack Community image. If you're using the LocalStack Pro image, see the LocalStack Pro documentation.
- LocalStack provides emulation services for different AWS APIs. For a complete list, see AWS Service Feature Coverage. Some advanced features might require a subscription for LocalStack Pro.

Architecture

The following diagram shows the architecture for this solution. The primary components are a source code repository, a CI/CD pipeline, and a LocalStack Docker container. The LocalStack Docker Container hosts the following AWS services locally:

- An Amazon S3 bucket for storing files
- Amazon CloudWatch for monitoring and logging
- An AWS Lambda function for running serverless code
- An AWS Step Functions state machine for orchestrating multi-step workflows

- An Amazon DynamoDB table for storing NoSQL data



The diagram shows the following workflow:

1. You add and commit a Terraform configuration file to the source code repository.
2. The CI/CD pipeline detects the changes and initiates a build process for static Terraform code analysis. The pipeline builds and runs the LocalStack Docker container. Then the pipeline starts the test process.
3. The pipeline uploads an object into an Amazon S3 bucket that is hosted in the LocalStack Docker container.
4. Uploading the object invokes an AWS Lambda function.

5. The Lambda function stores the Amazon S3 event notification in a CloudWatch log.
6. The Lambda function starts an AWS Step Functions state machine.
7. The state machine writes the name of the Amazon S3 object into a DynamoDB table.
8. The test process in the CI/CD pipeline verifies that the name of the uploaded object matches the entry in the DynamoDB table. It also verifies that the S3 bucket is deployed with the specified name and that the AWS Lambda function has been successfully deployed.

Tools

AWS services

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- Docker Compose is a tool for defining and running multi-container applications.
- LocalStack is a cloud service emulator that runs in a single container. By using LocalStack, you can run workloads on your local machine that use AWS services, without connecting to the AWS Cloud.
- Terraform is an IaC tool from HashiCorp that helps you create and manage cloud and on-premises resources.
- Terraform Tests helps you validate Terraform module configuration updates through tests that are analogous to integration or unit testing.

Code repository

The code for this pattern is available in the GitHub Test AWS infrastructure using LocalStack and Terraform Tests repository.

Best practices

- This solution tests AWS infrastructure that is specified in Terraform configuration files, and it does not deploy those resources in the AWS Cloud. If you want to deploy the resources, follow the principle of least-privilege (IAM documentation) and properly configure the Terraform backend (Terraform documentation).
- When integrating LocalStack in a CI/CD pipeline, we recommend that you don't run the LocalStack Docker container in privilege mode. For more information, see Runtime privilege and Linux capabilities (Docker documentation) and Security for self-managed runners (GitLab documentation).

Epics

Deploy the solution

Task	Description	Skills required
Clone the repository.	In a bash shell, enter the following command. This clones the Test AWS infrastructure using LocalStack and Terraform Tests repository from GitHub: `git clone https://github.com/aws-samples/localstack-terraform-test.git`	DevOps engineer
Run the LocalStack container.	1. Enter the following command to navigate into the cloned repository:	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>cd localstack-terrafo
rm-test</code></pre>
2. Enter the following command to start the LocalStack Docker container in detached mode:
<pre><code>docker-compose up -d</code></pre>
3. Wait until the LocalStack Docker container is operational.</td><td></td></tr>
<tr><td>Initialize Terraform.</td><td>Enter the following command to initialize Terraform:
<pre><code>terraform init</code></pre></td><td>DevOps engineer</td></tr>
<tr><td>Run Terraform Tests.</td><td>1. Enter the following command to run Terraform Tests:
<pre><code>terraform test</code></pre>
2. Verify that all tests completed successfully. The output should be similar to the following:
<pre><code>Success! 3 passed, 0
 failed.</code></pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
Clean up resources.	Enter the following command to destroy the LocalStack container: `docker-compose down`	DevOps engineer

Troubleshooting

Issue	Solution

`Error: reading DynamoDB Table Item (Files|README.md): empty` result when running the `terraform test` command.	1. Reenter the `terraform test` command.
	2. If that doesn't resolve the error, edit the **main.tf** file to increase the sleep timeout to a value greater than 15 seconds:
	`resource "time_sleep" "wait" {`
	`  create_duration = "15s"`
	`  triggers = {`
	`    s3_object = local.key_json`
	`  }`
	`}`

Related resources

- Getting started with Terraform: Guidance for AWS CDK and AWS CloudFormation experts (AWS Prescriptive Guidance)
- Best practices for using the Terraform AWS Provider (AWS Prescriptive Guidance)
- Terraform CI/CD and testing on AWS with the new Terraform Test Framework (AWS blog post)
- Accelerating software delivery using LocalStack Cloud Emulator from AWS Marketplace (AWS blog post)

Additional information

Integration with GitHub Actions

You can integrate LocalStack and Terraform Tests in a CI/CD pipeline by using GitHub Actions. For more information, see the [GitHub Actions documentation](). The following is a sample GitHub Actions configuration file:

```
name: LocalStack Terraform Test

on:
  push:
    branches:
      - '**'

  workflow_dispatch: {}

jobs:
  localstack-terraform-test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4

    - name: Build and Start LocalStack Container
      run: |
        docker compose up -d

    - name: Setup Terraform
      uses: hashicorp/setup-terraform@v3
      with:
        terraform_version: latest

    - name: Run Terraform Init and Validation
      run: |
        terraform init
        terraform validate
        terraform fmt --recursive --check
        terraform plan
        terraform show

    - name: Run Terraform Test
      run: |
```

```
        terraform test

    - name: Stop and Delete LocalStack Container
      if: always()
      run: docker compose down
```

Upgrade SAP Pacemaker clusters from ENSA1 to ENSA2

Gergely Cserdi and Balazs Sandor Skublics, Amazon Web Services

Summary

This pattern explains the steps and considerations for upgrading an SAP Pacemaker cluster that is based on Standalone Enqueue Server (ENSA1) to ENSA2. The information in this pattern applies to both SUSE Linux Enterprise Server (SLES) and Red Hat Enterprise Linux (RHEL) operating systems.

Pacemaker clusters on SAP NetWeaver 7.52 or S/4HANA 1709 and earlier versions run on an ENSA1 architecture and are configured specifically for ENSA1. If you run your SAP workloads on Amazon Web Services (AWS) and you're interested in moving to ENSA2, you might find that the SAP, SUSE, and RHEL documentation doesn't provide comprehensive information. This pattern describes the technical steps required to reconfigure SAP parameters and Pacemaker clusters to upgrade from ENSA1 to ENSA2. It provides examples of SUSE systems, but the concept is the same for RHEL clusters.

Notes: ENSA1 and ENSA2 are concepts that pertain to SAP applications only, so the information in this pattern doesn't apply to SAP HANA or other types of clusters.

Technically, ENSA2 can be used with or without Enqueue Replicator 2. However, high availability (HA) and failover automation (through a cluster solution) require Enqueue Replicator 2. This pattern uses the term *ENSA2 clusters* to refer to clusters with Standalone Enqueue Server 2 and Enqueue Replicator 2.

Prerequisites and limitations

Prerequisites

- A working ENSA1-based cluster that uses Pacemaker and Corosync on SLES or RHEL.
- At least two Amazon Elastic Compute Cloud (Amazon EC2) instances where the (ABAP) SAP Central Services (ASCS/SCS) and Enqueue Replication Server (ERS) instances are running.
- Knowledge of managing SAP applications and clusters.
- Access to the Linux environment as root user.

Limitations

- ENSA1-based clusters support a two-node architecture only.
- ENSA2-based clusters cannot be deployed to SAP NetWeaver versions before 7.52.
- EC2 instances in clusters should be in different AWS Availability Zones.

Product versions

- SAP NetWeaver version 7.52 or later
- Starting with S/4HANA 2020, only ENSA2 clusters are supported
- Kernel 7.53 or later, which supports ENSA2 and Enqueue Replicator 2
- SLES for SAP Applications version 12 or later
- RHEL for SAP with High Availability (HA) version 7.9 or later

Architecture

Source technology stack

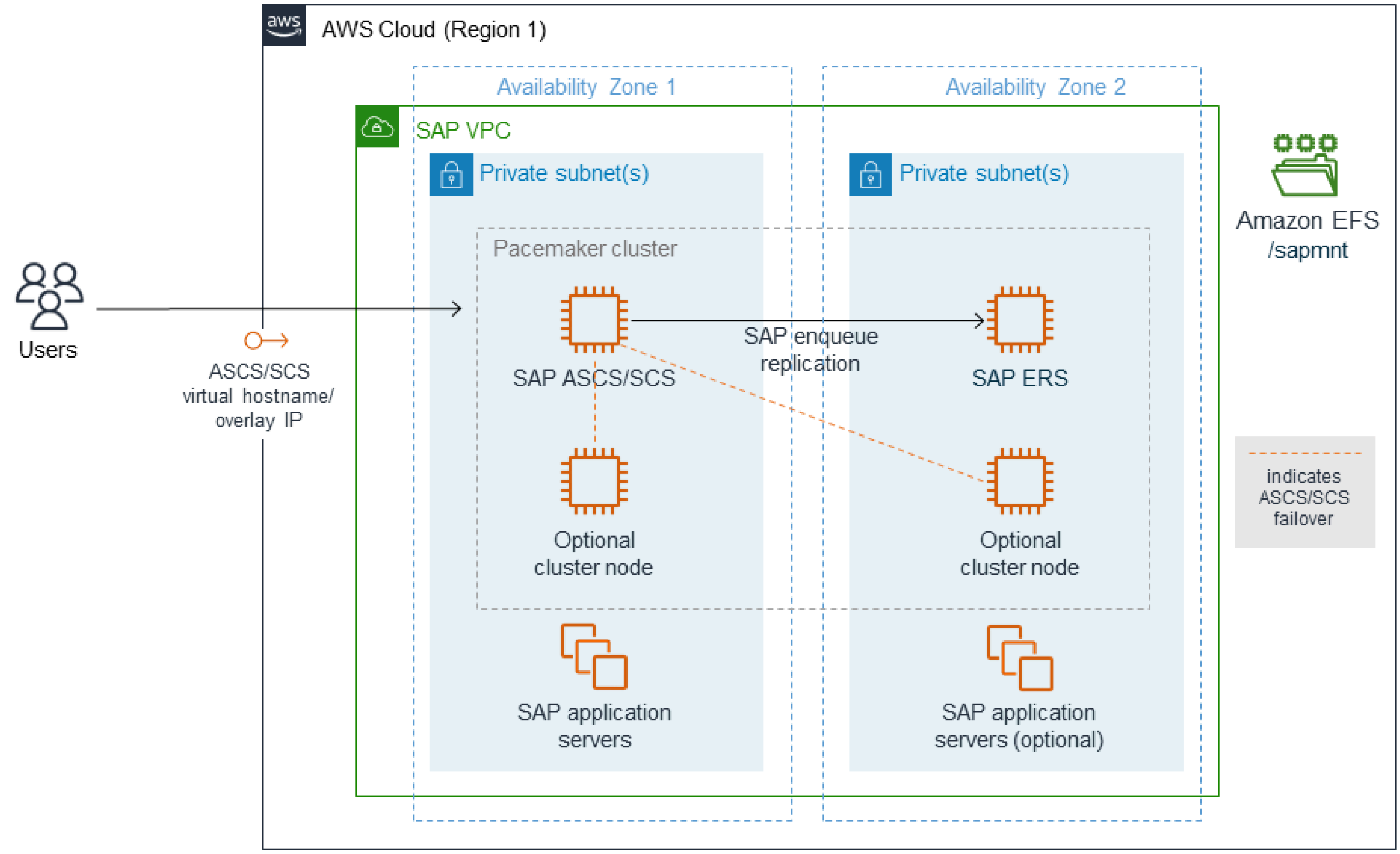
- SAP NetWeaver 7.52 with SAP Kernel 7.53 or later
- SLES or RHEL operating system

Target technology stack

- SAP NetWeaver 7.52 with SAP Kernel 7.53 or later, including S/4HANA 2020 with ABAP platform
- SLES or RHEL operating system

Target architecture

The following diagram shows an HA configuration of ASCS/SCS and ERS instances based on an ENSA2 cluster.



Comparison of ENSA1 and ENSA2 clusters

SAP introduced ENSA2 as the successor to ENSA1. An ENSA1-based cluster supports a two-node architecture where the ASCS/SCS instance fails over to ERS when an error occurs. This limitation stems from how the ASCS/SCS instance regains the lock table information from the shared memory of the ERS node after failover. ENSA2-based clusters with Enqueue Replicator 2 eliminate this limitation, because the ASCS/SCS instance can collect the lock information from the ERS instance over the network. ENSA2-based clusters can have more than two nodes, because the ASCS/SCS instance is no longer required to fail over to the ERS node. (However, in a two-node ENSA2 cluster environment, the ASCS/SCS instance will still fail over to the ERS node because there are no other nodes in the cluster to fail over to.) ENSA2 is supported starting with SAP Kernel 7.50 with some limitations. For HA setup that supports Enqueue Replicator 2, the minimum requirement is NetWeaver 7.52 (see SAP OSS Note 2630416). S/4HANA 1809 comes with ENSA2 architecture recommended by default, whereas S/4HANA supports only ENSA2 starting with version 2020.

Automation and scale

The HA cluster in the target architecture makes ASCS fail over to other nodes automatically.

Scenarios for moving to ENSA2-based clusters

There are two main scenarios for upgrading to ENSA2-based clusters:

- Scenario 1: You choose to upgrade to ENSA2 without an accompanying SAP upgrade or S/4HANA conversion, assuming that your SAP release and Kernel version support ENSA2.
- Scenario 2: You move to ENSA2 as part of an upgrade or conversion (for example, to S/4HANA 1809 or later) by using SUM.

The Epics section covers the steps for these two scenarios. The first scenario requires you to manually set up SAP-related parameters before you change the cluster configuration for ENSA2. In the second scenario, the binaries and SAP-related parameters are deployed by SUM, and your only remaining task is to update the cluster configuration for HA. We still recommend that you validate SAP parameters after you use SUM. In most cases, S/4HANA conversion is the main reason for a cluster upgrade.

Tools

- For OS package managers, we recommend the Zypper (for SLES) or YUM (for RHEL) tools.
- For cluster management, we recommend **crm** (for SLES) or **pcs** (for RHEL) shells.
- SAP instance management tools such as SAPControl.
- (Optional) SUM tool for S/4HANA conversion upgrade.

Best practices

- For best practices for using SAP workloads on AWS, see the SAP Lens for the AWS Well-Architected Framework.
- Consider the number of cluster nodes (odd or even) in your ENSA2 multi-node architecture.
- Set up the ENSA2 cluster for SLES 15 in alignment with the SAP S/4-HA-CLU 1.0 certification standard.
- Always save or back up your existing cluster and application state before upgrading to ENSA2.

Epics

Configure SAP parameters manually for ENSA2 (scenario 1 only)

Task	Description	Skills required
Configure the parameters in the default profile.	If you want to upgrade to ENSA2 while staying on the same SAP release or if your target release defaults to ENSA1, set the parameters in the default profile (DEFAULT.PFL file) to the following values. ```enq/enable=TRUE enq/serverhost=sapascsvirt enq/serverinst=10 (instance number of ASCS/SCS instance) enque/process_location=REMOTESA enq/replicatorhost=sapersvirt enq/replicatorinst=11 (instance number of ERS instance)``` where `sapascsvirt` is the virtual hostname for the ASCS instances, and `sapersvirt` is the virtual hostname for the ERS instances. You can change these to fit your target environment.	SAP

Task	Description	Skills required
	Note To use this upgrade option, your SAP release and Kernel version must support ENSA2 and Enqueue Replicator 2.	

Task	Description	Skills required

Configure the ASCS/SCS instance profile.	If you want to upgrade to ENSA2 while staying on the same SAP release or if your target release defaults to ENSA1, set the following parameters in the ASCS/SCS instance profile.
	
	The section of the profile where ENSA1 is defined looks something like the following.
	
	`#-----------------------------------------------------------------------------------`
	`Start SAP enqueue server`
	`#-----------------------------------------------------------------------------------`
	`_EN = en.sap$(SAPSYSTEMNAME)$(INSTANCE_NAME)`
	`Execute_04 = local rm -f $(_EN)`
	`Execute_05 = local ln -s -f $(DIR_EXECUTABLE)/enserver$(FT_EXE) $(_EN)`
	`Start_Program_01 = local $(_EN) pf=$(_PF)`
	
	To reconfigure this section for ENSA2:
		SAP

Task	Description	Skills required

	1. Change the _EN program prefix to _ENQ based on the latest information from SAP (OSS Note 2501860; requires an [SAP ONE Support Launchpad user account](#)).
	2. Change the binary for enqueue server from `enserver` to `enq_server` .
	3. Set the new parameter `enq/server/replication/enable` to TRUE.
	4. Make sure that `Autostart = 0`.
	
	This profile section would look something like the following after your changes.
	
	`#-----------------------------------------------------`
	`Start SAP enqueue server`
	`#-------------------------------------------------------`
	`_ENQ = enq.sap$(SAPSYSTEMNAME)$(INSTANCE_NAME)`
	`Execute_04 = local rm -f $(_ENQ)`	

Task	Description	Skills required
	``` Execute_05 = local ln - s -f $(DIR_EXECUTABLE)/ enq_server$(FT_EXE) $(_ENQ) Start_Program_01 = local $(_ENQ) pf= $(_PF) ... enq/server/replic ation/enable = TRUE Autostart = 0 ``` **Important** _ENQ must not have the restart option enabled. If `RestartProgram_01` is set for _ENQ, change it to `StartProgram_01`. This prevents SAP from restarting the service or interfering with cluster-managed resources.	

Task	Description	Skills required

Configure the ERS profile.	If you want to upgrade to ENSA2 while staying on the same SAP release or if your target release defaults to ENSA1, set the following parameters in the ERS instance profile.
	
	Find the section where the enqueue replicator is defined. It will be similar to the following.
	
	`#----------------------------------------------------- Start enqueue replication server`
	`#-----------------------------------------------------`
	`_ER = er.sap$(SAPSYSTEMNAME)$(INSTANCE_NAME)`
	`Execute_03 = local rm -f $(_ER)`
	`Execute_04 = local ln -s -f $(DIR_EXECUTABLE)/enrepserver$(FT_EXE) $(_ER)`
	`Start_Program_00 = local $(_ER) pf=$(_PF) NR=$(SCSID)`
	
	To reconfigure this section for Enqueue Replicator 2:
		SAP

Task	Description	Skills required

	1. Change the _ER program prefix to _ENQR based on the latest notes from SAP (OSS Note 2501860; requires an SAP ONE Support Launchpad user account).
	2. Change the binary for the enqueue replicator to `enq_replicator` instead of `enrepserver` .
	3. Make sure that `Autostart = 0`.
	
	This profile section should look something like the following after your changes.
	
	`#-----------------------------------------------------Start enqueue replication server`
	`#----------------------------------------------------`
	`_ENQR = enqr.sap$(SAPSYSTEMNAME)$(INSTANCE_NAME)`
	`Execute_01 = local rm -f $(_ENQR)`
	`Execute_02 = local ln -s -f $(DIR_EXECUTABLE)/enq_replicator$(FT_EXE) $(_ENQR)`	

Task	Description	Skills required

	<pre>Start_Program_00 =
	 local $(_ENQR) pf=
	$(_PF) NR=$(SCSID)
	…
	Autostart = 0</pre> **Important** _ENQR must not have the restart option enabled. If `RestartProgram_01` is set for _ENQR, change it to `StartProgram_01` . This prevents SAP from restarting the service or interfering with cluster-managed services.	

Task	Description	Skills required

Restart SAP Start Services.	After you change the profiles described previously in this epic, restart SAP Start Services for both ASCS/SCS and ERS.
	
	`sapcontrol -nr 10 -function RestartService SCT`
	
	`sapcontrol -nr 11 -function RestartService SCT`
	
	where SCT refers to the SAP system ID, and assuming that 10 and 11 are the instance numbers for ASCS/SCS and ERS instances, respectively.
		SAP

Reconfigure the cluster for ENSA2 (required for both scenarios)

Task	Description	Skills required
Verify version numbers in SAP resource agents.	When you use SUM to upgrade SAP to S/4HANA 1809 or later, SUM handles the parameter changes in the SAP profiles. Only the cluster requires manual adjustment. However, we recommend that you verify the parameter settings before you make any changes to the cluster.	AWS systems administrator

Task	Description	Skills required

	Note
	
	The examples in this epic assume that you're using the SUSE operating system. If you're using RHEL, you will need to use tools such as YUM and the **pcs** shell instead of Zypper and **crm**.
	
	Check both nodes in the architecture to confirm that the `resource-agents` package matches the minimum version recommended by SAP. For SLES, check SAP OSS Note 2641019. For RHEL, check SAP OSS Note 2641322. (SAP Notes require an [SAP ONE Support Launchpad user account](https://me.sap.com/).)
	
	`sapers:sctadm 23> zypper search -s -i resource-agents`
	`Loading repository data...`
	`Reading installed packages...`
	`S | Name | Type | Version | Arch | Repository`
	`--+-------------`
	`----+---------+---`	

Task	Description	Skills required
	``` ------------------ ---------------+-- ------+----------- ------------------ i | resource-agents | package | 4.8.0+git 30.d0077df0-150300 .8.28.1 | x86_64 | SLE-Product-HA15-SP3- Updates ``` Update the `resource-agents` version if necessary.	
Back up cluster configuration.	Back up the CRM cluster configuration as follows. `crm configure show > /tmp/cluster_config_backup.txt`	AWS systems administrator
Set maintenance mode.	Set the cluster to maintenance mode. `crm configure property maintenance-mode="true"`	AWS systems administrator

Task	Description	Skills required

Check cluster configuration.	Check the current cluster configuration.
	
	`crm configure show`
	
	Here is an excerpt from the full output:
	
	`node 1: sapascs`
	`node 2: sapers`
	`...`
	`primitive rsc_sap_SCT_ASCS10 SAPInstance \`
	`operations $id=rsc_sap_SCT_ASCS10-operations \`
	`op monitor interval=120 timeout=60 on-fail=restart \`
	`params InstanceName=SCT_ASCS10_sapascsvirt START_PROFILE="/sapmnt/SCT/profile/SCT_ASCS10_sapascsvirt" \`
	`   AUTOMATIC_RECOVER=false \`
	`meta resource-stickiness=5000 failure-timeout=60 migration-threshold=1 priority=10`
	`primitive rsc_sap_SCT_ERS11 SAPInstance \`
	`operations $id=rsc_sap_SCT_ERS11-operations \`
	`op monitor interval=120 timeout=60 on-fail=restart \`
		AWS systems administrator

Task	Description	Skills required
	``` params InstanceN ame=SCT_ERS11_sape rsvirt START_PRO FILE="/sapmnt/SCT/ profile/SCT_ERS11_ sapersvirt" \ AUTOMATIC_RECOVER= false IS_ERS=true \ meta priority=1000 ... colocation col_sap_S CT_no_both -5000: grp_SCT_ERS11 grp_SCT_ASCS10 location loc_sap_S CT_failover_to_ers rsc_sap_SCT_ASCS10 \ rule 2000: runs_ers_SCT eq 1 order ord_sap_S CT_first_start_asc s Optional: rsc_sap_S CT_ASCS10:start rsc_sap_SCT_ERS11: stop symmetrical=false ... ``` where `sapascsvirt` refers to the virtual hostname for the ASCS instances, `sapersvirt` refers to the virtual hostname for the ERS instances, and SCT refers to the SAP system ID.	

Task	Description	Skills required
Remove failover colocation constraint.	In the previous example, the location constraint `loc_sap_SCT_failover_to_ers` specifies that the ENSA1 feature of ASCS should always follow the ERS instance upon failover. With ENSA2, ASCS should be able to fail over freely to any participating nodes, so you can remove this constraint. `crm configure delete loc_sap_SCT_failover_to_ers`	AWS systems administrator

Task	Description	Skills required
Adjust primitives.	You will also need to make minor changes to the ASCS and ERS SAPInstance primitives. Here is an example of an ASCS SAPInstance primitive that is configured for ENSA1. `primitive rsc_sap_SCT_ASCS10 SAPInstance \ operations $id=rsc_sap_SCT_ASCS10-operations \ op monitor interval=120 timeout=60 on-fail=restart \ params InstanceName=SCT_ASCS10_sapascsvirt START_PROFILE="/sapmnt/SCT/profile/SCT_ASCS10_sapascsvirt" \ AUTOMATIC_RECOVER=false \ meta resource-stickiness=5000 failure-timeout=60 migration-threshold=1 priority=10`  To upgrade to ENSA2, change this configuration to the following.  `primitive rsc_sap_SCT_ASCS10 SAPInstance \`	AWS systems administrator

Task	Description	Skills required

	``
	operations $id=rsc_s
	ap_SCT_ASCS10-oper
	ations \
	op monitor interval=120
	 timeout=60 on-fail=r
	estart \
	params InstanceN
	ame=SCT_ASCS10_sap
	ascsvirt START_PRO
	FILE="/sapmnt/SCT/
	profile/SCT_ASCS10
	_sapascsvirt" \
	   AUTOMATIC_RECOVER=
	false \
	meta resource-stickines
	s=3000
	``
	This is an example of an ERS SAPInstance primitive that is configured for ENSA1.
	``
	primitive rsc_sap_S
	CT_ERS11 SAPInstance \
	operations $id=rsc_s
	ap_SCT_ERS11-opera
	tions \
	op monitor interval=120
	 timeout=60 on-fail=r
	estart \
	params InstanceN
	ame=SCT_ERS11_sape
	rsvirt START_PRO
	FILE="/sapmnt/SCT/
	profile/SCT_ERS11_
	sapersvirt" \
	   AUTOMATIC_RECOVER=
	false IS_ERS=true \
	meta priority=1000
	``	

Task	Description	Skills required

	To upgrade to ENSA2, change this configuration to the following.
	`primitive rsc_sap_SCT_ERS11 SAPInstance \`
	`operations $id=rsc_sap_SCT_ERS11-operations \`
	`op monitor interval=120 timeout=60 on-fail=restart \`
	`params InstanceName=SCT_ERS11_sapersvirt START_PROFILE="/sapmnt/SCT/profile/SCT_ERS11_sapersvirt" \`
	`   AUTOMATIC_RECOVER=false IS_ERS=true`
	
	You can change primitives in various ways. For example, you can revise them in an editor such as vi, as in the following example.
	
	`crm configure edit rsc_sap_SCT_ERS11`	

Task	Description	Skills required
Disable maintenance mode.	Disable maintenance mode on the cluster. `crm configure property maintenance-mode=" false"` When the cluster is out of maintenance mode, it attempts to bring the ASCS and ERS instances online with the new ENSA2 settings.	AWS systems administrator

(Optional) Add cluster nodes

Task	Description	Skills required
Review best practices.	Before you add more nodes, make sure to understand best practices such as whether to use an odd or even number of nodes.	AWS systems administrator
Add nodes.	Adding more nodes involves a series of tasks, such as updating the operating system, installing software packages that match the existing nodes, and making mounts available. You can use the **Prepare Additional Host** option in SAP Software Provisioning Manager (SWPM) to create an SAP-specific baseline of the host. For	AWS systems administrator

Task	Description	Skills required
	more information, see the SAP guides listed in the next section.	

Related resources

SAP and SUSE references

To access SAP Notes, you must have an SAP ONE Support Launchpad user account. For more information, see the SAP Support website.

- SAP Note 2501860 – Documentation for SAP NetWeaver Application Server for ABAP 7.52
- SAP Note 2641019 – Installation of ENSA2 and update from ENSA1 to ENSA2 in SUSE HA environment
- SAP Note 2641322 – Installation of ENSA2 and update from ENSA1 to ENSA2 when using the Red Hat HA solutions for SAP
- SAP Note 2711036 – Usage of the Standalone Enqueue Server 2 in an HA Environment
- Standalone Enqueue Server 2 (SAP documentation)
- SAP S/4 HANA – Enqueue Replication 2 High Availability Cluster - Setup Guide (SUSE documentation)

AWS references

- SAP HANA on AWS: High Availability Configuration Guide for SLES and RHEL
- SAP Lens - AWS Well-Architected Framework

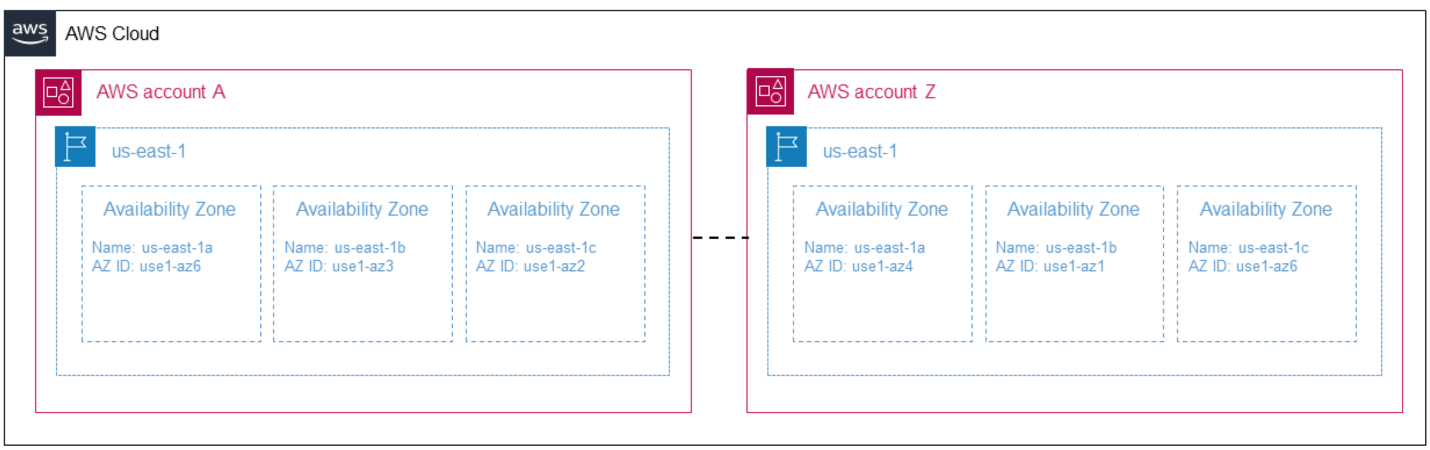
Use consistent Availability Zones in VPCs across different AWS accounts

Adam Spicer, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, an Availability Zone has a name that can vary between your AWS accounts and an Availability Zone ID (AZ ID) that identifies its location. If you use AWS CloudFormation to create virtual private clouds (VPCs), you must specify the Availability Zone's name or ID when creating the subnets. If you create VPCs in multiple accounts, the Availability Zone name is randomized, which means that subnets use different Availability Zones in each account.

To use the same Availability Zone across your accounts, you must map the Availability Zone name in each account to the same AZ ID. For example, the following diagram shows that the `use1-az6` AZ ID is named `us-east-1a` in AWS account A and `us-east-1c` in AWS account Z.



This pattern helps ensure zonal consistency by providing a cross-account, scalable solution for using the same Availability Zones in your subnets. Zonal consistency ensures that your cross-account network traffic avoids cross-Availability Zone network paths, which helps reduce data transfer costs and lower network latency between your workloads.

This pattern is an alternative approach to the AWS CloudFormation AvailabilityZoneId property.

Prerequisites and limitations

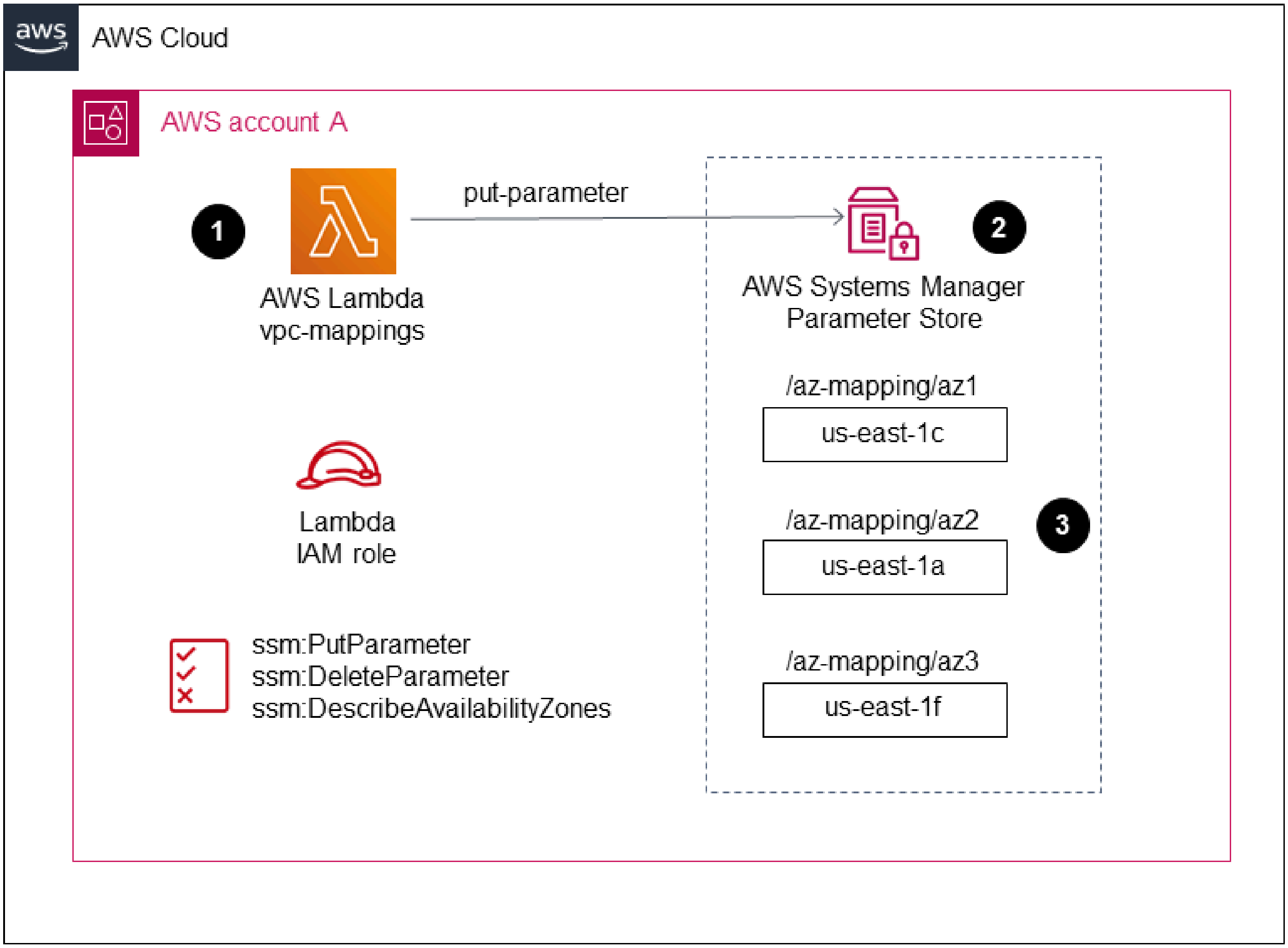
Prerequisites

- At least two active AWS accounts in the same AWS Region.

- Evaluate how many Availability Zones are needed to support your VPC requirements in the Region.
- Identify and record the AZ ID for each Availability Zone that you need to support. For more information about this, see [Availability Zone IDs for your AWS resources]() in the AWS Resource Access Manager documentation.
- An ordered, comma-separated list of your AZ IDs. For example, the first Availability Zone on your list is mapped as `az1`, the second Availbility Zone is mapped as `az2`, and this mapping structure continues until your comma-separated list is fully mapped. There is no maximum number of AZ IDs that can be mapped.
- The `az-mapping.yaml` file from the GitHub [Multi-account Availability Zone mapping]() repository, copied to your local machine

Architecture

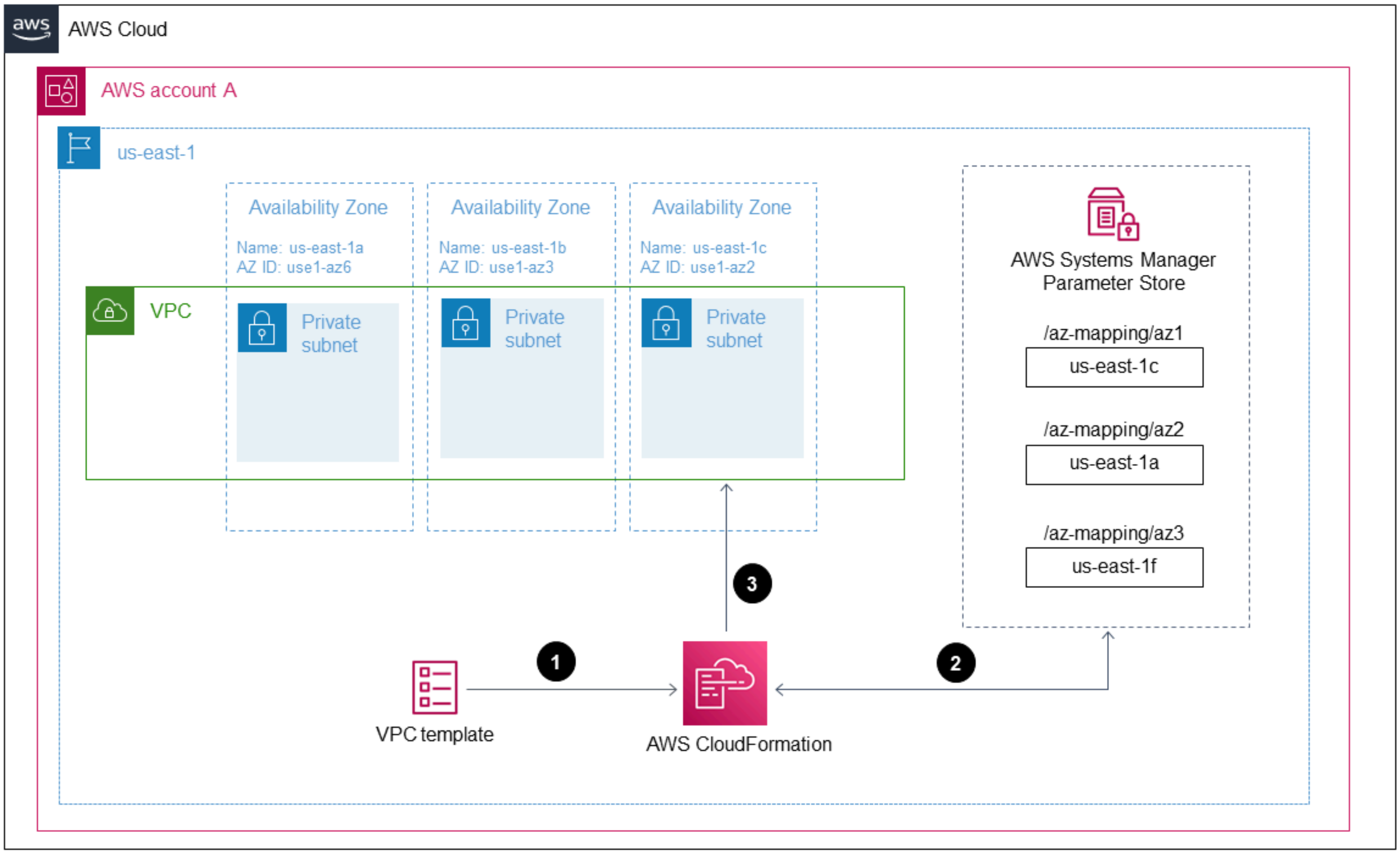
The following diagram shows the architecture that is deployed in an account and that creates AWS Systems Manager Parameter Store values. These Parameter Store values are consumed when you create a VPC in the account.



The diagram shows the following workflow:

1. This pattern’s solution is deployed to all accounts that require zonal consistency for a VPC.
2. The solution creates Parameter Store values for each AZ ID and stores the new Availability Zone name.
3. The AWS CloudFormation template uses the Availability Zone name stored in each Parameter Store value and this ensures zonal consistency.

The following diagram shows the workflow for creating a VPC with this pattern's solution.



The diagram shows the following workflow:

1. Submit a template for creating a VPC to AWS CloudFormation.
2. AWS CloudFormation resolves the Parameter Store values for each Availability Zone and returns the Availability Zone name for each AZ ID.
3. A VPC is created with the correct AZ IDs required for zonal consistency.

After you deploy this pattern's solution, you can create subnets that reference the Parameter Store values. If you use AWS CloudFormation, you can reference the Availability Zone mapping parameter values from the following YAML-formatted sample code:

```
Resources:
    PrivateSubnet1AZ1:
        Type: AWS::EC2::Subnet
        Properties:
            VpcId: !Ref VPC
            CidrBlock: !Ref PrivateSubnetAZ1CIDR
            AvailabilityZone:
```

```
                  !Join
                      - ''
                      - - '{{resolve:ssm:/az-mapping/az1:1}}'
```

This sample code is contained in the `vpc-example.yaml` file from the GitHub [Multi-account Availability Zone mapping] repository. It shows you how to create a VPC and subnets that align to the Parameter Store values for zonal consistency.

Technology stack

- AWS CloudFormation
- AWS Lambda
- AWS Systems Manager Parameter Store

Automation and scale

You can deploy this pattern to all your AWS accounts by using AWS CloudFormation StackSets or the Customizations for AWS Control Tower solution. For more information, see Working with AWS CloudFormation StackSets in the AWS Cloudformation documentation and Customizations for AWS Control Tower in the AWS Solutions Library.

After you deploy the AWS CloudFormation template, you can update it to use the Parameter Store values and deploy your VPCs in pipelines or according to your requirements.

Tools

AWS services

- AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- AWS Systems Manager Parameter Store is a capability of AWS Systems Manager. It provides secure, hierarchical storage for configuration data management and secrets management.

Code

The code for this pattern is provided in the GitHub [Multi-account Availability Zone mapping] repository.

Epics

Deploy the az-mapping.yaml file

Task	Description	Skills required
Determine the required Availability Zones for the Region.	1. Determine the AZ IDs that must be consistently used in your Region. 2. Record these AZ IDs in a comma-separated list and in the order that you want them applied in. For example, the first Availabil ity Zone on your list is mapped as `az1` and the second is mapped as `az2`. There is no maximum number of AZ IDs that can be mapped.	Cloud architect
Deploy the az-mapping.yaml file.	Use the `az-mapping.yaml` file to create an AWS CloudFormation stack in all required AWS accounts. In the `AZIds` parameter, use the comma-separated list that you created earlier. We recommend that you use AWS CloudFormation StackSets or the Customiza	Cloud architect

Task	Description	Skills required
	tions for AWS Control Tower Solution.	

Deploy the VPCs in your accounts

Task	Description	Skills required
Customize the AWS CloudFormation templates.	When you create the subnets using AWS CloudFormation, customize the templates to use the Parameter Store values that you created earlier. For a sample template, see the `vpc-example.yaml` file in the GitHub Multi-account Availability Zone mapping repository.	Cloud architect
Deploy the VPCs.	Deploy the customized AWS CloudFormation templates into your accounts. Each VPC in the Region then has zonal consistency in the Availability Zones used for the subnets	Cloud architect

Related resources

- Availability Zone IDs for your AWS resources (AWS Resource Access Manager documentation)
- AWS::EC2::Subnet (AWS CloudFormation documentation)

Use user IDs in IAM policies for access control and automation

Srinivas Ananda Babu and Ram Kandaswamy, Amazon Web Services

Summary

This pattern explains the potential pitfalls of using username-based policies in AWS Identity and Access Management (IAM), the benefits of using user IDs, and how to integrate this approach with AWS CloudFormation for automation.

In the AWS Cloud, the IAM service helps you manage user identities and access control with precision. However, reliance on usernames in IAM policy creation can lead to unforeseen security risks and access control issues. For example, consider this scenario: A new employee, John Doe, joins your team, and you create an IAM user account with the username `j.doe`, which grants them permissions through IAM policies that reference usernames. When John leaves the company, the account is deleted. The trouble begins when a new employee, Jane Doe, joins your team, and the `j.doe` username is recreated. The existing policies now grant Jane Doe the same permissions as John Doe. This creates a potential security and compliance nightmare.

Manually updating each policy to reflect new user details is a time-consuming, error-prone process, especially as your organization grows. The solution is to use a unique and immutable user ID. When you create an IAM user account, AWS assigns the IAM user a unique user ID (or principal ID). You can use these user IDs in your IAM policies to ensure consistent and reliable access control that isn't affected by username changes or reuse.

For example, an IAM policy that uses a user ID might look like this:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::example-bucket",
            "Principal": { "AWS": "arn:aws:iam::123456789012:user/abcdef01234567890" }
        }
      ]
}
```

The benefits of using user IDs in IAM policies include:

- **Uniqueness.** User IDs are unique across all AWS accounts, so they provide correct and consistent permission application.
- **Immutability.** User IDs cannot be changed, so they provide a stable identifier for referencing users in policies.
- **Auditing and compliance.** AWS services often include user IDs in logs and audit trails, which makes it easy to trace actions back to specific users.
- **Automation and integration.** Using user IDs in AWS APIs, SDKs, or automation scripts ensures that processes remain unaffected by username changes.
- **Future-proofing.** Using user IDs in policies from the start can prevent potential access control issues or extensive policy updates.

Automation

When you use infrastructure as code (IaC) tools such as AWS CloudFormation, the pitfalls of username-based IAM policies can still cause issues. The IAM user resource returns the username when you call the `Ref` intrinsic function. As your organization's infrastructure evolves, the cycle of creating and deleting resources, including IAM user accounts, can lead to unintended access control issues if you reuse usernames.

To address this issue, we recommend that you incorporate user IDs into your CloudFormation templates. However, obtaining user IDs for this purpose can be challenging. This is where custom resources can be helpful. You can use CloudFormation custom resources to extend the service's functionality by integrating with AWS APIs or external services. By creating a custom resource that fetches the user ID for a given IAM user, you can make the user ID available within your CloudFormation templates. This approach streamlines the process of referencing user IDs and ensures that your automation workflows remain robust and future-proof.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An IAM role for a cloud administrator to run the AWS CloudFormation template

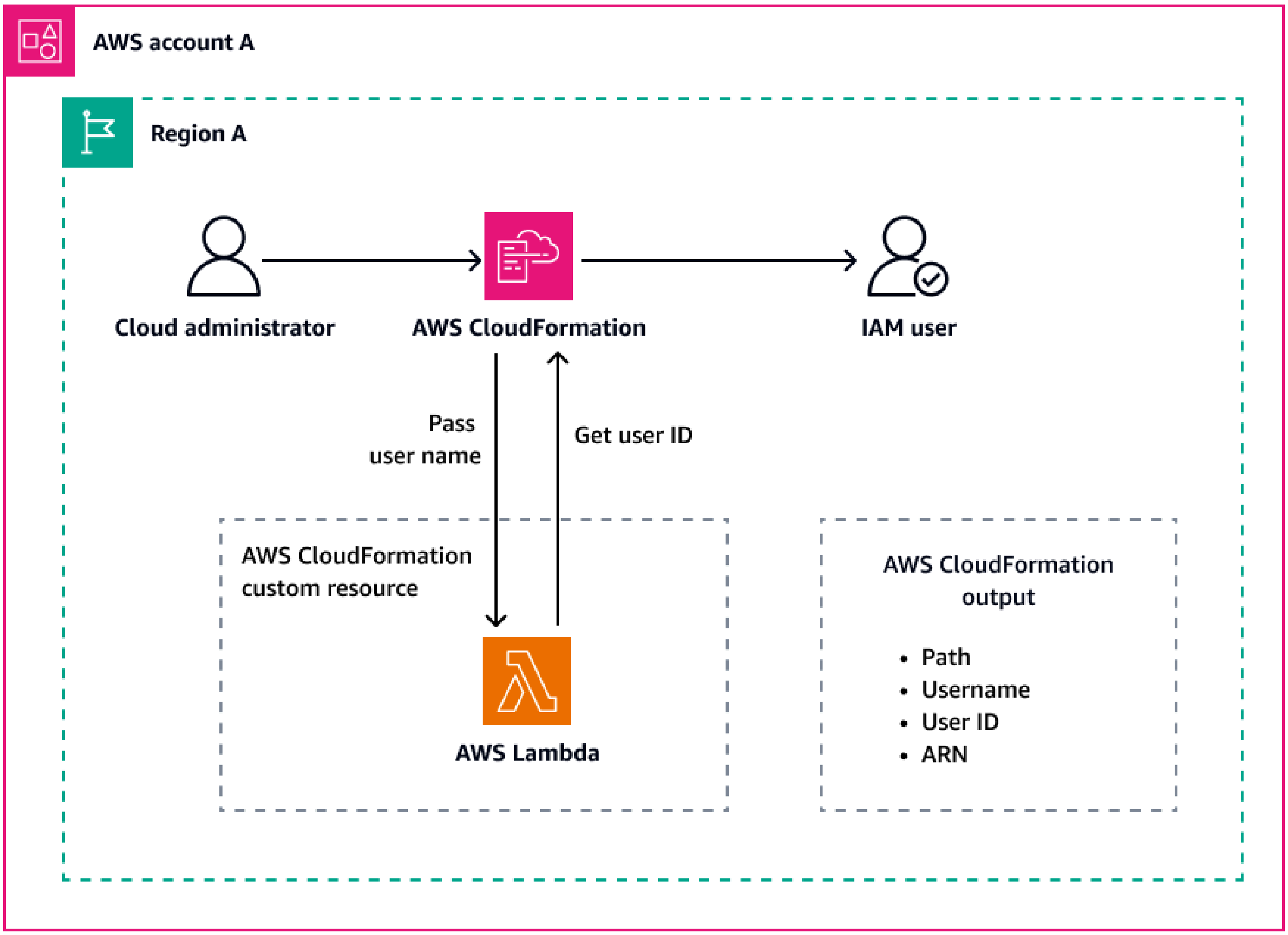
Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

Target architecture

The following diagram shows how AWS CloudFormation uses a custom resource backed by AWS Lambda to retrieve the IAM user ID.



Automation and scale

You can use the CloudFormation template multiple times for different AWS Regions and accounts. You need to run it only once in each Region or account.

Tools

AWS services

- IAM – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run on AWS. You create a template that describes the AWS resources that you want, and CloudFormation takes care of provisioning and configuring those resources for you.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.

Best practices

If you're starting from scratch or planning a greenfield deployment, we strongly recommend that you use AWS IAM Identity Center for centralized user management. IAM Identity Center integrates with your existing identity providers (such as Active Directory or Okta) to federate user identities on AWS, which eliminates the need to create and manage IAM users directly. This approach not only ensures consistent access control but also simplifies user lifecycle management and helps enhance security and compliance across your AWS environment.

Epics

Validate permissions

Task	Description	Skills required
Validate your AWS account and IAM role.	Confirm that you have an IAM role with permissions to deploy CloudFormation templates in your AWS account. If you're planning to use the AWS CLI instead of the	Cloud architect

Task	Description	Skills required
	CloudFormation console to deploy the template in the last step of this procedure , you should also set up temporary credentials to run AWS CLI commands. For instructions, see the IAM documentation.	

Build a CloudFormation template

Task	Description	Skills required
Create a CloudFormation template.	1. Create a CloudFormation template by following the instructions in the CloudFormation documentation. You can use either JSON or YAML format. This pattern assumes that you're using YAML format. 2. Save the template with the name `get_unique_user_id.yaml` .	AWS DevOps, Cloud architect

Add an input parameter for the username.	Add the following code to the `Parameters` section of the CloudFormation template:
	`Parameters:`
	`  NewIamUserName:`
	`    Type: String`
		AWS DevOps, Cloud architect

Task	Description	Skills required
	<pre> Description: Unique username for the new IAM user</pre> This parameter prompts the user for the username.	

Add a custom resource to create an IAM user.	Add the following code to the `Resources` section of the CloudFormation template: <pre>Resources:
	  rNewIamUser:
	    Type: 'AWS::IAM
	::User'
	    Properties:
	      UserName: !Ref
	 NewIamUserName</pre> This code adds a CloudFormation resource that creates an IAM user with the name provided by the `NewIamUserName` parameter.
		AWS DevOps, Cloud architect

Task	Description	Skills required
Add an execution role for the Lambda function.	In this step, you create an IAM role that grants an AWS Lambda function permission to get the IAM `UserId`. Specify the following minimum required permissions for Lambda to run: • `logs:CreateLogStream` • `logs:PutLogEvents` • `CreateLogGroup` • `iam:GetUser` • `AssumeRole` for `lambda.amazonaws.com` For instructions on creating an execution role, see the [Lambda documentation](). You will reference this role in the next step, when you create the Lambda function.	AWS administrator, Cloud architect

Task	Description	Skills required

Add a Lambda function to get the unique IAM `UserId`.	In this step, you define a Lambda function with a Python runtime to get the unique IAM `UserId`. To do this, add the following code to the `Resources` section of the CloudFormation template. Replace `<<ROLENAME>>` with the name of the execution role that you created in the last step.
	`GetUserLambdaFunction:`
	`Type: 'AWS::Lambda::Function'`
	`Properties:`
	`Handler: index.handler`
	`Role: <<ROLENAME>>`
	`Timeout: 30`
	`Runtime: python3.11`
	`Code:`
	`ZipFile: |`
	`import cfnresponse, boto3`
	`def handler(event, context):`
	`try:`
	`print(event)`
	`user = boto3.client('iam').get_user(UserName=event['ResourceP`
		AWS DevOps, Cloud architect

Task	Description	Skills required
	```roperties']['NewIa mUserName'])['User'] cfnrespon se.send(event, context, cfnrespon se.SUCCESS, {'NewIamU serId': user['Use rId'], 'NewIamUs erPath': user['Pat h'], 'NewIamUserArn': user['Arn']}) except Exception as e: cfnrespon se.send(event, context, cfnrespon se.FAILED, {'NewIamU ser': str(e)})```	

Task	Description	Skills required

Add a custom resource.	Add the following code to the `Resources` section of the CloudFormation template:
	`rCustomGetUniqueUserId:`
	`Type: 'Custom::rCustomGetUniqueUserIdWithLambda'`
	`Properties:`
	`ServiceToken: !GetAtt GetUserLambdaFunction.Arn`
	`NewIamUserName: !Ref NewIamUserName`
	This custom resource calls the Lambda function to get the IAM `UserID`.
		AWS DevOps, Cloud architect

Define CloudFormation outputs.	Add the following code to the `Outputs` section of the CloudFormation template:
	`Outputs:`
	`NewIamUserId:`
	`Value: !GetAtt rCustomGetUniqueUserId.NewIamUserId`
	This displays the IAM `UserID` for the new IAM user.
		AWS DevOps, Cloud architect

Save the template.	Save your changes to the CloudFormation template.	AWS DevOps, Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Deploy the CloudFormation template.	To deploy the `get_unique_user_id.yaml` template by using the CloudFormation console, follow the instructions in the [CloudFormation documentation](). Alternatively, you can run the following AWS CLI command to deploy the template: `aws cloudformation create-stack \ --stack-name DemoNewUser \ --template-body file://get_unique_user_id.yaml \ --parameters ParameterKey=NewIamUserName,ParameterValue=demouser \ --capabilities CAPABILITY_NAMED_IAM`	AWS DevOps, Cloud architect

Related resources

- [Create a stack from the CloudFormation console]() (CloudFormation documentation)
- [Lambda-backed custom resources]() (CloudFormation documentation)
- [Unique identifiers]() (IAM documentation)
- [Use temporary credentials with AWS resources]() (IAM documentation)

Validate Account Factory for Terraform (AFT) code locally

Alexandru Pop and Michal Gorniak, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. [Learn more]

This pattern shows how to locally test HashiCorp Terraform code that's managed by AWS Control Tower Account Factory for Terraform (AFT). Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources. AFT sets up a Terraform pipeline that helps you provision and customize multiple AWS accounts in AWS Control Tower.

During code development, it can be helpful to test your Terraform infrastructure as code (IaC) locally, outside of the AFT pipeline. This pattern shows how to do the following:

- Retrieve a local copy of the Terraform code that's stored in the AWS CodeCommit repositories in your AFT management account.
- Simulate the AFT pipeline locally by using the retrieved code.

This procedure can also be used to run Terraform commands that aren't part of the normal AFT pipeline. For example, you can use this method to run commands such as `terraform validate`, `terraform plan`, `terraform destroy`, and `terraform import`.

Prerequisites and limitations

Prerequisites

- An active AWS multi-account environment that uses AWS Control Tower
- A fully deployed of AFT environment
- AWS Command Line Interface (AWS CLI), installed and configured
- AWS CLI credential helper for AWS CodeCommit, installed and configured
- Python 3.x
- Git, installed and configured on your local machine
- `git-remote-commit` utility, installed and configured

- Terraform, installed and configured (the local Terraform package version must match the version that's used in the AFT deployment)

Limitations

- This pattern doesn't cover the deployment steps required for AWS Control Tower, AFT, or any specific Terraform modules.
- The output that's generated locally during this procedure isn't saved in the AFT pipeline runtime logs.

Architecture

Target technology stack

- AFT infrastructure deployed within an AWS Control Tower deployment
- Terraform
- Git
- AWS CLI version 2

Automation and scale

This pattern shows how to locally invoke Terraform code for AFT global account customizations in a single AFT-managed AWS account. After your Terraform code is validated, you can apply it to the remaining accounts in your multi-account environment. For more information, see Re-invoke customizations in the AWS Control Tower documentation.

You can also use a similar process to run AFT account customizations in a local terminal. To locally invoke Terraform code from AFT account customizations, clone the **aft-account-customizations** repository instead of **aft-global-account-customizations** repository from CodeCommit in your AFT management account.

Tools

AWS services

- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices.

- [AWS Command Line Interface (AWS CLI)](#) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.

Other services

- [HashiCorp Terraform](#) is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources.
- [Git](#) is an open-source, distributed version control system.

Code

The following is an example bash script that can be used to locally run Terraform code that's managed by AFT. To use the script, follow the instructions in the Epics section of this pattern.

```
#! /bin/bash
# Version: 1.1 2022-06-24 Unsetting AWS_PROFILE since, when set, it interferes with
 script operation
#          1.0 2022-02-02 Initial Version
#
# Purpose: For use with AFT: This script runs the local copy of TF code as if it were
 running within AFT pipeline.
#        * Facilitates testing of what the AFT pipline will do
#           * Provides the ability to run terraform with custom arguments (like 'plan'
 or 'move') which are currently not supported within the pipeline.
#
# © 2021 Amazon Web Services, Inc. or its affiliates. All Rights Reserved.
# This AWS Content is provided subject to the terms of the AWS Customer Agreement
# available at http://aws.amazon.com/agreement or other written agreement between
# Customer and either Amazon Web Services, Inc. or Amazon Web Services EMEA SARL or
 both.
#
# Note: Arguments to this script are passed directly to 'terraform' without parsing nor
 validation by this script.
#
# Prerequisites:
#    1. local copy of ct GIT repositories
#    2. local backend.tf and aft-providers.tf filled with data for the target account
 on which terraform is to be run
#       Hint: The contents of above files can be obtain from the logs of a previous
 execution of the AFT pipeline for the target account.
#    3. 'terraform' binary is available in local PATH
```

```
#    4. Recommended: .gitignore file containing 'backend.tf', 'aft_providers.tf' so the
 local copy of these files are not pushed back to git

readonly credentials=$(aws sts assume-role \
    --role-arn arn:aws:iam::$(aws sts get-caller-identity --query "Account" --output
 text ):role/AWSAFTAdmin \
    --role-session-name AWSAFT-Session \
    --query Credentials )

unset AWS_PROFILE
export AWS_ACCESS_KEY_ID=$(echo $credentials | jq -r '.AccessKeyId')
export AWS_SECRET_ACCESS_KEY=$(echo $credentials | jq -r '.SecretAccessKey')
export AWS_SESSION_TOKEN=$(echo $credentials | jq -r '.SessionToken')
terraform "$@"
```

Epics

Save the example code as a local file

Task	Description	Skills required
Save the example code as a local file.	1. Copy the example bash script that's in the *Code* section of this pattern and paste it into a code editor. 2. Name the file `ct_terraform.sh`, and then save the file locally inside a dedicated folder, such as `~/scripts` or `~/bin`.	AWS administrator
Make the example code runnable.	Open a terminal window and authenticate into your AWS AFT management account by doing one of the following: • Use an existing [AWS CLI profile](#) that's configured with the permissions required to access the AFT	AWS administrator

Task	Description	Skills required
	management account. To use the profile, you can run the following command: `export AWS_PROFILE=<aft account profile name>` • If your organization uses SSO to access AWS, enter the credentials for your AFT management account on your organization's SSO page. **Note** Your organization might also have a custom tool to provide authentication credentials to your AWS environment.	

Task	Description	Skills required

Verify access to AFT management account in the correct AWS Region.	**Important**
	Make sure that you use the same terminal session with which you authenticated into your AFT management account.
	
	1. Navigate to your AFT deployment's AWS Region by running the following command:
	`export AWS_REGION=<aft_region>`
	2. Make sure that you're in the correct account.
	a. Run the following command:
	`aws code-commit list-repositories`
	b. Verify that the repositories listed in the output match the names of the repositories that are in your AFT management account.
		AWS administrator

Task	Description	Skills required

Create a new, local directory to store the AFT repository code.	In the same terminal session, run the following commands:
	`mkdir my_aft`
	`cd my_aft`
		AWS administrator

Task	Description	Skills required

Clone the remote AFT repository code.	1. In your local terminal, run the following command:
	`git clone codecommit::$AWS_REGION://aft-global-customizations`
	Note
	For simplicity, this procedure and AFT use a main code branch only. To use code branching, you can enter code branching commands here as well. However, any applied changes from the non-main branch will be rolled back when AFT automation applies code from the main branch.
	2. Navigate into the cloned directory:
	`cd aft-global-customizations/terraform`
		AWS administrator

Create the Terraform configuration files required for the AFT pipeline to run locally

Task	Description	Skills required
Open a previously run AFT pipeline and copy the Terraform configuration files to a local folder.	**Note** The `backend.tf` and `aft-providers.tf` configuration files that are created in this epic are needed for the AFT pipeline to run locally. These files are created automatically within the cloud-based AFT pipeline, but must be created manually for the pipeline to run locally. Running the AFT pipeline locally requires one set of files that represent running the pipeline within a single AWS account. 1. Using your AWS Control Tower management account credentials, sign in to the AWS Management Console, and open the [AWS CodePipeline console](https://console.aws.amazon.com/codesuite/codepipeline/home). Make sure that you're in the same AWS Region where you deployed AFT.	AWS administrator

Task	Description	Skills required
	2. In the left navigation pane, choose **Pipelines**. 3. Choose **############-customizations-pipeline**. (The **############** is the AWS account ID that you're using to run Terraform code locally.) 4. Make sure that **Most Recent Execution Marked** shows a **Succeeded** value. If it the value is different , you must re-invoke your customizations in the AFT pipeline. For more information, see Re-invoke customizations in the AWS Control Tower documenta tion. 5. Choose the most recent runtime to bring up its details. 6. In the **Apply-AFT-Global-Customizations** section, find the **Apply-Ter raform** stage. 7. Select the **Details** section of the **Apply-Terraform** stage. 8. Find the runtime log for the **Apply-Terraform** stage.	

Task	Description	Skills required

	9. In the runtime log, look for the section that begins and ends with the following lines:
	`"\n\n aft-provi ders.tf ... "\n \n backend.tf"`
	10. Copy the output between these two labels and save them as a local file named `aft-provi ders.tf` within the local Terraform folder (your terminal session's current working directory).
	Example auto generated providers.tf statement
	`## Autogenerated providers.tf ## ## Updated on: 2022-05-31 16:27:45 ## provider "aws" { region = "us-east-2" assume_role { role_arn = "arn:aws:iam::############:role/AWSAFTExecution" } default_tags { tags = {`	

Task	Description	Skills required

	``
	        managed_by
	               = "AFT"
	      }
	    }
	}
	``
	11. In the runtime log, look for the section that begins and ends with the following lines:
	``
	"\n\n tf
	...
	"\n \n backend.tf"
	``
	12. Copy the output between these two labels and save them as a local file named `tf` within the local Terraform folder (your terminal session's current working directory).
	
	Example auto-generated backend.tf statement
	``
	## Autogenerated
	 backend.tf ##
	## Updated on: 2022-05-3
	1 16:27:45 ##
	terraform {
	  required_version =
	 ">= 0.15.0"
	  backend "s3" {
	    region =
	 "us-east-2"
	    bucket =
	 "aft-backend-#####
	``	

Task	Description	Skills required

	``
	#######-primary-re
	gion"
	    key =
	 "############-aft-
	global-customizati
	ons/terraform.tfst
	ate"
	    dynamodb_table =
	 "aft-backend-#####
	#######"
	    encrypt =
	 "true"
	    kms_key_id =
	 "########-####-####-
	####-############"
	    role_arn =
	 "arn:aws:iam::####
	#########:role/AWS
	AFTExecution"
	  }
	}
	``
	
	Note
	
	The `backend.tf` and `aft-providers.tf` files are tied to a specific AWS account, AFT deployment, and folder. These files are also different , depending on whether they're in the **aft-global-customizations** and **aft-account-customizations** repository within the	

Task	Description	Skills required
	same AFT deployment. Make sure that you generate both files from the same runtime listing.	

Run the AFT pipeline locally by using the example bash script

Task	Description	Skills required
Implement the Terraform configuration changes that you want to validate.	1. Navigate to the cloned **aft-global-customizations** repository by running the following command: `cd aft-global-customizations/terraform` **Note** The files `backend.tf` and `aft-providers.tf` are in this directory. The directory also contains Terraform files from the **aft-global-customizations** repository. 2. Incorporate the Terraform code changes that you	AWS administrator

Task	Description	Skills required
	want to test locally into the configuration files.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Run the <code>ct_terraform.sh</code> script and review the output.</td><td>
1. Navigate to the local folder that contains the sh script.
2. To validate your modified Terraform code, run the <code>ct_terraform.sh</code> script by running the following command:
<pre>~/scripts/ct_terraform.sh apply</pre>
<pre>terraform --help</pre>
Note
You can run any Terraform command during this step. To see a complete list of Terraform commands, run the following command:
3. Review the command output, and then debug the code changes locally before committing and pushing the changes back to the AFT repository.
</td><td>AWS administrator</td></tr>
</table>

Task	Description	Skills required
	⚠ **Important** • Any changes made locally and not pushed back to the remote repository are temporary and might be undone at any time by a running AFT pipeline automation. • AFT automation can run at any time, because it can be invoked by other users and AFT automation triggers. • AFT will always apply code from the main branch of the repository, undoing any uncommitted changes.	

Push your local code changes back to the AFT repository

Task	Description	Skills required
Add references to the `backend.tf` and `aft-providers.tf` files to a `.gitignore` file.	Add the `backend.tf` and `aft-providers.tf` files that you created to a `.gitignore` file by running the following commands: `echo backend.tf >> .gitignore`	AWS administrator

Task	Description	Skills required
	`echo aft-providers.tf >>.gitignore` **Note** Moving the files to the `.gitignore` file ensures that they don't get committed and pushed back to the remote AFT repository.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Commit and push your code changes to the remote AFT repository.</td><td>
1. To add any new Terraform configuration files to the repository, run the following command:
<pre><code>git add <filename></code></pre>
2. To commit your changes and push them to the remote AFT repository in CodeCommitt, run the following commands:
<pre><code>git commit -a
git push</code></pre>
Important
The code changes that you introduce by following this procedure up until this point are applied to one AWS account only.
</td><td>AWS administrator</td></tr>
</table>

Roll out the changes to multiple accounts

Task	Description	Skills required
Roll out the changes to all of your accounts that are managed by AFT.	To roll out the changes to multiple AWS accounts that are managed by AFT, follow	AWS administrator

Task	Description	Skills required
	the instructions in Re-invoke customizations in the AWS Control Tower documenta tion.	

More patterns

- Add HA to Oracle PeopleSoft on Amazon RDS Custom by using a read replica
- Automatically audit AWS security groups that allow access from public IP addresses
- Automate account creation by using the Landing Zone Accelerator on AWS
- Automate adding or updating Windows registry entries using AWS Systems Manager
- Automate AWS resource assessment
- Automate AWS Service Catalog portfolio and product deployment by using AWS CDK
- Automate cross-Region failover and failback by using DR Orchestrator Framework
- Automate deletion of AWS CloudFormation stacks and associated resources
- Automate ingestion and visualization of Amazon MWAA custom metrics on Amazon Managed Grafana by using Terraform
- Automate RabbitMQ configuration in Amazon MQ
- Automate the replication of Amazon RDS instances across AWS accounts
- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK
- Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK
- Automatically detect changes and initiate different CodePipeline pipelines for a monorepo in CodeCommit
- Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config
- Build a data pipeline to ingest, transform, and analyze Google Analytics data using the AWS DataOps Development Kit
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager
- Build and push Docker images to Amazon ECR using GitHub Actions and Terraform
- Centralize IAM access key management in AWS Organizations by using Terraform
- Centralize software package distribution in AWS Organizations by using Terraform
- Chain AWS services together using a serverless approach
- Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode
- Configure model invocation logging in Amazon Bedrock by using AWS CloudFormation
- Configure read-only routing in an Always On availability group in SQL Server on AWS
- Configure VMware vRealize Automation to provision VMs on VMware Cloud on AWS

- Create a portal for micro-frontends by using AWS Amplify, Angular, and Module Federation
- Create a cross-account Amazon EventBridge connection in an organization
- Create dynamic CI pipelines for Java and Python projects automatically
- Deploy an Amazon API Gateway API on an internal website using private endpoints and an Application Load Balancer
- Deploy and debug Amazon EKS clusters
- Deploy and manage AWS Control Tower controls by using AWS CDK and CloudFormation
- Deploy and manage AWS Control Tower controls by using Terraform
- Deploy CloudWatch Synthetics canaries by using Terraform
- Deploy a Lustre file system for high-performance data processing by using Terraform and DRA
- Deploy a RAG use case on AWS by using Terraform and Amazon Bedrock
- Deploy resources in an AWS Wavelength Zone by using Terraform
- Deploy SQL Server failover cluster instances on Amazon EC2 and Amazon FSx by using Terraform
- Deploy the Security Automations for AWS WAF solution by using Terraform
- Detect Amazon RDS and Aurora database instances that have expiring CA certificates
- Document your AWS landing zone design
- Ensure that an IAM profile is associated with an EC2 instance
- Export AWS Backup reports from across an organization in AWS Organizations as a CSV file
- Generate personalized and re-ranked recommendations using Amazon Personalize
- Govern permission sets for multiple accounts by using Account Factory for Terraform
- Identify and alert when Amazon Data Firehose resources are not encrypted with an AWS KMS key
- Implement Account Factory for Terraform (AFT) by using a bootstrap pipeline
- Implement path-based API versioning by using custom domains in Amazon API Gateway
- Install SSM Agent on Amazon EKS worker nodes by using Kubernetes DaemonSet
- Install the SSM Agent and CloudWatch agent on Amazon EKS worker nodes using preBootstrapCommands
- Integrate VMware vRealize Network Insight with VMware Cloud on AWS
- Manage AWS IAM Identity Center permission sets as code by using AWS CodePipeline
- Manage AWS permission sets dynamically by using Terraform
- Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions

- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK
- Manage AWS Organizations policies as code by using AWS CodePipeline and Amazon Bedrock
- Migrate DNS records in bulk to an Amazon Route 53 private hosted zone
- Migrate Oracle PeopleSoft to Amazon RDS Custom
- Migrate RHEL BYOL systems to AWS License-Included instances by using AWS MGN
- Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX
- Set up a minimum viable data space to share data between organizations
- Monitor Amazon ElastiCache clusters for at-rest encryption
- Monitor application activity by using CloudWatch Logs Insights
- Monitor ElastiCache clusters for security groups
- Monitor SAP RHEL Pacemaker clusters by using AWS services
- Optimize multi-account serverless deployments by using the AWS CDK and GitHub Actions workflows
- Provision AWS Service Catalog products based on AWS CloudFormation templates by using GitHub Actions
- Provision least-privilege IAM roles by deploying a role vending machine solution
- Remove Amazon EC2 entries across AWS accounts from AWS Managed Microsoft AD by using AWS Lambda automation
- Remove Amazon EC2 entries in the same AWS account from AWS Managed Microsoft AD by using AWS Lambda automation
- Rotate database credentials without restarting containers
- Secure file transfers by using Transfer Family, Amazon Cognito, and GuardDuty
- Send a notification when an IAM user is created
- Send logs from VMware Cloud on AWS to Splunk by using VMware Aria Operations for Logs
- Set up a serverless cell router for a cell-based architecture
- Set up a CI/CD pipeline for hybrid workloads on Amazon ECS Anywhere by using AWS CDK and GitLab
- Set up an HA/DR architecture for Oracle E-Business Suite on Amazon RDS Custom with an active standby database
- Set up DNS resolution for hybrid networks in a multi-account AWS environment
- Set up Multi-AZ infrastructure for a SQL Server Always On FCI by using Amazon FSx

- Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible
- Simplify application authentication with mutual TLS in Amazon ECS by using Application Load Balancer
- Simplify private certificate management by using AWS Private CA and AWS RAM
- Streamline machine learning workflows from local development to scalable experiments by using SageMaker AI and Hydra
- Tag Transit Gateway attachments automatically using AWS Organizations
- Transition roles for an Oracle PeopleSoft application on Amazon RDS Custom for Oracle
- Use Amazon Q Developer as a coding assistant to increase your productivity

Web & mobile apps

Topics

- Authenticate existing React application users by using Amazon Cognito and AWS Amplify UI
- Continuously deploy a modern AWS Amplify web application from an AWS CodeCommit repository
- Create a React app by using AWS Amplify and add authentication with Amazon Cognito
- Create a portal for micro-frontends by using AWS Amplify, Angular, and Module Federation
- Deploy a React-based single-page application to Amazon S3 and CloudFront
- Deploy an Amazon API Gateway API on an internal website using private endpoints and an Application Load Balancer
- Embed an Amazon QuickSight dashboard in a local Angular application
- Explore full-stack cloud-native web application development with Green Boost
- Run unit tests for a Node.js application from GitHub by using AWS CodeBuild
- Structure a Python project in hexagonal architecture using AWS Lambda
- More patterns

Authenticate existing React application users by using Amazon Cognito and AWS Amplify UI

Daniel Kozhemyako, Amazon Web Services

Summary

This pattern shows how to add authentication capabilities to an existing frontend React application by using an AWS Amplify UI library and an Amazon Cognito user pool.

The pattern uses Amazon Cognito to provide authentication, authorization, and user management for the application. It also uses a component from Amplify UI, an open-source library that extends the capabilities of AWS Amplify to user interface (UI) development. The Authenticator UI component manages login sessions and runs the cloud-connected workflow that authenticates users through Amazon Cognito.

After you implement this pattern, users can sign in by using any of the following credentials:

- User name and password
- Social identity providers, such as Apple, Facebook, Google, and Amazon
- Enterprise identity providers that are either SAML 2.0 compatible or OpenID Connect (OIDC) compatible

> **Note**
>
> To create a custom authentication UI component, you can run the Authenticator UI component in headless mode.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A React 18.2.0 or later web application
- Node.js and npm 6.14.4 or later, installed

Limitations

- This pattern applies to React web applications only.
- This pattern uses a prebuilt Amplify UI component. The solution doesn't cover the steps required to implement a custom UI component.

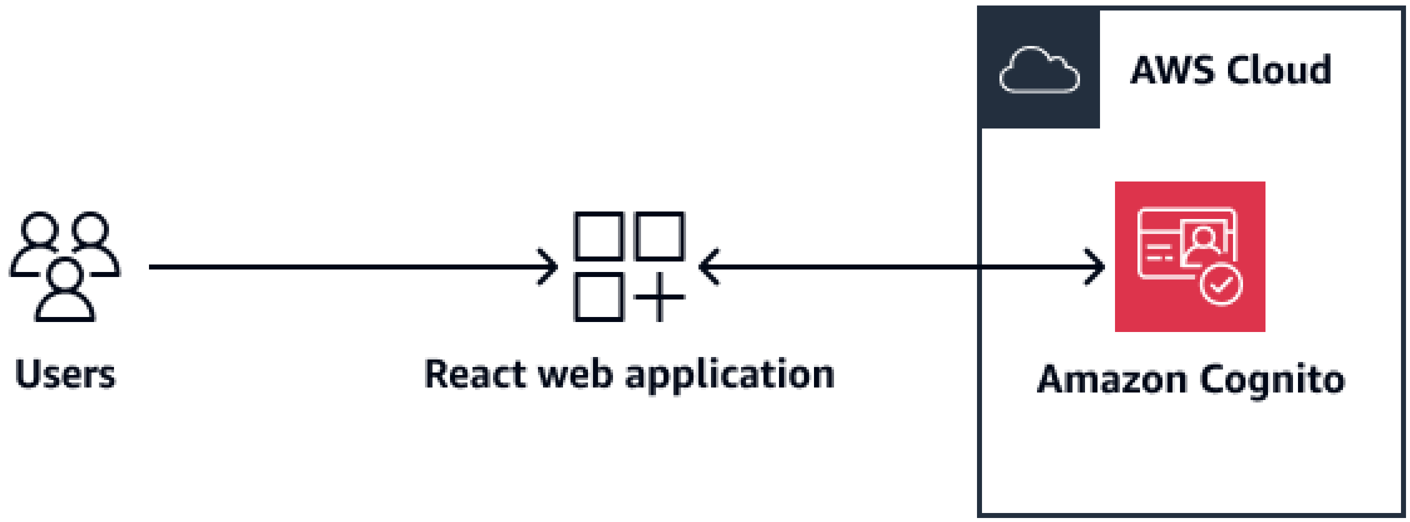
Product versions

- Amplify UI 6.1.3 or later (Gen 1)
- Amplify 6.0.16 or later (Gen 1)

Architecture

Target architecture

The following diagram shows an architecture that uses Amazon Cognito to authenticate users for a React web application.



Tools

AWS services

- [Amazon Cognito](https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html) provides authentication, authorization, and user management for web and mobile apps.

Other tools

- Amplify UI is an open-source UI library that provides customizable components that you can connect to the cloud.
- Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.

Best practices

If you're building a new application, we recommend that you use Amplify Gen 2.

Epics

Create an Amazon Cognito user pool

Task	Description	Skills required
Create a user pool.	Create an Amazon Cognito user pool. Configure the user pool's sign-in options and security requirements to fit your use case.	App developer
Add an app client.	Configure a user pool app client. This client is required for your application to interact with the Amazon Cognito user pool.	App developer

Integrate your Amazon Cognito user pool with the Authenticator UI component

Task	Description	Skills required
Install dependencies.	To install the `aws-amplify` and `@aws-amplify/ui-react` packages, run	App developer

Task	Description	Skills required
	the following command from your application's root directory: `npm i @aws-amplify/ui-react aws-amplify`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Configure the user pool.</td><td>Based on the following example, create an <code>aws-exports.js</code> file and save in the <code>src</code> folder. The file should include the following information:

AWS Region that your Amazon Cognito user pool is in
Amazon Cognito user pool ID
App client ID

<pre><code>// replace the user pool region, id, and app client id details
const awsmobile = {
    "aws_project_region": "put_your_region_here",
    "aws_cognito_region": "put_your_region_here",
    "aws_user_pools_id": "put_your_user_pool_id_here",
    "aws_user_pools_web_client_id": "put_your_user_pool_app_id_here"
}

export default awsmobile;</code></pre></td><td>App developer</td></tr>
</table>

Task	Description	Skills required

Import and configure the Amplify service.	1. In the application's entry point file (for example, `App.js`), import and load the `aws-exports.js` file by entering the following lines of code:
	`import { Amplify } from 'aws-amplify'; import awsExports from './aws-exports';`
	2. Based on the following example, configure the Amplify client by using the `aws-exports.js` file:
	`// Configure Amplify in index file or root file Amplify.configure({ ...awsExports });`
	For more information, see [Configure Amplify categories] in the Amplify documentation.
		App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add the Authenticator UI component.</td><td>To display the <code>Authenticator</code> UI component, add the following lines of code to the application's entry point file (<code>App.js</code>):<pre><code>import { Authenticator } from '@aws-amplify/ui-react';
import '@aws-amplify/ui-react/styles.css';</code></pre>Note
The example code snippet imports the <code>Authenticator</code> UI component and the Amplify UI styles.css file, which is required when using the component's prebuilt themes.

The <code>Authenticator</code> UI component provides two return values:
User details
A function that can be invoked to sign the user out
See the following example component:</td><td>App developer</td></tr>
</table>

Task	Description	Skills required

	<pre>function App() {
	    return (
	        <Authenticator>
	            {({ signOut,
	 user }) => (
	
	  Welcome {user.use
	rname}
	
	  <button onClick={
	signOut}>Sign out</
	button>
	
	            )}
	        </Authent
	icator>
	    );
	}</pre> **Note** For an example `App.js` file, see the Additional information section of this pattern.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>(Optional) Retrieve session information.</td><td>After a user is authenticated, you can retrieve data from the Amplify client about their session. For example, you can retrieve the JSON web token (JWT) from a user’s session so that you can authenticate the requests from their session to a backend API.

See the following example of a request header that includes a JWT:

<pre>import { fetchAuth
Session } from 'aws-
amplify/auth';
(await fetchAuth
Session()).tokens?
.idToken?.toString();</pre></td><td>App developer</td></tr>
</table>

Troubleshooting

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td>New users can’t sign up for the application.</td><td>As follows, make sure that your Amazon Cognito user pool is configured to allow users to sign themselves up to be in the user pool:
Sign in to the AWS Management Console, and then open the Amazon Cognito console.
In the left navigation pane, choose User pools.
Choose your user pool from the list.</td></tr>
</table>

Issue	Solution
	• Under **General Settings**, choose **Policies**. • Choose **Allow users to sign themselves up**.
Auth component stopped working after upgrading from v5 to v6.	The `Auth` category has moved to a functional approach and named parameters in Amplify v6. You must now import the functional APIs directly from the `aws-amplify/auth` path. For more information, see Migrate from v5 to v6 in the Amplify documentation.

Related resources

- Getting started with Amazon Cognito (AWS website)
- Create a new React app (React documentation)
- What is Amazon Cognito? (Amazon Cognito documentation)
- Amplify UI library (Amplify documentation)

Additional information

The `App.js` file should contain the following code:

```
import './App.css';
import { Amplify } from 'aws-amplify';
import awsExports from './aws-exports';
import { fetchAuthSession } from 'aws-amplify/auth';
import { Authenticator } from '@aws-amplify/ui-react';
import '@aws-amplify/ui-react/styles.css';
Amplify.configure({ ...awsExports });
let token = (await fetchAuthSession()).tokens?.idToken?.toString();
function App() {
  return (
      <Authenticator>
        {({ signOut, user }) => (
            <div>
              <p>Welcome {user.username}</p>
                <p>Your token is: {token}</p>
```

```
              <button onClick={signOut}>Sign out</button>
            </div>
        )}
      </Authenticator>
  );
}

export default App;
```

Continuously deploy a modern AWS Amplify web application from an AWS CodeCommit repository

Deekshitulu Pentakota and Sai Katakam, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more.

Modern web applications are constructed as single-page applications (SPAs) that package all application components into static files. By using AWS Amplify Hosting, you can build a continuous integration and continuous deployment (CI/CD) pipeline that builds, deploys, and hosts a modern web application that is managed in a Git-based repository. When you connect Amplify Hosting to the code repository, each commit initiates a single workflow to deploy the application frontend and backend. The benefit of this approach is that the web application is updated only after the deployment is successfully completed, which prevents inconsistencies between the frontend and backend.

In this pattern, you use an AWS CodeCommit repository to manage your modern web application. The sample web application in these instructions uses the React SPA framework. However, Amplify Hosting supports many other SPA frameworks, such as Angular, Vue, Next.js, and it also support single-site generators, such as Gatsby, Hugo, and Jekyll.

This pattern is intended for AWS builders who have experience with the following services and concepts:

- AWS CodeCommit
- AWS Amplify Hosting
- React
- JavaScript
- Node.js
- npm
- Git

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Permissions to create resources in Amplify and CodeCommit. For more information, see Identity and Access Management for Amplify and Identity and Access Management for AWS CodeCommit.
- AWS Command Line Interface (AWS CLI), installed and configured.
- A text editor or code editor.
- CodeCommit, set up for HTTPS users using Git credentials.
- An IAM service role for Amplify.
- npm and Node.js, installed (npm documentation).

Limitations

- This pattern doesn't discuss development and integration of a backend for the Amplify application, such as an API, authentication, or database. For more information about backends, see Create a backend in the Amplify documentation.

Product versions

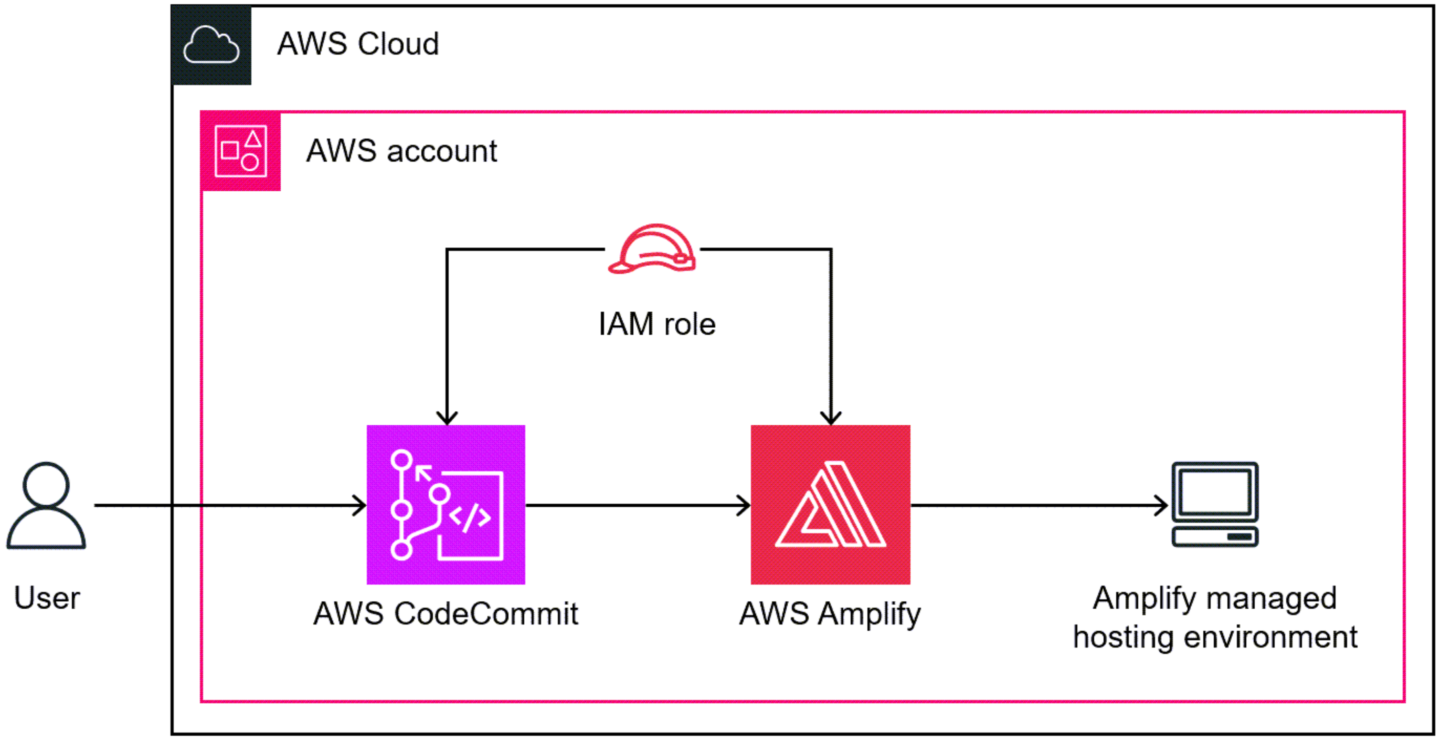
- AWS CLI version 2.0
- Node.js version 16.x or later

Architecture

Target technology stack

- AWS CodeCommit repository containing a React SPA
- AWS Amplify Hosting workflow

Target architecture



Tools

AWS services

- AWS Amplify Hosting provides a Git-based workflow for hosting full-stack serverless web applications with continuous deployment.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Other tools

- Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.

Epics

Create a CodeCommit repository

Task	Description	Skills required
Create a repository.	For instructions, see Create an AWS CodeCommit repository in the CodeCommit documentation.	AWS DevOps
Clone the repository.	For instructions, see Connect to the CodeCommit repository by cloning the repository in the CodeCommit documentation. If you are prompted, provide the Git credentials.	App developer

Create a React application

Task	Description	Skills required
Create a new React application.	1. Enter the following command to navigate into cloned repo. Replace `<repo name>` with the name of your CodeCommit repo. `$ cd <repo name>` 2. Enter the following command to create a new React application in the cloned repository.	App developer

Task	Description	Skills required

	`$ npx create-react-app .`
	3. Code the application, and then enter the following command to start it.
	`$ npm start`
	
	For more information about creating a custom React application, see the Create React App instructions in the *Create React App* documentation. You can also deploy a sample React application to your Amplify account by following the instructions in Deploy a frontend in the Amplify documentation.	

Task	Description	Skills required

Create a branch and push the code.	1. Enter the following command to create a new branch locally, where `<branch>` is the name you want to assign to the new branch.
	`$ git checkout -b <branch>`
	2. Enter the following command to push the branch to the CodeCommit repository, where `<branch>` is the name you assigned in the previous step. For more information, see [Working with commits](#).
	`$ git push --set-upstream origin <branch>`
		App developer

Deploy the application in AWS Amplify Hosting

Task	Description	Skills required
Connect Amplify to the repository.	For instructions, see [Connect a repository](#) in the Amplify Hosting documentation. Select AWS CodeCommit and the repository and branch you created previously.	App developer
Define the frontend build settings.	For instructions, see [Confirm build settings for the](#)	App developer

Task	Description	Skills required

	frontend in the Amplify Hosting documentation. Accept the defaults or enter the following.
	
	`Build settings:`
	`version: 0.1`
	`frontend:`
	`  phases:`
	`    preBuild:`
	`      commands:`
	`        - npm ci`
	`    build:`
	`      commands:`
	`        - npm run build`
	`  artifacts:`
	`    baseDirectory: build`
	`    files:`
	`      - '**/*'`
	`  cache:`
	`    paths:`
	`      - node_modules/**/*`	

Review and deploy.	For instructions, see Save and deploy in the Amplify Hosting documentation. Wait until the deployment process is complete.	App developer

Validate continuous deployment

Task	Description	Skills required
Verify initial deployment.	When the deployment process is complete, under	App developer

Task	Description	Skills required
	Domain, choose the link. Verify that the application is operating as expected.	
Push a change to the code repository.	Edit the code on your local workstation and push the changes to CodeCommit repository. Amplify Hosting detects the change in the repository and automatically starts the build and deployment process. Confirm that the application updates are visible on the domain.	App developer

Related resources

AWS CodeCommit documentation

- Setting up for AWS CodeCommit
  - Setup for HTTPS users using Git credentials
  - Setup steps for HTTPS connections to AWS CodeCommit repositories on Linux, macOS, or Unix with the AWS CLI credential helper
- Getting started with AWS CodeCommit

AWS Amplify Hosting documentation

- Getting started with existing code
- Setting up custom domains

React resources

- Create React App website
- Create React App documentation

- Create React App repository (GitHub)

Create a React app by using AWS Amplify and add authentication with Amazon Cognito

Rishi Singla, Amazon Web Services

Summary

This pattern demonstrates how to use AWS Amplify to create a React-based app and how to add authentication to the frontend by using Amazon Cognito. AWS Amplify consists of a set of tools (open source framework, visual development environment, console) and services (web app and static website hosting) to accelerate the development of mobile and web apps on AWS.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Node.js and npm installed on your machine

Product versions

- Node.js version 10.x or later (to verify your version, run `node -v` in a terminal window)
- npm version 6.x or later (to verify your version, run `npm -v` in a terminal window)

Architecture

Target technology stack

- AWS Amplify
- Amazon Cognito

Tools

- Amplify Command Line Interface (CLI)
- Amplify Libraries (open source client libraries)
- Amplify Studio (visual interface)

Epics

Install AWS Amplify CLI

Task	Description	Skills required

Install the Amplify CLI.	The Amplify CLI is a unified toolchain for creating AWS Cloud services for your React app. To install the Amplify CLI, run:
	`npm install -g @aws-amplify/cli`
	npm will notify you if a new major version is available . If so, use the following command to upgrade your version of npm:
	`npm install -g npm@9.8.0`
	where 9.8.0 refers to the version you want to install.
		App developer

Create a React app

Task	Description	Skills required
Create a React app.	To create a new React app, use the command: `npx create-react-app amplify-react-application`	App developer

Task	Description	Skills required
	where `ampify-react-application` is the name of the app. When the app has been created successfully, you will see the message: `Success! Created amplify-react-application` A directory with various subfolders will be created for the React app.	
Launch the app on your local machine.	Go to the directory `amplify-react-application` that was created in the previous step and run the command: `amplify-react-application% npm start` This launches the React app on your local machine.	App developer

Configure the Amplify CLI

Task	Description	Skills required
Configure Amplify to connect to your AWS account.	Configure Amplify by running the command:	General AWS, App developer

Task	Description	Skills required
	```amplify-react-appl ication % amplify configure``` The Amplify CLI asks you to follow these steps to set up access to your AWS account: 1. Sign in to your AWS administrator account. 2. Specify the AWS Region you want to use. 3. Create an AWS Identity and Access Management (IAM) user with programmatic access, and attach the `AdministratorAccess-Amplify` permissions policy to the user. 4. Create and then copy the access key ID and secret access key. 5. Enter these details in the terminal. 6. Create a profile name or use the default profile. **Warning** This scenario requires IAM users with programmatic access and long-term credentials, which	

Task	Description	Skills required

	present a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed. Access keys can be updated if necessary. For more information, see [Updating access keys](#) in the *IAM User Guide*.
	
	These steps appear in the terminal as follows.
	
	`Follow these steps to set up access to your AWS account:`
	`Sign in to your AWS administrator account:`
	`https://console.aws.amazon.com/`
	`Press Enter to continue`
	`Specify the AWS Region`
	`? region:  us-east-1`
	`Follow the instructions at`
	`https://docs.amplify.aws/cli/start/`	

Task	Description	Skills required
	``` install/#configure- the-amplify-cli to complete the user creation in the AWS console https://console.aws .amazon.com/iamv2/ home#/users/create Press Enter to continue Enter the access key of the newly created user: ? accessKeyId: ******************** ? secretAccessKey: ****************** ****************** **** This would update/cr eate the AWS Profile in your local machine ? Profile Name: new Successfully set up the new user. ``` For more information about these steps, see the documentation in the Amplify Dev Center.	

Initialize Amplify

Task	Description	Skills required
Initialize Amplify.	1. To initialize Amplify in the new directory, run: `amplify init`	App developer, General AWS

Task	Description	Skills required
	Amplify prompts you for the project name and configuration parameters 2. Specify all parameters, and then press **Y** to initializ e the project with the specified configuration. `Project information	Name: amplifyre actproject	Environment: dev	Default editor: Visual Studio Code	App type: javascrip t	Javascript framework: react	Source Directory Path: src	Distribution Directory Path: build	Build Command: npm run-script build	Start Command: npm run-script start` 3. Select the profile you created in the previous step. The resources will be deployed into the `dev`	

Task	Description	Skills required

	environment in the Amplify project you created.
	4. To confirm that the resources have been created, you can open the [AWS Amplify console](#) and view the AWS CloudFormation template that was used to create the resources and the details.
	`Deploying root stack amplifyreactproject [ ==================================---- ] 2/4 amplify-amplifyreactproject-d… AWS::CloudFormation::Stack CREATE_IN_PROGRESS`
	
	`UnauthRole AWS::IAM::Role CREATE_COMPLETE`
	
	`DeploymentBucket AWS::S3::Bucket CREATE_IN_PROGRESS`
	
	`AuthRole AWS::IAM:`	

Task	Description	Skills required
	`:Role CREATE_COMPLETE`	

Add authentication to the frontend

Task	Description	Skills required
Adding authentication.	You can use the `amplify add <category>` command to add features such as a user login or a backend API. In this step you will use the command to add authentication. Amplify provides a backend authentication service with Amazon Cognito, frontend libraries, and a drop-in Authenticator UI component . Features include user sign-up, user sign-in, multi-fac tor authentication, user sign-out, and passwordless sign-in. You can also authentic ate users by integrating with federated identity providers such as Amazon, Google, and Facebook. The Amplify authentication category integrates seamlessly with other Amplify categories such as API, analytics, and storage, so you can define authoriza	App developer, General AWS

Task	Description	Skills required

	tion rules for authenticated and unauthenticated users.
	
	1. To configure authentic ation for your React app, run the command:
	
	`amplify-react-appl ication1 % amplify add auth`
	
	This displays the following information and prompts. You can choose the appropriate configura tion depending on your business and security requirements.
	
	`Using service: Cognito, provided by: awscloudformation The current configure d provider is Amazon Cognito. Do you want to use the default authentic ation and security configuration? (Use arrow keys)`
	
	`# Default configura tion`
	
	`Default configura tion with Social Provider (Federati on)`	

Task	Description	Skills required

	``
	  Manual configura
	tion
	
	  I want to learn
	 more.
	``
	2. For a simple example, choose the default configuration and then select the sign-in mechanism for users (in this case, email):
	``
	How do you want users
	 to be able to sign
	 in?
	 Username
	
	# Email
	
	  Phone Number
	
	  Email or Phone
	 Number
	
	  I want to learn
	 more.
	``
	3. Bypass the advanced settings to complete adding authentication resources:
	``
	Do you want to
	 configure advanced
	 settings? (Use arrow
	 keys)
	# No, I am done.
	``	

Task	Description	Skills required

	``
	 Yes, I want to make
	 some additional
	 changes.
	``
	4. Build your local backend resources and provision them in the cloud:
	``
	amplify-react-appl
	ication1 % amplify
	 push
	``
	This command makes the appropriate changes to the Congito user pools in your account.
	5. Press **Y** to configure the `auth` resource by using CloudFormation.
	This configures the following resources:
	``
	UserPool
	           AWS::Cogn
	ito::UserPool
	  CREATE_COMPLETE
	UserPoolClientWeb
	            AWS::Cogn
	ito::UserPoolClient
	   CREATE_COMPLETE
	
	UserPoolClientWeb
	            AWS::Cogn
	ito::UserPoolClient
	   CREATE_COMPLETE
	
	UserPoolClientRole
	            AWS::IAM:
	``	

Task	Description	Skills required

	``
	:Role
	 CREATE_COMPLETE
	
	UserPoolClientLambda
	           AWS::Lamb
	da::Function
	  CREATE_COMPLETE
	UserPoolClientLam
	bdaPolicy
	 AWS::IAM::Policy
	            CREATE_CO
	MPLETE
	
	UserPoolClientLog
	Policy
	 AWS::IAM::Policy
	            CREATE_IN
	_PROGRESS
	``
	
	You can also use the [AWS Cognito console](https://console.aws.amazon.com/cognito/) to view these resources (look for Cognito user pools and identity pools).
	
	This step updates the `aws-exports.js` file in the `src` folder for your React app with the Cognito user pool and identity pool configurations.	

Change the App.js file

Task	Description	Skills required

Change the App.js file.	In the `src` folder, open and revise the `App.js` file. The modified file should look like this:
	<pre>{ App.Js File after
	 modifications:
	import React from
	 'react';
	import logo from './
	logo.svg';
	import './App.css';
	import { Amplify } from
	 'aws-amplify';
	import { withAuthe
	nticator, Button,
	 Heading } from '@aws-
	amplify/ui-react';
	import awsconfig from
	 './aws-exports';
	Amplify.configure(a
	wsconfig);
	function
	 App({ signOut }) {
	  return (
	      <div>
	      <h1>Thankyou for
	 doing verification</
	h1>
	      <h2>My Content</
	h2>
	       <button onClick={
	signOut}>Sign out</
	button>
	    </div>
	  );
	}</pre>
		App developer

Task	Description	Skills required
	```export default withAuthenticator(App);```	
Import React packages.	The `App.js` file imports two React packages. Install these packages by using the command: ```amplify-react-application1 % npm install --save aws-amplify @aws-amplify/ui-react```	App developer

Launch the React app and check authentication

Task	Description	Skills required
Launch the app.	Launch the React app on your local machine: ```amplify-react-application1 % npm start```	App developer, General AWS
Check authentication.	Check whether the app prompts for authentication parameters. (In our example, we've configured email as the sign-in method.) The frontend UI should prompt you for sign-in credentials and provide an option to create an account.	App developer, General AWS

Task	Description	Skills required
	You can also configure the Amplify build process to add the backend as part of a continuous deploymen t workflow. However, this pattern doesn’t cover that option.	

Related resources

- Getting started (npm documentation)
- Create a standalone AWS account (AWS Account Management documentation)
- AWS Amplify documentation
- Amazon Cognito documentation

Create a portal for micro-frontends by using AWS Amplify, Angular, and Module Federation

Milena Godau and Pedro Garcia, Amazon Web Services

Summary

A micro-frontend architecture enables multiple teams to work on different parts of a frontend application independently. Each team can develop, build, and deploy a fragment of the frontend without interfering with other parts of the application. From the end user's perspective, it appears to be a single, cohesive application. However, they are interacting with several independent applications that are published by different teams.

This document describes how to create a micro-frontend architecture by using AWS Amplify, the Angular frontend framework, and Module Federation. In this pattern, the micro-frontends are combined on the client side by a shell (or *parent*) application. The shell application acts as a container that retrieves, displays, and integrates the micro-frontends. The shell application handles the global routing, which loads different micro-frontends. The @angular-architects/module-federation plugin integrates Module Federation with Angular. You deploy the shell application and micro-frontends by using AWS Amplify. End users access the application through a web-based portal.

The portal is split vertically. This means that the micro-frontends are entire views or groups of views, instead of parts of the same view. Therefore the shell application loads only one micro-frontend at a time.

The micro-frontends are implemented as remote modules. The shell application lazily loads these remote modules, which defers the micro-frontend initialization until it is required. This approach optimizes application performance by loading only the necessary modules. This reduces the initial load time and improves the overall user experience. Additionally, you share common dependencies across modules through the webpack configuration file (**webpack.config.js**). This practice promotes code reuse, reduces duplication, and streamlines the bundling process.

Prerequisites and limitations

Prerequisites

- An active AWS account

- Node.js and npm, installed
- Amplify CLI, installed
- Angular CLI, installed
- Permissions to use AWS Amplify
- Familiarity with Angular

Product versions

- Angular CLI version 13.1.2 or later
- @angular-architects/module-federation version 14.0.1 or later
- webpack version 5.4.0 or later
- AWS Amplify Gen 1

Limitations

A micro-frontend architecture is a powerful approach for building scalable and resilient web applications. However, it's crucial to understand the following potential challenges before adopting this approach:

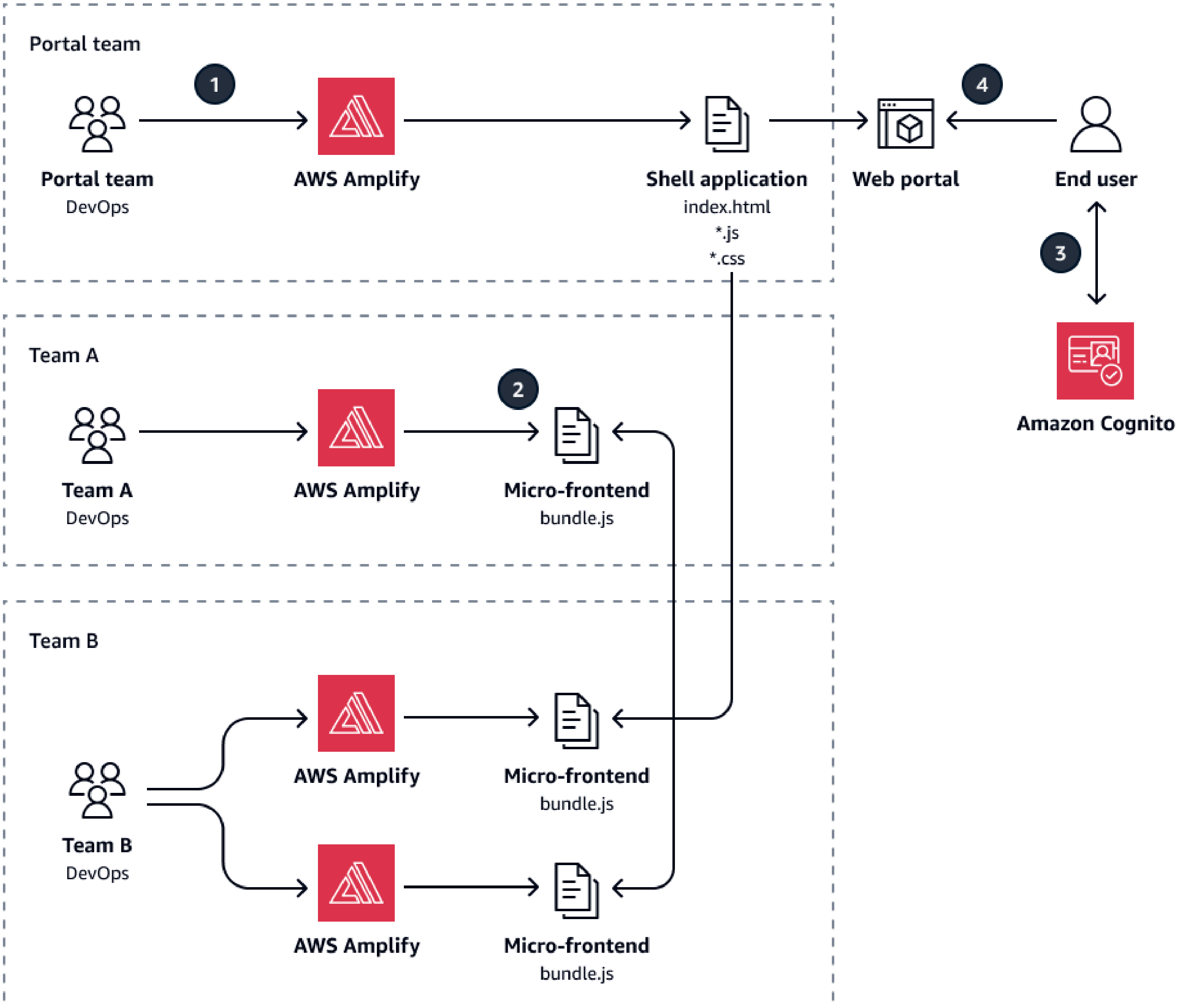
- **Integration –** One of the key challenges is the potential increase in complexity compared to monolithic frontends. Orchestrating multiple micro-frontends, handling communication between them, and managing shared dependencies can be more intricate. Additionally, there may be a performance overhead associated with communication between the micro-frontends. This communication can increase latency and reduce performance. This needs to be addressed through efficient messaging mechanisms and data-sharing strategies.
- **Code duplication –** Because each micro-frontend is developed independently, there is a risk of duplicating code for common functionality or shared libraries. This can increase the overall application size and introduce maintenance challenges.
- **Coordination and management –** Coordinating the development and deployment processes across multiple micro-frontends can be challenging. Ensuring consistent versioning, managing dependencies, and maintaining compatibility between components becomes more critical in a distributed architecture. Establishing clear governance, guidelines, and automated testing and deployment pipelines is essential for seamless collaboration and delivery.
- **Testing –** Testing micro-frontend architectures can be more complex than testing monolithic frontends. It requires additional effort and specialized testing strategies to perform cross-

component integration testing and end-to-end testing, and to validate consistent user experiences across multiple micro-frontends.

Before committing to the micro-frontend approach, we recommend that you review Understanding and implementing micro-frontends on AWS.

Architecture

In a micro-frontend architecture, each team develops and deploys features independently. The following image shows how multiple DevOps teams work together. The portal team develops the shell application. The shell application acts as a container. It retrieves, displays, and integrates the micro-frontend applications that are published by other DevOps teams. You use AWS Amplify to publish the shell application and micro-frontend applications.



The architecture diagram shows the following workflow:

1. The portal team develops and maintains the shell application. The shell application orchestrates the integration and rendering of the micro-frontends in order to compose the overall portal.
2. Teams A and B develop and maintain one or more micro-frontends or features that are integrated into the portal. Each team can work independently on their respective micro-frontends.
3. The end user authenticates by using Amazon Cognito.

4. The end user accesses the portal, and the shell application is loaded. As the user navigates, the shell application deals with the routing and retrieves the requested micro-frontend, loading its bundle.

Tools

AWS services

- AWS Amplify is a set of purpose-built tools and features that helps frontend web and mobile developers quickly build full-stack applications on AWS. In this pattern, you use the Amplify CLI to deploy the Amplify micro-frontend applications.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.

Other tools

- @angular-architects/module-federation is a plugin that integrates Angular with Module Federation.
- Angular is an open source web application framework for building modern, scalable, and testable single-page applications. It follows a modular and component-based architecture that promotes code reuse and maintenance.
- Node.js is an event-driven JavaScript runtime environment designed that is for building scalable network applications.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.
- Webpack Module Federation helps you load code that is independently compiled and deployed, such as micro-frontends or plugins, into an application.

Code repository

The code for this pattern is available in the Micro-frontend portal using Angular and Module Federation GitHub repository. This repository contains the following two folders:

- `shell-app` contains the code for the shell application.
- `feature1-app` contains a sample micro-frontend. The shell application fetches this micro-frontend and displays it as a page within the portal application.

Best practices

Micro-frontend architectures offer numerous advantages, but they also introduce complexity. The following are some best practices for smooth development, high-quality code, and a great user experience:

- **Planning and communication –** To streamline collaboration, invest in upfront planning, design, and clear communication channels.
- **Design consistency –** Enforce a consistent visual style across micro-frontends by using design systems, style guides, and component libraries. This provides a cohesive user experience and accelerates development.
- **Dependency management –** Because micro-frontends evolve independently, adopt standardized contracts and versioning strategies to manage dependencies effectively and prevent compatibility issues.
- **Micro-frontend architecture –** To enable independent development and deployment, each micro-frontend should have a clear and well-defined responsibility for an encapsulated functionality.
- **Integration and communication –** To facilitate smooth integration and minimize conflicts, define clear contracts and communication protocols between micro-frontends, including APIs, events, and shared data models.
- **Testing and quality assurance –** Implement test automation and continuous integration pipelines for micro-frontends. This improves the overall quality, reduces manual testing effort, and validates functionality between micro-frontend interactions.
- **Performance optimization –** Continuously monitor performance metrics and track dependencies between micro-frontends. This helps you identify bottlenecks and maintain optimal application performance. Use performance monitoring and dependency analysis tools for this purpose.
- **Developer experience –** Focus on the developer experience by providing clear documentation, tooling, and examples. This helps you streamline development and onboard new team members.

Epics

Create the shell application

Task	Description	Skills required

Create the shell application.	1. In the Angular CLI, enter the following command:
	`ng new shell --routing`
	2. Enter the following command to navigate to the project folder:
	`cd shell`
	Note
	The folder and project structure for the shell and micro-frontend applications can be totally independent. They can be handled as independent Angular applications.
		App developer

Install the plugin.	In the Angular CLI, enter the following command to install the [@angular-architects/module-federation](https://www.npmjs.com/package/@angular-architects/module-federation) plugin:	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>
<pre><code>ng add @angular-
architects/module-
federation --project
 shell --port 4200</code></pre>
</td><td></td></tr>
<tr><td>Add the micro-frontend URL as an environment variable.</td><td>
1. Open the environment.ts file.
2. Add <code>mfe1URL: 'http://localhost:5000'</code> to the <code>environment</code> object:
<pre><code>export const environme
nt = {
  production: false,
  mfe1URL: 'http://l
ocalhost:5000',
};</code></pre>
3. Save and close the environment.ts file.
</td><td>App developer</td></tr>
</table>

Task	Description	Skills required

Define routing.	1. Open the **app-routi ng.module.ts** file.
	2. In the Angular CLI, enter the following command to import the `loadRemot eModule` module from the @angular-architects/ module-federation plugin:
	`import { loadRemot eModule } from '@angular-architec ts/module-federati on';`
	3. Set the default route as the following:
	`{ path: '', pathMatch: 'full', redirectTo: 'mfe1' },`
	4. Set the route for the micro-frontend:
	`{ path: 'mfe1', loadChildren: () => loadRemot eModule({ type: 'module', remoteEntry: `${environment.mfe`
		App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

```
1URL}/remoteEntry.
js`,
        exposedMo
dule: './Module'
      })
      .then(m =>
 m.Mfe1Module)
  },
```

5. Save and close the **app-routing.module.ts** file.</td><td></td></tr>
<tr><td>Declare the `mfe1` module.</td><td>

1. In the `src` folder, create a new file named **decl.d.ts**.
2. Open the **decl.d.ts** file.
3. Add the following to the file:

```
declare module 'mfe1/
Module';
```

4. Save and close the **decl.d.ts** file.</td><td>App developer</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Prepare preloading for the micro-frontend.</td><td>Preloading the micro-frontend helps the webpack properly negotiate the shared libraries and packages.

1. Open the main.ts file.
2. Replace the content with the following:
<pre><code>import { loadRemoteEntry } from
 '@angular-architects/module-federation';

Promise.all([
    loadRemoteEntry(`${environment.mfe1URL}/remoteEntry.js`, 'mfe1'),
])
.catch(err =>
 console.error('Error loading remote
 entries', err))
.then(() => import('./bootstrap'))
.catch(err =>
 console.error(err)
);</code></pre>3. Save and close the main.ts file.</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
Adjust the HTML content.	1. Open the **app.component.html** file. 2. Replace the content with the following: `<h1>Shell application is running!</h1> <router-outlet></router-outlet>` 3. Save and close the **app.component.html** file.	App developer

Create the micro-frontend application

Task	Description	Skills required

Create the micro-frontend.	1. In the Angular CLI, enter the following command:
	`ng new mfe1 --routing`
	2. Enter the following command to navigate to the project folder:
	`cd mfe1`
		App developer

Install the plugin.	Enter the following command to install the @angular-architects/module-federation plugin: `ng add @angular-architects/module-`	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>federation --project
 mfe1 --port 5000</code></pre></td><td></td></tr>
<tr><td>Create a module and component.</td><td>Enter the following commands to create a module and component and export them as the remote entry module:<pre><code>ng g module mfe1 --
routing
ng g c mfe1</code></pre></td><td>App developer</td></tr>
<tr><td>Set the default routing path.</td><td>Open the mfe-routi ng.module.ts file.
Set the default route as the following:<pre><code>{
    path: '',
    component:
 Mfe1Component
  },</code></pre>
Save and close the mfe-routing.module.ts file.</td><td>App developer</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add the <code>mfe1</code> route.</td><td>
1. Open the app-routing.module.ts file.
2. Set the default route as the following:
<pre><code>{
    path: '',
    pathMatch: 'full',
    redirectTo: 'mfe1'
  },</code></pre>
3. Add the following <code>mfe1</code> route:
<pre><code>{
    path: 'mfe1',
    loadChildren: () =>
      import('./mfe1/mfe1.module').then((m) => m.Mfe1Module),
  },</code></pre>
4. Save and close the app-routing.module.ts file.
</td><td>App developer</td></tr>
</table>

Task	Description	Skills required

Edit the **webpack.config.js** file.	1. Open the **webpack.config.js** file.
	2. Edit the `For remotes` section to match the following:
	<pre>// For remotes (please adjust)
	    name: "mfe1",
	    filename: "remoteEntry.js",
	    exposes: {
	        './Module': './src/app/mfe1/mfe1.module.ts',
	    },</pre>3. In the `shared` section, add any dependencies that the `mfe1` application shares with the shell application:
	<pre>shared: share({
	        "@angular/core": { singleton: true, strictVersion: true, requiredVersion: 'auto' },
	        "@angular/common": { singleton: true, strictVersion: true, requiredVersion: 'auto' },
	        "@angular/common/http": { singleton: true, strictVersion: true, requiredVersion: 'auto' },</pre>
		App developer

Task	Description	Skills required

	``
	        "@angular/
	router": { singleton
	: true, strictVer
	sion: true, requiredV
	ersion: 'auto' },
	
	        ...shared
	Mappings.getDescri
	ptors()
	    })
	``
	4. Save and close the **webpack.config.js** file.	

Adjust the HTML content.	1. Open the **app.compo nent.html** file. 2. Replace the contents with the following: `<router-outlet></router-outlet>` 3. Save and close the **app.component.html** file.	App developer

Run the applications locally

Task	Description	Skills required
Run the `mfe1` application.	1. Enter the following command to start the `mfe1` application: `npm start`	App developer

Task	Description	Skills required
	2. In a web browser, access `http://localhost:5000`. 3. Verify that the micro-frontend can be run standalone. The `mfe1` application should properly render without any errors.	
Run the shell application.	1. Enter the following command to start the shell application: `npm start` 2. In a web browser, access `http://localhost:4200/mfe1`. 3. Verify that the `mfe1` micro-frontend is embedded in the shell application. The portal application should properly render without any errors, and the `mfe1` application should be embedded within it.	App developer

Refactor the shell application to handle a micro-frontend loading error

Task	Description	Skills required
Create a module and component.	In the root folder of the shell application, enter the following commands	App developer

Task	Description	Skills required

	to create a module and component for an error page:
	`ng g module error-page --routing`
	`ng g c error-page`	

Adjust the HTML content.	1. Open the **error-page.component.html** file. 2. Replace the contents with the following: `<p>Sorry, this page is not available.</p>` 3. Save and close the **error-page.component.html** file.	App developer
Set the default routing path.	1. Open the **error-page-routing.module.ts** file. 2. Set the default route as the following: `{ path: '', component: ErrorPageComponent },` 3. Save and close the **error-page-routing.module.ts** file.	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a function to load micro-frontends.</td><td>

1. Open the **app-routi ng.module.ts** file.
2. Create the following function:

```
function loadMFE(url:
 string) {
    return loadRemot
eModule({
        type:
 'module',
        remoteEntry:
 `${url}/remoteEntr
y.js`,
        exposedMo
dule: './Module'
    })
    .then(m =>
 m.Mfe1Module)
    .catch(
        () =>
 import('./error-pa
ge/error-page.modu
le').then(m =>
 m.ErrorPageModule)
    );
}
```

3. Modify the `mfe1` route to be the following:

```
{
    path: 'mfe1',
    loadChildren:
 () => loadMFE(e
nvironment.mfe1URL)
},
```

</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
	4. Save and close the **app-routing.module.ts** file.	

Test the error handling.	1. If it is not already running, enter the following command to start the shell application:
	`npm start`
	2. In a web browser, access `http://localhost:4200/mfe1` .
	3. Verify that the error page is rendered. You should see the following text:
	`Sorry, this page is not available.`
		App developer

Deploy the applications by using AWS Amplify

Task	Description	Skills required
Deploy the micro-frontend.	1. In the Amplify CLI, navigate to the root folder of the micro-frontend application. 2. Enter the following command to initialize Amplify: `amplify init`	App developer, AWS DevOps

Task	Description	Skills required
	3. When you are prompted to enter a name for your Amplify project, press **Enter**. This re-uses the name from the **package.json** file. 4. When you are prompted to initialize the project with the above configuration, enter `Yes`. 5. When you are prompted to select an authentication method, choose `AWS Profile`. 6. Select the profile you want to use. 7. Wait for Amplify to initialize the project. When this process is complete, you receive a confirmation message in the terminal. 8. Enter the following command to add an Amplify hosting category to the micro-frontend: `amplify add hosting` 9. When you are prompted to select the plugin module, choose `Hosting with Amplify Console`.	

Task	Description	Skills required

	10. When you are prompted to choose a type, choose `Manual deployment` .
	11. Install the project npm dependencies by entering the following command:
	`npm install`
	12. Publish the application to the Amplify console by entering the following command:
	`amplify publish -y`
	When publishing is complete, Amplify returns the URL of the micro-frontend.
	13. Copy the URL. You need this value to update the shell application.	

Task	Description	Skills required

Deploy the shell application.	1. In the **src/app/environments** folder, open the **environments.prod.ts** file.
	2. Replace the `mfe1URL` value with the URL of the deployed micro-frontend:
	`export const environment = {`
	`   production: true,`
	`   mfe1URL: 'https://<env>.<Amplify-app-ID>.amplifyapp.com'`
	` };`
	3. Save and close the **environments.prod.ts** file.
	4. In the Amplify CLI, navigate to the root folder of the shell application.
	5. Enter the following command to initialize Amplify:
	`amplify init`
	6. When you are prompted to enter a name for your Amplify project, press **Enter**. This re-uses the name from the **package.json** file.
	7. When you are prompted to initialize the project with
		App developer, App owner

Task	Description	Skills required

	the above configuration, enter `Yes`.
	8. When you are prompted to select an authentication method, choose `AWS Profile`.
	9. Select the profile you want to use.
	10. Wait for Amplify to initialize the project. When this process is complete, you receive a confirmation message in the terminal.
	11. Add an Amplify hosting category to the shell application:
	`amplify add hosting`
	12. When you are prompted to select the plugin module, choose `Hosting with Amplify Console`.
	13. When you are prompted to choose a type, choose `Manual deployment` .
	14. Install project npm dependencies by entering the following command:
	`npm install`
	15. Publish the shell application to the Amplify console	

Task	Description	Skills required
	by entering the following command: `amplify publish -y` When publishing is complete, Amplify returns the URL of the deployed shell application. 16. Make note of the URL for the shell application.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Enable CORS.</td><td>Because the shell and micro-frontend applications are hosted independently on different domains, you must enable cross-origin resource sharing (CORS) on the micro-frontend. This allows the shell application to load the content from a different origin. To enable CORS, you add custom headers.

1. In the Amplify CLI, navigate to the root folder of the micro-frontend.
2. Enter the following command:
<pre><code>amplify configure hosting</code></pre>3. When you are prompted to configure custom settings, enter Y.
4. Sign in to the AWS Management Console, and then open the Amplify console.
5. Choose the micro-frontend.
6. In the navigation pane, choose Hosting, and then choose Custom headers.
7. Choose Edit.</td><td>App developer, AWS DevOps</td></tr>
</table>

Task	Description	Skills required

	8. In the **Edit custom headers** window, enter the following:
	``customHeaders:
	  - pattern: '*.js'
	    headers:
	      - key: Access-Control-Allow-Origin
	        value: '*'
	      - key: Access-Control-Allow-Methods
	        value: 'GET, OPTIONS'
	      - key: Access-Control-Allow-Headers
	        value: '*'```
	9. Choose **Save**.
	10. Redeploy the micro-frontend to apply the new custom headers.	

Task	Description	Skills required
Create a rewrite rule on the shell application.	The Angular shell application is configured to use HTML5 routing. If the user performs a hard refresh, Amplify tries to load a page from the current URL. This generates a 403 error. To avoid this, you add a rewrite rule in the Amplify console. To create the rewrite rule, follow these steps: 1. In the Amplify CLI, navigate to the root folder of the shell application. 2. Enter the following command: `amplify configure hosting` 3. When you are prompted to configure custom settings, enter Y. 4. Open the [Amplify console](https://console.aws.amazon.com/amplify/). 5. Choose the shell applicati on. 6. In the navigation pane, choose **Hosting**, and then choose **Rewrites and redirects**. 7. On the **Rewrites and redirects** page, choose **Manage redirects**.	App developer, AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

8. Choose **Open text editor**.
9. In the JSON editor, enter the following redirect:

```
[
    {
        "source": "/<*>",
        "target": "/index.html",
        "status": "404-200",
        "condition": null
    }
]
```

10. Choose **Save**.

</td><td></td></tr>
<tr><td>Test the web portal.</td><td>

1. In a web browser, enter the URL of the deployed shell application.
2. Verify that the shell application and the micro-frontend load properly.

</td><td>App developer</td></tr>
</table>

Clean up resources

Task	Description	Skills required
Delete the applications.	If you no longer need the shell and micro-frontend applications, delete them. This helps prevent charges for resources that you aren't using.	General AWS

Task	Description	Skills required
	1. Sign in to the AWS Management Console, and then open the [Amplify console](). 2. Choose the micro-fro ntend. 3. In the navigation pane, choose **App settings**, and then choose **General settings**. 4. Choose **Delete app**. 5. In the confirmation window, enter `delete`, and then choose **Delete app**. 6. Repeat these steps to delete the shell applicati on.	

Troubleshooting

Issue	Solution
No AWS profile available when running the `amplify init` command	If you don't have an AWS profile configure d, you can still proceed with the `amplify init` command. However, you need to select the `AWS access keys` option when you're prompted for the authentication method. Have your AWS access key and secret key available. Alternatively, you can configure a named profile for the AWS CLI. For instructions, see

Issue	Solution
	Configuration and credential file settings in the AWS CLI documentation.
Error loading remote entries	If you encounter an error when loading the remote entries in the **main.ts** file of the shell application, make sure that the `environment.mfe1URL` variable is set correctly. The value of this variable should be the URL of the micro-frontend.
404 error when accessing the micro-frontend	If you get a 404 error when trying to access the local micro-frontend, such as at `http://localhost:4200/mfe1`, check the following: • For the shell application, make sure that the routing configuration in the **app-routing.module.ts** file is set up correctly, and make sure that the `loadRemoteModule` function is properly calling the micro-frontend. • For the micro-frontend, verify that the **webpack.config.js** file has the correct `exposes` configuration, and make sure that the **remoteEntry.js** file is being generated correctly.

Additional information

AWS documentation

- Understanding and implementing micro-frontends on AWS (AWS Prescriptive Guidance)
- Amplify CLI (Amplify documentation)
- Amplify Hosting (Amplify documentation)

Other references

- Module Federation
- Node.js
- Angular
- @angular-architects/module-federation

Deploy a React-based single-page application to Amazon S3 and CloudFront

Jean-Baptiste Guillois, Amazon Web Services

Summary

A single-page application (SPA) is a website or web application that dynamically updates the contents of a displayed webpage by using JavaScript APIs. This approach enhances the user experience and performance of a website because it updates only new data instead of reloading the entire webpage from the server.

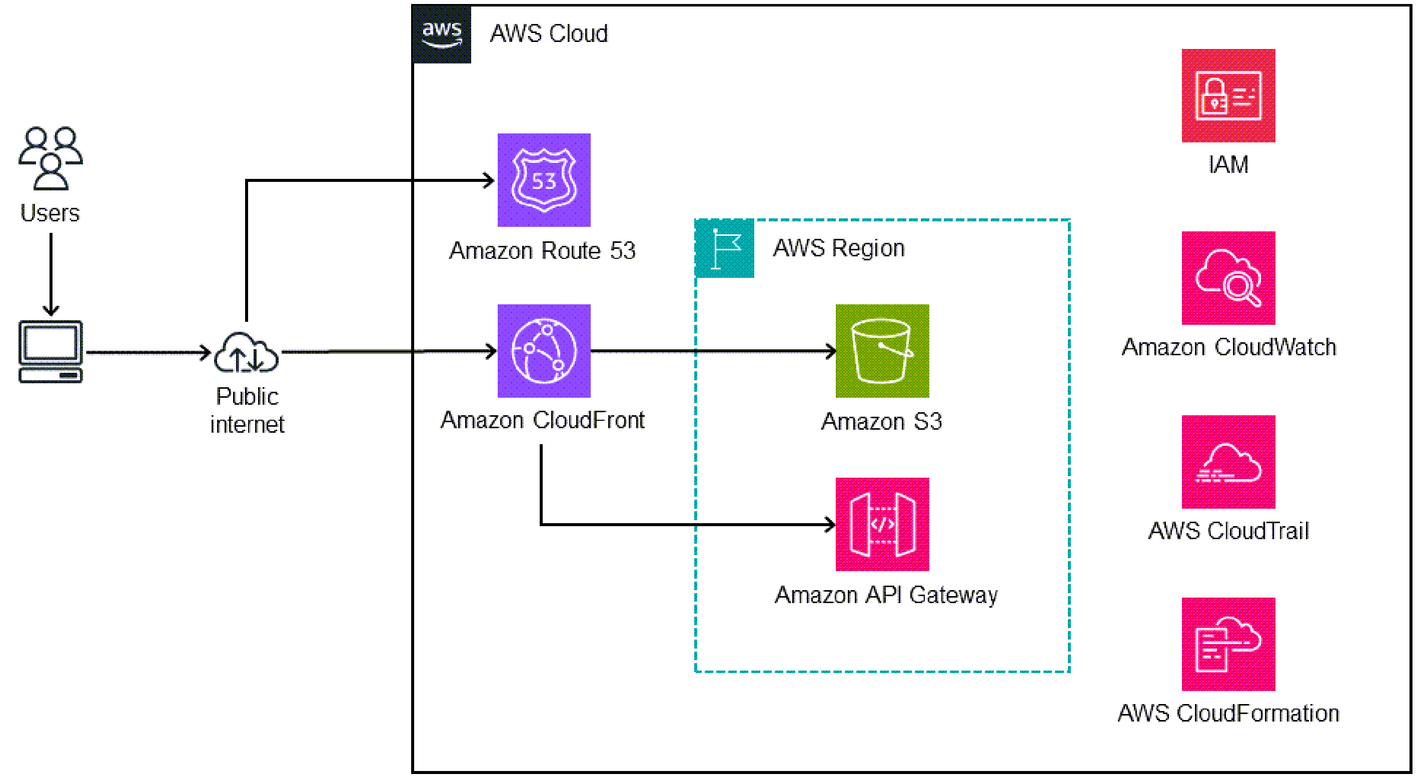
This pattern provides a step-by-step approach to coding and hosting an SPA that's written in React on Amazon Simple Storage Service (Amazon S3) and Amazon CloudFront. The SPA in this pattern uses a REST API that's configured in Amazon API Gateway and exposed through an Amazon CloudFront distribution to simplify cross-origin resource sharing (CORS) management.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Node.js and `npm`, installed and configured. For more information, see the Downloads section of the Node.js documentation.
- Yarn, installed and configured. For more information, see the Yarn documentation.
- Git, installed and configured. For more information, see the Git documentation.

Architecture



This architecture is automatically deployed by using AWS CloudFormation (infrastructure as code). It uses Regional services such as Amazon S3 to store the static assets and Amazon CloudFront with Amazon API Gateway to expose Regional API (REST) endpoints. The application logs are collected by using Amazon CloudWatch. All AWS API calls are audited in AWS CloudTrail. All security configuration (for example, identities and permissions) is managed in AWS Identity and Access Management (IAM). Static content is delivered through the Amazon CloudFront content delivery network (CDN), and DNS queries are handled by Amazon Route 53.

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- Amazon CloudFront speeds up distribution of your web content by delivering it through a worldwide network of data centers, which lowers latency and improves performance.

- AWS CloudTrail helps you audit the governance, compliance, and operational risk of your AWS account.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Route 53 is a highly available and scalable DNS web service.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code

This pattern's sample application code is available in the GitHub React-based CORS single-page application repository.

Best practices

By using Amazon S3 object storage, you can store your application’s static assets in a secure, highly resilient, performant, and cost-effective way. There is no need to use a dedicated container or an Amazon Elastic Compute Cloud (Amazon EC2) instance for this task.

By using the Amazon CloudFront content delivery network, you can reduce the latency your users might experience when they access your application. You can also attach a web application firewall (AWS WAF) to protect your assets from malicious attacks.

Epics

Locally build and deploy your application

Task	Description	Skills required
Clone the repository.	Run the following command to clone the sample application's repository: `git clone https://github.com/aws-samples/react-cors-spa`	App developer, AWS DevOps

Task	Description	Skills required
	`react-cors-spa && cd react-cors-spa`	
Locally deploy the application.	1. In the project directory, run the `npm install` command to initiate the application dependencies. 2. Run the `yarn dev` command to start the application locally.	App developer, AWS DevOps
Locally access the application.	Open a browser window and enter the `http://localhost:3000` URL to access the application.	App developer, AWS DevOps

Deploy the application

Task	Description	Skills required
Deploy the AWS CloudFormation template.	1. Sign in to the AWS Management Console, and then open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/). 2. Choose **Create Stack**, and then choose **With new resources (standard)**. 3. Choose **Upload a template file**. 4. Choose **Choose file**, choose the `react-cors-spa-stack.yaml` file from the cloned repository, and then choose **Next**.	App developer, AWS DevOps

Task	Description	Skills required
	5. Enter a name for your stack, and then choose **Next**. 6. Keep all default options, and then choose **Next**. 7. Review the final settings for your stack, and then choose **Create stack**.	
Customize your application source files.	1. After your stack is deployed, open the **Output** tab and identify the `Bucket` name and `APIDomain` value. 2. Copy the CloudFront distribution domain for the REST API. 3. Navigate to `<project_root>/src/pages/index.tsx`, and then insert or paste this domain into the `APIEndPoint` variable value on line 13 of the `index.tsx` file.	App developer
Build the application package.	In your project directory, run the `yarn build` command to build the application package.	App developer

Task	Description	Skills required
Deploy the application package.	1. Open the [Amazon S3 console](https://console.aws.amazon.com/s3/). 2. Identify and choose the S3 bucket that was created earlier by the CloudFormation stack. 3. Choose **Upload**, and then choose **Add files**. 4. Choose the content of your `out` folder. 5. Choose **Add folder**, and then choose the `_next` directory. **Important** Choose the `_next` directory, not the contents. 6. Choose **Upload** to upload the files and directory to your S3 bucket.	App developer, AWS DevOps

Test the application

Task	Description	Skills required
Access and test the application.	Open a browser window, and then paste the CloudFront distribution domain (the `SPADomain` output from the CloudFormation stack that	App developer, AWS DevOps

Task	Description	Skills required
	you deployed previously) to access the application.	

Clean up the resources

Task	Description	Skills required
Delete the S3 bucket contents.	1. Open the [Amazon S3 console](#) and choose the bucket that was created earlier by the stack (the first bucket whose name starts with `react-cors-spa-`). 2. Choose **Empty** to delete the bucket’s contents. 3. Choose the second bucket that was created earlier by the stack (the second bucket whose name starts with `react-cors-spa-` and ends with `-logs`). 4. Choose **Empty** to delete the bucket’s contents.	AWS DevOps, App developer
Delete the AWS CloudFormation stack.	1. Open the [AWS CloudFormation console](#) and choose the stack that you created earlier. 2. Choose **Delete** to delete the stack and all related resources.	AWS DevOps, App developer

Related resources

To deploy and host your web application, you can also use [AWS Amplify Hosting](), which provides a Git-based workflow for hosting full-stack, serverless web apps with continuous deployment. Amplify Hosting is part of [AWS Amplify](), which provides a set of purpose-built tools and features that enable frontend web and mobile developers to quickly and easily build full-stack applications on AWS.

Additional information

To handle invalid URLs requested by the user that might generate 403 errors, a custom error page that's configured in the CloudFront distribution catches 403 errors and redirects them to the application entry point (`index.html`).

To simplify the management of CORS, the REST API is exposed through a CloudFront distribution.

Deploy an Amazon API Gateway API on an internal website using private endpoints and an Application Load Balancer

Saurabh Kothari, Amazon Web Services

Summary

This pattern shows you how to deploy an Amazon API Gateway API on an internal website that's accessible from an on-premises network. You learn to create a custom domain name for a private API by using an architecture that's designed with private endpoints, an Application Load Balancer, AWS PrivateLink, and Amazon Route 53. This architecture prevents the unintended consequences of using a custom domain name and proxy server to help with domain-based routing on an API. For example, if you deploy a virtual private cloud (VPC) endpoint in a non-routable subnet, your network can't reach API Gateway. A common solution is to use a custom domain name and then deploy the API in a routable subnet, but this can break other internal sites when the proxy configuration passes traffic (`execute-api.{region}.vpce.amazonaws.com`) to AWS Direct Connect. Finally, this pattern can help you meet organizational requirements for using a private API that's unreachable from the internet and a custom domain name.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Server Name Indication (SNI) certificate for your website and API
- A connection from an on-premises environment to an AWS account that's set up by using AWS Direct Connect or AWS Site-to-Site VPN
- A private hosted zone with a corresponding domain (for example, domain.com) that's resolved from an on-premises network and forwards DNS queries to Route 53
- A routable private subnet that's reachable from an on-premises network

Limitations

For more information about quotas (formerly referred to as limits) for load balancers, rules, and other resources, see Quotas for your Application Load Balancers in the Elastic Load Balancing documentation.

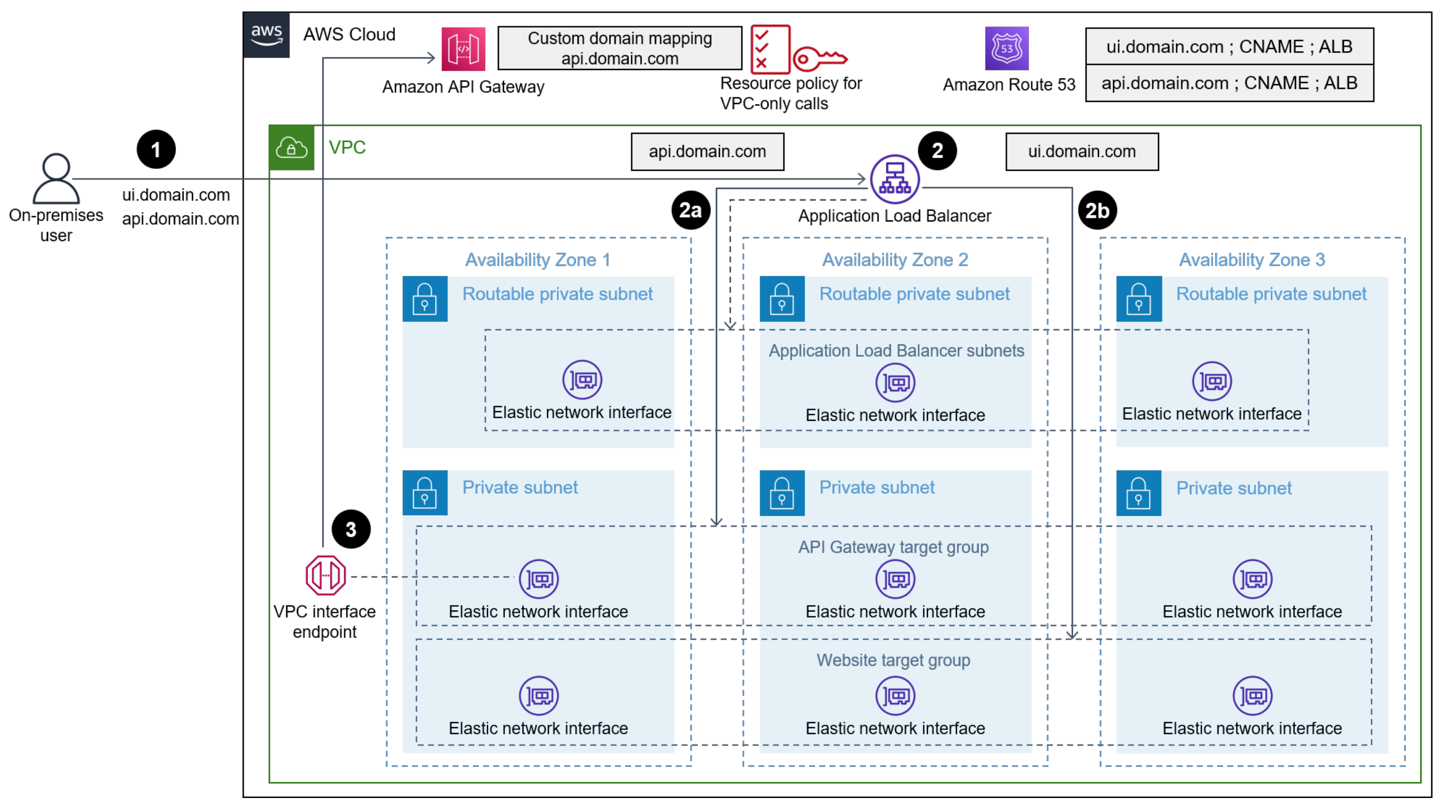
Architecture

Technology stack

- Amazon API Gateway
- Amazon Route 53
- Application Load Balancer
- AWS Certificate Manager
- AWS PrivateLink

Target architecture

The following diagram shows how an Application Load Balancer is deployed in a VPC that directs web traffic to a website target group or API Gateway target group based on Application Load Balancer listener rules. The API Gateway target group is a list of IP addresses for the VPC endpoint in API Gateway. API Gateway is configured to make the API private with its resource policy. The policy denies all calls that are not from a specific VPC endpoint. Custom domain names in API gateway are updated to use api.domain.com for the API and its stage. Application Load Balancer rules are added to route traffic based on the host name.



The diagram shows the following workflow:

1. A user from an on-premises network tries to access an internal website. The request is sent to ui.domain.com and api.domain.com. Then, the request is resolved to the internal Application Load Balancer of the routable private subnet. The SSL is terminated at the Application Load Balancer for ui.domain.com and api.domain.com.
2. Listener rules, configured on the Application Load Balancer, check for the host header.

   a. If the host header is api.domain.com, the request is forwarded to the API Gateway target group. The Application Load Balancer initiates a new connection to API Gateway over port 443.

   b. If the host header is ui.domain.com, the request is forwarded to the website target group.
3. When the request reaches API Gateway, the custom domain mapping configured in API Gateway determines the hostname and which API to run.

Automation and scale

The steps in this pattern can be automated by using AWS CloudFormation or the AWS Cloud Development Kit (AWS CDK). To configure the target group of the API Gateway calls, you must use a custom resource to retrieve the IP address of the VPC endpoint. API calls to describe-vpc-endpoints and describe-network-interfaces return the IP addresses and the security group, which can be used to create the API target group of IP addresses.

Tools

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- Amazon Route 53 is a highly available and scalable DNS web service.
- AWS Certificate Manager (ACM) helps you create, store, and renew public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS PrivateLink helps you create unidirectional, private connections from your VPCs to services outside of the VPC.

Epics

Create an SNI certificate

Task	Description	Skills required
Create an SNI certificate and import the certificate into ACM.	1. Create an SNI certificate for ui.domain.com and api.domain.com. For more information, see [Choosing how CloudFront serves HTTPS requests] in the Amazon CloudFront documentation. 2. Import the SNI certificates into AWS Certificate Manager (ACM). For more information, see [Importing certificates into AWS Certificate Manager] in the ACM documentation.	Network administrator

Deploy a VPC endpoint in a non-routable private subnet

Task	Description	Skills required
Create an interface VPC endpoint in API Gateway.	To create an interface VPC endpoint, follow the instructions from [Access an AWS service using an interface VPC endpoint] in the Amazon Virtual Private Cloud (Amazon VPC) documentation.	Cloud administrator

Configure the Application Load Balancer

Task	Description	Skills required
Create a target group for your application.	Create a target group for the UI resources of your applicati on.	Cloud administrator
Create a target group for the API Gateway endpoint.	1. Create a target group with an IP address type, and then add the IP address of the VPC endpoint for the API Gateway endpoint to the target group. 2. Configure health checks for your target groups with success code 403. 403 is required because the VPC endpoint for the API Gateway returns a 403 code when it is invoked without any headers by the target group health check.	Cloud administrator
Create an Application Load Balancer.	1. Create an Application Load Balancer (internal) in a routable private subnet. 2. Add the 443 listener to the Application Load Balancer, and the choose the certificate from ACM.	Cloud administrator
Create listeners rules.	Create listener rules to do the following:	Cloud administrator

Task	Description	Skills required
	1. Forward the host api.domain.com to the API Gateway target group 2. Forward the host ui.domain.com to the target group for the UI resources	

Configure Route 53

Task	Description	Skills required
Create a private hosted zone.	[Create a private hosted zone](#) for domain.com.	Cloud administrator
Create domain records.	[Create CNAME records](#) for the following: • An API with the value set to the DNS name of the Application Load Balancer • A UI with the value set to the DNS name of the Application Load Balancer	Cloud administrator

Create a private API endpoint in API Gateway

Task	Description	Skills required
Create and configure a private API endpoint.	1. To create a private API endpoint, follow the instructions from [Creating a private API in Amazon](#)	App developer, Cloud administrator

Task	Description	Skills required
	API Gateway in the API Gateway documentation. 2. Configure the resource policy to allow calls to only the API from the VPC endpoint. For more information, see Controlling access to an API with API Gateway resource policies in the API Gateway documentation.	
Create a custom domain name.	1. Create a custom domain name for api.domain.com. For more information, see Setting up custom domain names for REST APIs in the API Gateway documentation. 2. Select the created API and stage. For more information, see Working with API mappings for REST APIs in the API Gateway documentation.	Cloud administrator

Related resources

- Amazon API Gateway
- Amazon Route 53
- Application Load Balancer
- AWS PrivateLink
- AWS Certificate Manager

Embed an Amazon QuickSight dashboard in a local Angular application

Sean Griffin and Milena Godau, Amazon Web Services

Summary

This pattern provides guidance for embedding an Amazon QuickSight dashboard into a locally hosted Angular application for development or testing. The embedded analytics feature in QuickSight doesn't support this functionality natively. It requires an QuickSight account with an existing dashboard and knowledge of Angular.

When you work with embedded QuickSight dashboards, you would typically have to host your application on a web server to view the dashboard. This makes development more difficult, because you have to continuously push your changes to the web server to make sure everything is behaving correctly. This pattern shows how to run a locally hosted server and use QuickSight embedded analytics to make the development process easier and more streamlined.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- An active QuickSight account with session capacity pricing
- QuickSight Embedding SDK installed
- Angular CLI installed
- Familiarity with Angular
- mkcert installed

Limitations

- This pattern provides guidance on embedding a QuickSight dashboard by using the `ANONYMOUS` (publicly accessible) authentication type. If you are using AWS Identity and Access Management (IAM) or QuickSight authentication with your embedded dashboards, the provided code won't apply. However, the steps for hosting the Angular application in the Epics section are still valid.
- Using the **GetDashboardEmbedUrl** API with the `ANONYMOUS` identity type requires a QuickSight capacity pricing plan.

Versions

- Angular CLI version 13.3.4
- QuickSight Embedding SDK version 2.3.1

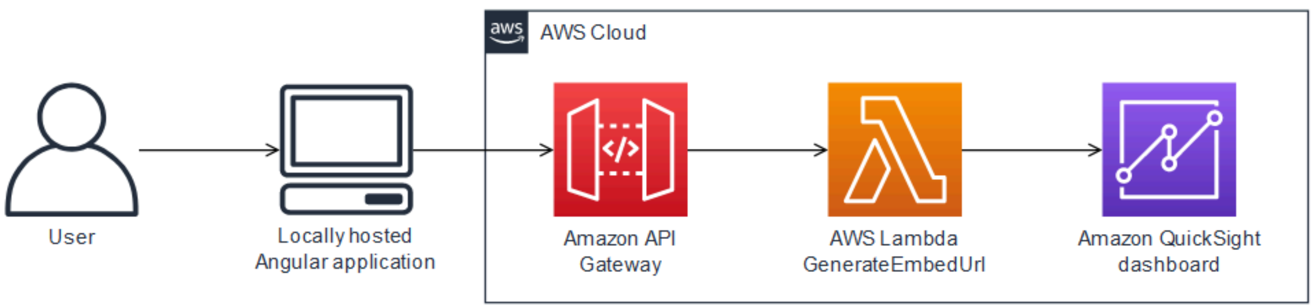
Architecture

Technology stack

- Angular frontend
- AWS Lambda and Amazon API Gateway backend

Architecture

In this architecture, the HTTP APIs in API Gateway enable the local Angular application to call the Lambda function. The Lambda function returns the URL for embedding the QuickSight dashboard.



Automation and scale

You can automate the backend deployment by using AWS CloudFormation or AWS Serverless Application Model (AWS SAM).

Tools

Tools

- Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell.
- QuickSight Embedding SDK is used to embed QuickSight dashboards into your HTML.

- mkcert is a simple tool for creating locally trusted development certificates. It requires no configuration. mkcert is required because QuickSight allows only HTTPS requests for embedding dashboards.

AWS services

- Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- Amazon QuickSight is a business analytics service for building visualizations, performing ad hoc analyses, and getting business insights from your data.

Epics

Generate EmbedURL

Task	Description	Skills required
Create an EmbedUrl policy.	Create an IAM policy named **QuicksightGetDashboardEmbedUrl** that has the following properties. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "quicksight:GetDashboardEmbedUrl",`	AWS administrator

Task	Description	Skills required

	``
	   "quickSight:GetAno
	nymousUserEmbedUrl"
	                  ],
	
	 "Resource": "*"
	            }
	      ]
	}
	``	

Task	Description	Skills required

Create the Lambda function.	1. On the Lambda console, open the [Functions page](https://console.aws.amazon.com/lambda/home#/functions).
	
	2. Choose **Create Function**.
	
	3. Choose **Author from scratch**.
	
	4. For **Function name**, enter `get-qs-embed-url` .
	
	5. For **Runtime**, choose **Python 3.9**.
	
	6. Choose **Create Function**.
	
	7. On the **Code** tab, copy the following code into the Lambda function.
	
	`import json`
	`import boto3`
	`from botocore.exceptions import ClientError`
	`import time`
	`from os import environ`
	
	`qs = boto3.client('quicksight',region_name='us-east-1')`
	`sts = boto3.client('sts')`
	
	`ACCOUNT_ID = boto3.client('sts').get_caller_identity().get('Account')`
		App developer

Task	Description	Skills required

	``
	DASHBOARD_ID =
	 environ['DASHBOARD
	_ID']
	
	def getDashboardURL(ac
	countId, dashboardId,
	 quicksightNamespac
	e, resetDisabled,
	 undoRedoDisabled):
	  try:
	    response = qs.get_da
	shboard_embed_url(
	      AwsAccountId =
	 accountId,
	      DashboardId =
	 dashboardId,
	      Namespace =
	 quicksightNamespace,
	      IdentityType =
	 'ANONYMOUS',
	      SessionLi
	fetimeInMinutes = 600,
	      UndoRedoDisabled
	 = undoRedoDisabled,
	      ResetDisabled =
	 resetDisabled
	    )
	    return response
	
	  except ClientError as
	 e:
	    print(e)
	    return "Error
	 generating embeddedU
	RL: " + str(e)
	
	def lambda_handler(eve
	nt, context):
	  url = getDashbo
	ardURL(ACCOUNT_ID,
	 DASHBOARD_ID,
	``	

Task	Description	Skills required

	``
	 "default", True, True)
	['EmbedUrl']
	  return {
	      'statusCode':
	 200,
	      'url': url
	  }
	``
	8. Choose **Deploy**.	

Task	Description	Skills required
Add the dashboard ID as an environment variable.	Add `DASHBOARD_ID` as an environment variable to your Lambda function: 1. On the **Configuration** tab, choose **Environment variables**, **Edit**, **Add environment variable**. 2. Add an environment variable with the key `DASHBOARD_ID`. 3. To get the value of `DASHBOARD_ID`, navigate to your dashboard in QuickSight and copy the UUID at the end of the URL in your browser. For example, if the URL is `https://us-east-1.quicksight.aws.amazon.com/sn/dashboards/<dashboard-id>`, specify the `<dashboard-id>` part of the URL as the key value. 4. Choose **Save**.	App developer

Task	Description	Skills required
Add permissions for the Lambda function.	Modify the execution role of the Lambda function and add the **QuicksightGetDashboardEmbedUrl** policy to it. 1. On the **Configuration** tab, choose **Permissions**, and then choose the role name. 2. Choose **Attach policies**, search for `QuicksightGetDashboardEmbedUrl` , select its check box, and then choose **Attach policy**.	App developer

Task	Description	Skills required
Test the Lambda function.	Create and run a test event. You can use the "Hello World" template, because the function won't use any of the data in the test event. 1. Choose the **Test** tab. 2. Give your test event a name, and then choose **Save**. 3. To test your Lambda function, choose **Test**. The response should look similar to the following. `{ "statusCode": 200, "url": "\"https://us-east-1.quicksight.aws.amazon.com/embed/f1acc0786687783b9a4543a05ba929b3a/dashboards/... }` **Note** As mentioned in the *Prerequisites and limitations* section, your QuickSight account must be under a session capacity pricing plan. Otherwise, this step	App developer

Task	Description	Skills required
	will display an error message.	

Task	Description	Skills required
Create an API in API Gateway.	1. On the [API Gateway console](), choose **Create API**, and then choose **REST API**, **Build**. • For API name, enter `qs-embed-api` . • Choose **Create API**. 2. In **Actions**, choose **Create Method**. • Choose **GET** and confirm by choosing the checkmark. • Choose **Lambda Function** as the integrati on type. • For **Lambda Function**, enter `get-qs-embed-url`. • Choose **Save**. • In the **Add Permission to Lambda Function** box, choose **OK**. 3. Enable CORS. • In **Actions**, choose **Enable CORS**. • For **Access-Control-Allow-Origin**, enter `'https://my-qs-app.net:4200'` . • Choose **Enable CORS and replace existing**	App developer

Task	Description	Skills required
	CORS headers, and confirm. 4. Deploy the API. • For **Actions**, choose **Deploy API**. • For **Deployment stage**, choose **[New Stage]**. • For **Stage name**, enter `dev`. • Choose **Deploy**. • Copy the **Invoke URL**. **Note** `my-qs-app.net` can be any domain. If you want to use a different domain name, make sure to update the **Access-Control-Allow-Origin** information in step 3, and change `my-qs-app.net` in subsequent steps.	

Create the Angular application

Task	Description	Skills required
Create the application with the Angular CLI.	1. Create the application.	App developer

Task	Description	Skills required

	``
	ng new quicksight-app
	 --defaults
	cd quicksight-app/src
	/app
	``
	2. Create the dashboard component.
	``
	ng g c dashboard
	``
	3. Navigate to your `src/environments/environment.ts` file and add `apiUrl: '<Invoke URL from previous steps>'` to the environment object.
	``
	export const environme
	nt = {
	  production: false,
	  apiUrl: '<Invoke
	 URL from previous
	 steps>',
	};
	``	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add the QuickSight Embedding SDK.</td><td>1. Install the QuickSight Embedding SDK by running the following command in your project's root folder.
<pre>npm i amazon-quicksight-embedding-sdk</pre>2. Create a new <code>decl.d.ts</code> file in the <code>src</code> folder with the following content.
<pre>declare module 'amazon-quicksight-embedding-sdk';</pre></td><td>App developer</td></tr>
</table>

Task	Description	Skills required

Add code to your dashboard .component.ts file.	```
	import { Component,
	 OnInit } from '@angular
	/core';
	import { HttpClient }
	 from '@angular/common/
	http';
	import * as Quicksigh
	tEmbedding from
	 'amazon-quicksight-
	embedding-sdk';
	import { environme
	nt } from "../../en
	vironments/environ
	ment";
	import { take } from
	 'rxjs';
	import { Embedding
	Context } from 'amazon-
	quicksight-embedding-
	sdk/dist/types';
	import { createEmb
	eddingContext } from
	 'amazon-quicksight-
	embedding-sdk';
	
	@Component({
	  selector: 'app-dash
	board',
	  templateUrl: './
	dashboard.compo
	nent.html',
	  styleUrls: ['./dashb
	oard.component.scss']
	})
	export class Dashboard
	Component implements
	 OnInit {
	
	  constructor(private
	 http: HttpClient) { }
	``
		App developer

Task	Description	Skills required

	``
	  loadingError = false;
	  dashboard: any;
	
	  ngOnInit() {
	    this.GetDashboardU
	RL();
	  }
	
	  public GetDashbo
	ardURL() {
	    this.http.get(envi
	ronment.apiUrl)
	    .pipe(
	      take(1),
	    )
	    .subscribe((data:
	 any) => this.Dash
	board(data.url));
	  }
	
	  public async Dashboard
	(embeddedURL: any) {
	    var containerDiv
	 = document.getElemen
	tById("dashboardCo
	ntainer") || '';
	    const frameOptions
	 = {
	      url: embeddedURL,
	      container:
	 containerDiv,
	      height: "850px",
	      width: "100%",
	      resizeHei
	ghtOnSizeChangedEv
	ent: true,
	  }
	    const embedding
	Context: Embedding
	Context = await
	 createEmbeddingCon
	text();
	``	

Task	Description	Skills required

	``
	    this.dashboard =
	  embeddingContext.e
	mbedDashboard(fram
	eOptions);
	  }
	}
	``	

Add code to your dashboard .component.html file.	Add the following code to your `src/app/dashboard/dashboard.component.html` file. `<div id="dashboardContainer"></div>`	App developer
Modify your app.component.html file to load your dashboard component.	1. Delete the contents of the `src/app/app.component.html` file. 2. Add the following. `<app-dashboard></app-dashboard>`	App developer
Import HttpClientModule into your app.module.ts file.	1. At the top of the `src/app/app.module.ts` file, add the following. `import { HttpClientModule } from '@angular/common/http';` 2. Add `HttpClientModule` in the `imports` array for your `AppModule` .	App developer

Host the Angular application

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Configure mkcert.</td><td>

Note

The following commands are for Unix or MacOS machines. If you're using Windows, see the [Additional information](#) section for the equivalent **echo** command.

1. Create a local certificate authority (CA) on your machine.

```
mkcert -install
```

2. Configure `my-qs-app.net` to always redirect to your local PC.

```
echo "127.0.0.1 my-qs-app.net" | sudo tee -a /private/etc/hosts
```

3. Make sure that you're in the `src` directory of the Angular project.

```
mkcert my-qs-app.net 127.0.0.1
```

</td><td>App developer</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Configure QuickSight to allow your domain.</td><td>In QuickSight, choose your name in the upper-right corner, and then choose Manage Quicksight.
Navigate to Domains and Embedding.
Add <code>https://my-qs-app.net:4200</code> as an allowed domain.</td><td>AWS administrator</td></tr>
<tr><td>Test the solution.</td><td> Start a local Angular development server by running the following command. <pre><code>ng serve --host my-qs-app.net --port 4200 --ssl --ssl-key "./src/my-qs-app.net-key.pem" --ssl-cert "./src/my-qs-app.net.pem" -o</code></pre> This enables Secure Sockets Layer (SSL) with the custom certificate that you created earlier. When the build is complete, it opens a browser window and you can view your embedded QuickSight dashboard hosted locally in Angular. </td><td>App developer</td></tr>
</table>

Related resources

- Angular website
- Embedding QuickSight data dashboards for anonymous (unregistered) users (QuickSight documentation)
- QuickSight Embedding SDK
- mkcert tool

Additional information

If you're using Windows, run the Command Prompt window as an administrator, and configure `my-qs-app.net` to always redirect to your local PC by using the following command.

```
echo 127.0.0.1 my-qs-app.net >> %WINDIR%\System32\Drivers\Etc\Hosts
```

Explore full-stack cloud-native web application development with Green Boost

Ben Stickley and Amiin Samatar, Amazon Web Services

Summary

In response to the evolving needs of developers, Amazon Web Services (AWS) recognizes the critical demand for an efficient approach to developing cloud-native web applications. The AWS focus is on helping you to overcome common roadblocks associated with deploying web apps on the AWS Cloud. By harnessing the capabilities of modern technologies such as TypeScript, AWS Cloud Development Kit (AWS CDK), React, and Node.js, this pattern aims to streamline and expedite the development process.

Underpinned by the Green Boost (GB) toolkit, the pattern offers a practical guide to constructing web applications that fully use the extensive capabilities of AWS. It acts as a comprehensive roadmap, leading you through the process of deploying a fundamental CRUD (Create, Read, Update, Delete) web application integrated with Amazon Aurora PostgreSQL-Compatible Edition. This is accomplished by using the Green Boost command line interface (Green Boost CLI) and establishing a local development environment.

Following the successful deployment of the application, the pattern delves into key components of the web app, including infrastructure design, backend and frontend development, and essential tools such as cdk-dia for visualization, facilitating efficient project management.

Prerequisites and limitations

Prerequisites

- Git installed
- Visual Studio Code (VS Code) installed
- AWS Command Line Interface (AWS CLI) installed
- AWS CDK Toolkit installed
- Node.js 18 installed, or Node.js 18 with pnpm activated
- pnpm installed, if it isn't part of your Node.js installation
- Basic familiarity with TypeScript, AWS CDK, Node.js, and React
- An active AWS account

- An AWS account bootstrapped by using AWS CDK in `us-east-1`. The `us-east-1` AWS Region is required for support of the Amazon CloudFront Lambda@Edge functions.
- AWS security credentials, including `AWS_ACCESS_KEY_ID`, correctly configured in your terminal environment
- For Windows users, a terminal in administrator mode (to accommodate the way pnpm handles node modules)

Product versions

- AWS SDK for JavaScript version 3
- AWS CDK version 2
- AWS CLI version 2.2
- Node.js version 18
- React version 18

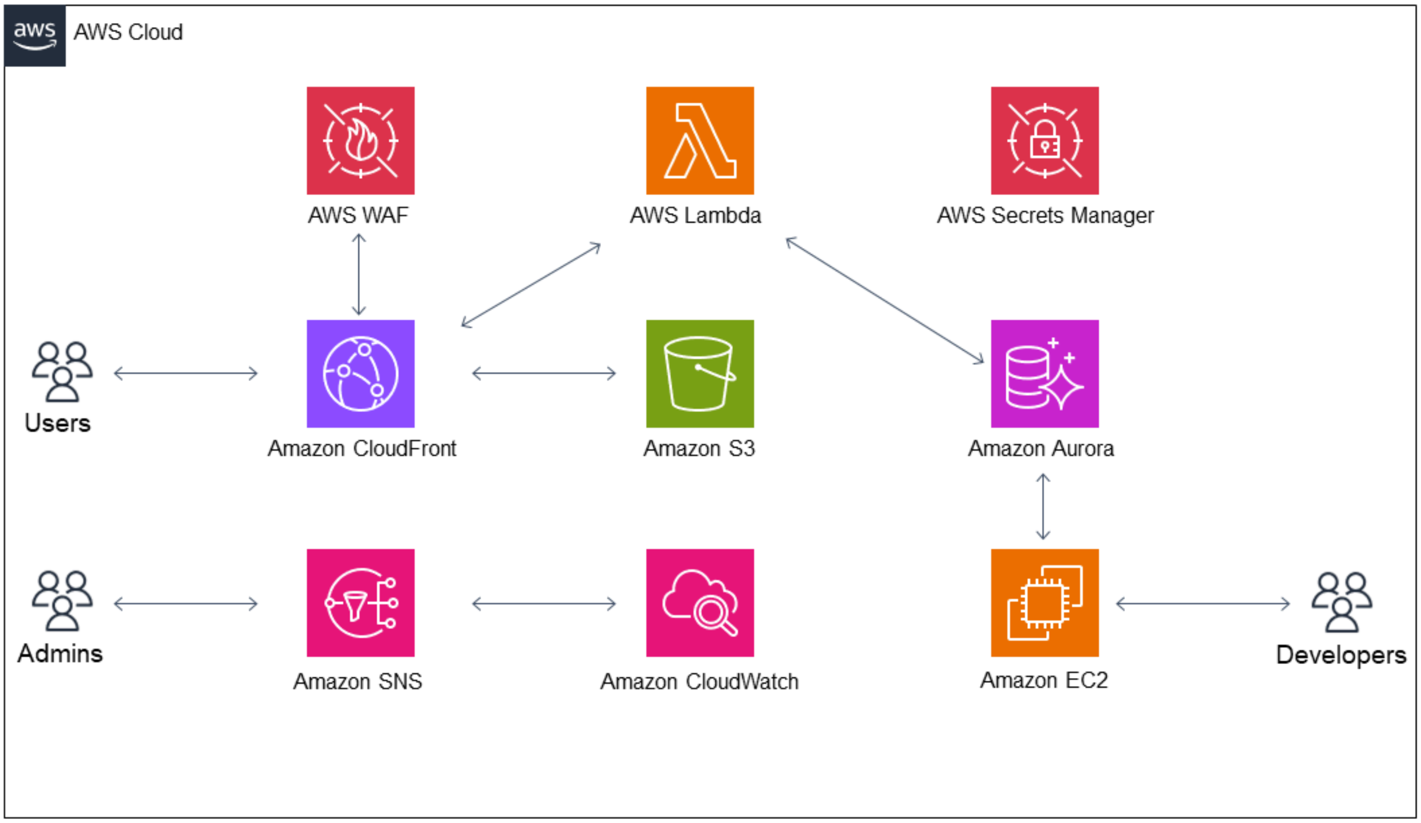
Architecture

Target technology stack

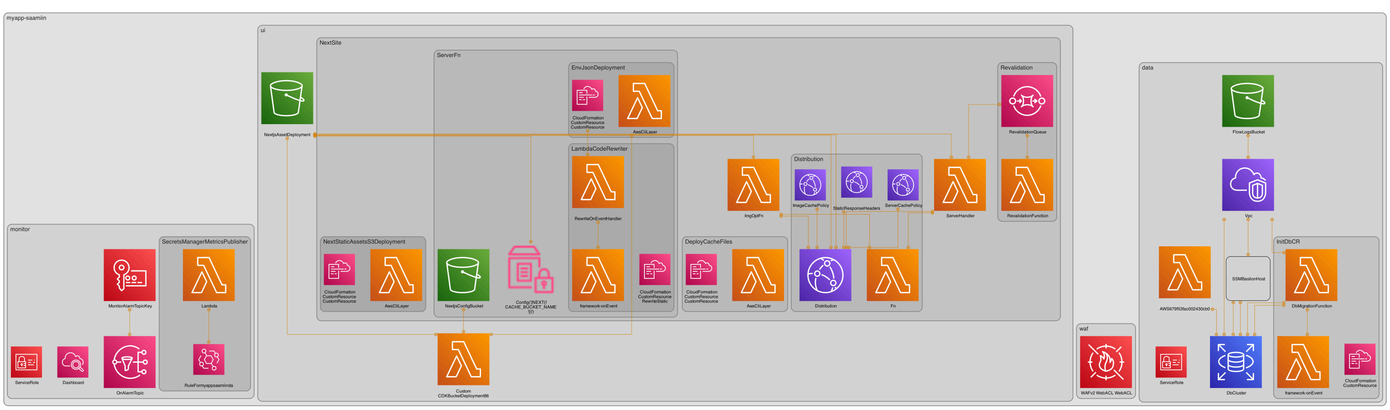
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon CloudFront
- Amazon CloudWatch
- Amazon Elastic Compute Cloud (Amazon EC2)
- AWS Lambda
- AWS Secrets Manager
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Storage Service (Amazon S3)
- AWS WAF

Target Architecture

The following diagram shows that user requests pass through Amazon CloudFront, AWS WAF, and AWS Lambda before interacting with an S3 bucket, an Aurora database, an EC2 instance, and ultimately reaching developers. Administrators, on the other hand, use Amazon SNS and Amazon CloudWatch for notifications and monitoring purposes.



To gain a more in-depth look at the application after deployment, you can create diagram by using [cdk-dia](https://github.com/pistazie/cdk-dia), as shown in the following example.



These diagrams showcase the web application architecture from two distinct angles. The cdk-dia diagram offers a detailed technical view of the AWS CDK infrastructure, highlighting specific AWS services such as Amazon Aurora PostgreSQL-Compatible and AWS Lambda. In contrast, the other diagram takes a broader perspective, emphasizing the logical flow of data and user interactions.

The key distinction lies in the level of detail: The cdk-dia delves into technical intricacies, while the first diagram provides a more user-centric view.

Creation of the cdk-dia diagram is covered in the epic *Understand the app infrastructure by using AWS CDK*.

Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon CloudFront speeds up distribution of your web content by delivering it through a worldwide network of data centers, which lowers latency and improves performance.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale. This pattern uses AWS Systems Manager Session Manager.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

- AWS WAF is a web application firewall that helps you monitor HTTP and HTTPS requests that are forwarded to your protected web application resources

Other tools

- Git is an open-source, distributed version control system.
- Green Boost is a toolkit for building web apps on AWS.
- Next.js is a React framework for adding features and optimizations.
- Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.
- pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.
- pnpm is a package manager for Node.js project dependencies.

Best practices

See the Epics section for more information about the following recommendations:

- Monitor infrastructure by using Amazon CloudWatch Dashboards and alarms.
- Enforce AWS best practices by using cdk-nag to run static infrastructure as code (IaC) analysis.
- Establish DB port forwarding through SSH (Secure Shell) tunneling with Systems Manager Session Manager, which is more secure than having a publicly exposed IP address.
- Manage vulnerabilities by running `pnpm audit`.
- Enforce best practices by using ESLint to perform static TypeScript code analysis, and Prettier to standardize code formatting.

Epics

Deploy a CRUD web app with Aurora PostgreSQL-Compatible

Task	Description	Skills required
Install the Green Boost CLI.	To install Green Boost CLI, run the following command.	App developer

Task	Description	Skills required
	`pnpm add -g gboost`	
Create a GB app.	1. To create an app by using Green Boost, run the command `gboost create`. 2. Choose the `CRUD App with Aurora PostgreSQL` template.	App developer

Task	Description	Skills required
Install dependencies and deploy the app.	1. Navigate to the project directory: `cd <your directory>` . 2. To install dependencies, run the command `pnpm i`. 3. Navigate to the infra directory: `cd infra`. 4. To deploy the app locally, run the command `pnpm deploy:local` . This is an alias for a `cdk deploy ...` command defined in `infra/package.json` . Wait for deployment to finish (approximately 20 minutes). While you're waiting, monitor AWS CloudFormation stacks in the CloudFormation console. Notice how the constructs defined in the code map to the resource deployed. Review the [CDK Construct tree view](#) in the CloudFormation console.	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Access the app.</td><td>After deploying your GB app locally, you can access it using the CloudFront URL. The URL is printed in the terminal output, but it can be a bit overwhelming to find. To find it more quickly, use the following steps:

1. Open the terminal where you ran the <code>pnpm deploy:local</code> command.
2. Look for a section in the terminal output that resembles the following text.

<pre>myapp5stickbui9C39A55A.CloudFrontDomainName = d1q16n5pof924c.cloudfront.net</pre>
The URL will be unique to your deployment.

Alternatively, you can find the CloudFront URL by accessing the Amazon CloudFront console:

1. Sign in to the AWS Management Console and</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
	navigate to the CloudFront service. 2. Look for the latest deployed distribution in the list. Copy the **Domain Name** that is associated with the distribution. It will look similar to `your-unique-id.cloudfront.net` .	

Monitor by using Amazon CloudWatch

Task	Description	Skills required
View the CloudWatch Dashboard.	1. Open the CloudWatc h console and choose **Dashboards**. 2. Select the dashboard that has the name **<appId>-<stageName>-dashboard**. 3. Review the dashboard . What resources are being monitored? What metrics are being recorded? This dashboard is made possible by the open-sour ce construct [cdk-monit oring-constructs]().	App developer
Enable alerts.	A CloudWatch Dashboard helps you to actively monitor your web app. To passively	App developer

Task	Description	Skills required

	monitor your web app, you can enable alerting.
	
	1. Navigate to `/infra/src/app/stateless/monitor-stack.ts` , which defines the monitor stack.
	2. Uncomment the following line, and replace `admin@example.com` with your email address.
	
	`onAlarmTopic.addSubscription(new EmailSubscription("admin@example.com"));`
	
	3. Add the following import information to the top of the file.
	
	`import { EmailSubscription } from "aws-cdk-lib/aws-sns-subscriptions";`
	
	4. Within `infra/`, run the following command.
	
	`cdk deploy "*/monitor" --exclusively.`
	
	5. To confirm your subscription to the SNS topic that is intitated when a monitoring alarm is initiated, choose	

Task	Description	Skills required
	the link in the email message.	

Understand the app infrastructure by using AWS CDK

Task	Description	Skills required
Create an architecture diagram.	Generate an architecture diagram of your web app by using cdk-dia. Visualizing the architecture helps improve understanding and communication among team members. It provides a clear overview of the system's components and their relationships. 1. Install Graphviz. 2. Within `infra/`, run the command `pnpm cdk-dia`. 3. View your `infra/diagram.png` .	App developer
Use cdk-nag to enforce best practices.	Use cdk-nag to help you maintain secure and compliant infrastructure by enforcing best practices, reducing the risk of security vulnerabilities and misconfigurations. 1. Explore cdk-nag's best practice enforcement through its rules section,	App developer

Task	Description	Skills required
	including checks from the AWS Solutions Library's Rules Pack. 2. To see how cdk-nag enforces rules, make a change in the code. For example, in `infra/src/app/stateful/data-stacks.ts`, change `storageEncrypted: true` to `storageEncrypted: false`. 3. Within `infra/`, run the command `cdk synth "*/data"`. During synthesis, you will encounter a build error that indicates a rule violation. `AwsSolutions-RDS2: The RDS instance or Aurora DB cluster does not have storage encryption enabled.` This error showcases how cdk-nag is a security mechanism for enforcing infrastructure best practices and preventing security misconfigurations. 4. If needed, you can also [suppress rules](https://github.com/cdklabs/cdk-nag) at different	

Task	Description	Skills required

	scopes. For example, to suppress AwsSolutions-RDS2, add the following code below the instantiation of `DbIamCluster` .
	
	`NagSuppressions.addResourceSuppressions(`
	`  cluster.node.findChild("Resource"),`
	`  [`
	`    {`
	`      id: "AwsSolutions-RDS2",`
	`      reason: "Customer requirement necessitates having unencrypted DB storage",`
	`    },`
	`  ],`
	`);`
	
	5. After suppression, run `cdk synth "*/data"` again. Your AWS CDK app should now synthesize successfully. You can find all suppressed rules in `infra/cdk.out/assembly-<appId>-<stageName>/AwsSolutions-<appId>-<stageName>-${stackId}-NagReport.csv` .	

Evaluate the database configuration and schema

Task	Description	Skills required
Acquire environment variables.	To obtain the required environment variables, use the following steps: 1. To find the `DB_BASTION_ID`, sign in to the console, and navigate to the EC2 console. Choose **Instances (running)**, and find the row that contains **<stageName>-ssm-db-bastion Name**. The instance ID starts with **i-**. 2. To find the `DB_ENDPOINT`, on the Amazon Relational Database Service (Amazon RDS) console, choose **DB Instances**, and select the regional cluster that has a DB identifier starting with **<appId>-<stageName>-data-**. Locate the writer instance endpoint, which ends with **rds.amazonaws.com**.	App developer
Establish port forwarding.	To establish port forwarding, use the following steps:	App developer

Task	Description	Skills required
	1. Install the AWS Systems Manager [Session Manager plugin](). 2. Start port forwarding by running `pnpm db:connect` within `core/` to establish a secure connection through the bastion host. 3. After you see the text `Waiting for connections...,` in your terminal, an SSH tunnel has been successfully established between your local machine and the Aurora server through the EC2 bastion host.	
Adjust the Systems Manager Session Manager timeout.	(Optional) If the default 20-minute session timeout is too short, you can increase it up to 60 minutes in the Systems Manager console by choosing **Session Manager**, **Preferences**, **Edit**, **Idle session timeout**.	App developer

Task	Description	Skills required
Visualize the database.	pgAdmin is a user-frie ndly open-source tool for managing PostgreSQ L databases. It simplifies database tasks, allowing you to efficiently create, manage, and optimize databases. This section guides you through [installing pgAdmin](#) and using its features for PostgreSQL database management. 1. In the Object Explorer, open the context (right-click) menu for **Servers**, and then choose **Register**, **Server**. 2. On the **General** tab, enter **<appId>-<stageName>** for the **Name** field. 3. To fetch the DB password, open the AWS Secrets Manager console, select the secret that has the description **Generated by the CDK for the stack: <appId>-<stageName >-data**, and choose the **Secret Value** card. Choose **Retrieve Secret Value**, and copy the **Secret value** with a key of **password**. 4. On the **Connection** tab, enter **0.0.0** for the **Host**	App developer

Task	Description	Skills required
	name/address field, and enter **<appId>_admin** for the **Username** field. For the **Password** field, use the secret that you fetched previously. Choose **yes** for the **Save password?** field. 5. Choose **Save**. 6. To view the tables, navigate to **<appId>-< stageName>**, **Databases** , **<appId>_db**, **Schemas**, **<appId>**, **Tables**. 7. Open the context (right-click) menu for the **item** table, and then select **View/Edit Data**, **All Rows**. 8. Explore the table.	

Debug with Node.js

Task	Description	Skills required
Debug the create item use case.	To debug the create item use case, follow these steps: 1. Open the `core/src/modules/item/create-item.use-case.ts` file, and insert the following code.	App developer

Task	Description	Skills required

	``
	import { fileURLTo
	Path } from "node:url
	";
	
	// existing create-it
	em.use-case.ts code
	 here
	
	if (process.argv[1]
	 === fileURLTo
	Path(import.meta.u
	rl)) {
	  createItemUseCase(
	{
	    description:
	 "Item 1's Descripti
	on",
	    name: "Item 1",
	  });
	}
	``
	2. The code added in the previous step ensures the `createItemUseCase` function will be called when this module is run directly. Set [breakpoints](#) on the lines within this code block where you want to initiate line-by-line debugging.
	
	1. Open the [VS Code JavaScript Debug Terminal](#), and then run `pnpm tsx core/src/modules/item/create-item.us`	

Task	Description	Skills required
	`e-case.ts` to run the code with line-by-line debugging. Alternatively, you can use `console.log` statements, but print statements can be inadequate when you're working with complex business logic. Line-by-line debugging gives you more context.	

Develop the frontend

Task	Description	Skills required
Set up the development server.	1. Navigate to `ui/`, and run `pnpm dev` to start the [Next.js](https://nextjs.org/) development server. 2. Access your web app locally at `http://localhost:3000`. The Next.js development server is set up with [Fast Refresh](https://nextjs.org/docs/architecture/fast-refresh) instantaneous feedback on edits made to your React components. 3. Experiment with customizing the app bar color. Open the `ui/src/components/theme/theme.tsx` file and locate the section that defines	App developer

Task	Description	Skills required
	the theme for the app bar. In the `colorSchemes.light.palette.primary` section, update the main value from `colors.lagoon` to `colors.carrot`. After making this change, save the file and observe the update in your browser. 4. Experiment by modifying text, components, and adding new pages.	

Tooling with Green Boost

Task	Description	Skills required
Set up monorepo and the pnpm package manager.	1. Review `pnpm-workspace.yaml` in the root of your GB repository, and notice how workspaces are defined. For more information about workspaces, see the pnpm documentation. 2. Review `ui/package.json`, and notice how it references the workspace in `core/` with the package name `"<appId>/core": "workspace:^",`.	App developer

Task	Description	Skills required
	3. Observe how TypeScript and ESLint configuration is centralized in utility packages defined within `packages/` . This configuration is then used by application packages such as `core/`, `infra/`, and `ui/`. This is helpful when your app scales and you define more application packages, which can reference the utility packages without duplicating configuration code.	

Task	Description	Skills required
Run pnpm scripts.	Run the following commands in the root of your repository: 1. Run `pnpm lint`. This command runs static code analysis with ESLint. 2. Run `pnpm typecheck .` This command runs the TypeScript compiler to check the types of your code. 3. Run `pnpm test`. This command runs Vitest to run unit tests. Notice how these commands are run in all workspaces. The commands are defined in each workspace's `package.json#scripts` field.	App developer

Task	Description	Skills required
Use ESLint for static code analysis.	To test the static code analysis capability of ESLint, do the following: 1. First, ensure the [VS Code ESLint extension]() (ID: `dbaeumer.vscode-eslint`) is installed. We recommend also installing [VS Code Error Lens]() (ID: `usernamehw.errorlens`) to see errors inline. 2. In your code, purposefully include a line of code that uses the `eval()` function, as shown in the following example. `const userInput = "import("fs").then((fs) => console.log(fs.readFileSync("/etc/passwd", { encoding: "utf8" })))"; eval(userInput);` **Important** This is for testing purposes only. Using `eval()` is considered potentially dangerous and should be avoided	App developer

Task	Description	Skills required
	because of security risks. 3. After you include the `eval()` line, open your code editor to confirm that ESLint indicated the code smell by using red squiggles. 4. Review ESLint plugins and configuration at `packages/eslint-config-{node,next}/.eslintrc.cjs` .	
Manage dependencies and vulnerabilities.	1. To identify any Common Vulnerabilities and Exposures (CVEs), run `pnpm audit` in the root of your repository. You should see **No known vulnerabilities found.** 2. Install an intentionally vulnerable package within `core/` by running `pnpm add minimist@0.2.3` , and then run `pnpm audit`. Notice the vulnerability being reported. 3. Uninstall the vulnerable package within `core/` by running `pnpm remove minimist`.	App developer

Task	Description	Skills required
Pre-commit hooks with Husky.	1. Make a couple small changes in TypeScript files throughout the repository. The changes can be as basic as adding comments. 2. Stage and commit these changes by using `git add -A` and then `git commit -m "test husky"`. The Husky pre-commit hook trigger, which is defined in `.husky/pre-commit`, runs the command `pnpm lint-staged`. 3. Observe how lint-staged runs commands specified in `*/.lintstagedrc.js` files throughout the repository on files that have been staged by Git. These tools are mechanisms to help prevent bad code from making its way into your application.	App developer

Tear down the infrastructure

Task	Description	Skills required
Remove the deployment from your account.	1. To tear down the infrastructure that you provisioned in the first epic, by run `pnpm destroy:local` in `infra/`. 2. Wait 15 minutes after `pnpm destroy:local` is complete, and then delete the retained Lambda@Edge function by searching for your app ID in the Lambda console. Lambda@Edge functions are replicated., which makes them difficult to delete. For more information about deleting Lambda@Edge functions, see the [CloudFront documentation]().	App developer

Troubleshooting

Issue	Solution
Unable to establish port forwarding	Ensure that your AWS credentials are properly configured and have the necessary permissions. Double-check that the bastion host ID (`DB_BASTION_ID`) and database endpoint

Issue	Solution
	(`DB_ENDPOINT`) environment variables are correctly set. If you still encounter issues, see the AWS documentation for [troubleshooting SSH connections and Session Manager]().
Website isn't loading on `localhost:3000`	Confirm that the terminal output indicates successful port forwarding, including the forwarding address. Ensure there are no conflicting processes using port 3000 on your local machine. Verify that the Green Boost application is properly configured and running on the expected port (3000). Check your web browser for any security extensions or settings that might block local connections.
Error messages during local deployment (`pnpm deploy:local`)	Review the error messages carefully to identify the cause of the issue. Verify that the necessary environment variables and configuration files are correctly set.

Related resources

- AWS CDK documentation
- Green Boost documentation
- Next.js documentation
- Node.js documentation
- React documentation

- [TypeScript documentation]

Run unit tests for a Node.js application from GitHub by using AWS CodeBuild

Thomas Scott and Jean-Baptiste Guillois, Amazon Web Services

Summary

This pattern provides sample source code and key unit test components for a Node.js game API. It also includes instructions for running these unit tests from a GitHub repository by using AWS CodeBuild, as part of your continuous integration and continuous delivery (CI/CD) workflow.

Unit testing is a software development process in which different parts of an application, called *units*, are individually and independently tested for correct operation. Tests validate the quality of the code and confirm that it functions as expected. Other developers can also easily gain familiarity with your code base by consulting the tests. Unit tests reduce future refactoring time, help engineers get up to speed on your code base more quickly, and provide confidence in the expected behavior.

Unit testing involves testing individual functions, including AWS Lambda functions. To create unit tests, you need a testing framework and a way of validating tests (assertions). The code examples in this pattern use the Mocha testing framework and the Chai assertion library.

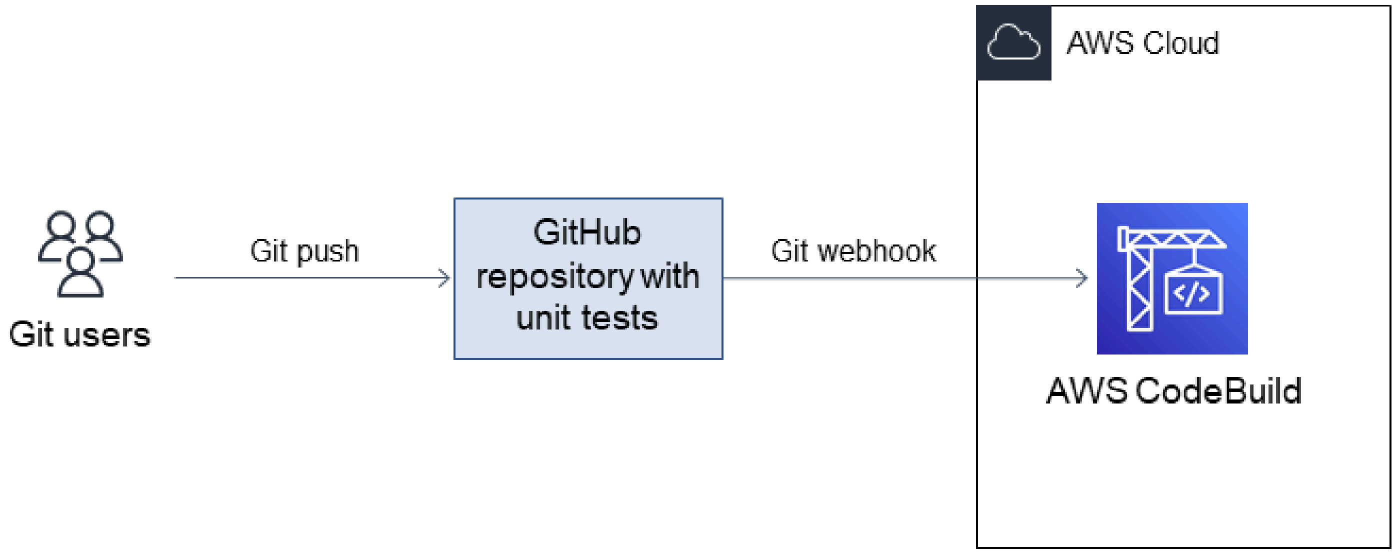
For more information about unit testing and examples of test components, see the Additional information section.

Prerequisites and limitations

- An active AWS account with correct CodeBuild permissions
- A GitHub account (see instructions for signing up)
- Git (see installation instructions)
- A code editor to make changes and push your code to GitHub

Architecture

This pattern implements the architecture that is shown in the following diagram.



Tools

Tools

- Git is a version control system that you can use for code development.
- AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. CodeBuild scales continuously and processes multiple builds concurrently, so your builds are not left waiting in a queue. You can get started quickly by using prepackaged build environments, or you can create custom build environments that use your own build tools. With CodeBuild, you are charged by the minute for the compute resources you use.

Code

The source code for this pattern is available on GitHub, in the Sample game unit test application repository. You can create your own GitHub repository from this sample (option 1) or use the sample repository directly (option 2) for this pattern. Follow the instructions for each option in the next section. The option you follow will depend on your use case.

Epics

Option 1 - Run unit tests on your personal GitHub repository with CodeBuild

Task	Description	Skills required
Create your own GitHub repository based on the sample project.	1. Log in to GitHub. 2. Create a new repository. For instructions, see the [GitHub documentation](#). 3. Clone and push the [sample repository](#) into the new repository in your account.	App developer, AWS administrator, AWS DevOps
Create a new CodeBuild project.	1. Sign in to the AWS Management Console and open the CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home. 2. Choose **Create build project**. 3. In the **Project configura tion** section, for **Project name**, type **aws-tests-sample-node-js**. 4. In the **Source** section, for **Source provider**, choose **GitHub**. 5. For **Repository**, choose **Repository in my GitHub account**, and then paste the URL to your newly created GitHub repository.	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required
	6. In the **Primary source webhook events** section, select **Rebuild every time a code change is pushed to this repository**. 7. For event type, choose **PUSH.** 8. In the **Environment** section, choose **Managed image**, Amazon Linux, and the latest image. 9. Leave the default settings for all other options, and then choose **Create build project**.	
Start the build.	On the **Review** page, choose **Start build** to run the build.	App developer, AWS administrator, AWS DevOps

Option 2 - Run unit tests on a public repository with CodeBuild

Task	Description	Skills required
Create a new CodeBuild build project.	1. Sign in to the AWS Management Console and open the CodeBuild console at https://console.aws.amazon.com/codesuite/codebuild/home. 2. Choose **Create build project**.	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required
	3. In the **Project configuration** section, for **Project name**, type **aws-tests-sample-node-js**. 4. In the **Source** section, for **Source provider**, choose **GitHub**. 5. For **Repository**, choose **Public repository**, and then paste the URL: https://github.com/aws-samples/node-js-tests-sample. 6. In the **Environment** section, choose **Managed image**, Amazon Linux, and the latest image. 7. Leave the default settings for all other options, and then choose **Create build project.**	
Start the build.	On the **Review** page, choose **Start build** to run the build.	App developer, AWS administrator, AWS DevOps

Analyze the unit tests

Task	Description	Skills required
View test results.	In the CodeBuild console, review the unit test results from the CodeBuild job. They should match the results	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required
	shown in the Additional information section. These results validate the GitHub repository integration with CodeBuild.	
Apply a webhook.	You can now apply a webhook, so you can automatically start a build whenever you push code changes to the main branch of your repository. For instructions, see the CodeBuild documentation.	App developer, AWS administrator, AWS DevOps

Related resources

- Sample game unit test application (GitHub repository with sample code)
- AWS CodeBuild documentation
- GitHub webhook events (CodeBuild documentation)
- Creating a new repository (GitHub documentation)

Additional information

Unit test results

In the CodeBuild console, you should see the following test results after the project builds successfully.

```
> node-game-unit-tests@1.0.0 test /codebuild/output/src813991724/src/github.com/aws-samples/node-js-tests-sample
> mocha



  Game Function Group
    Initialize Game
      ✓ Check that the game ID is between 0 and 1000
      ✓ Check that the play date is today
    Start Game
      ✓ Check that the game has two competitors
      ✓ Check that the two competitors have different names
    End Game
      ✓ Check that there are no more competitors after the game ends
    Game ID Stubs and Mocks
      ✓ Check that an out of range game ID is returned with Stub
      ✓ Using a mock check that the function was called and returns 9


  7 passing (9ms)
```

Example unit test components

This section describes the four types of test components that are used in unit testing: assertions, spies, stubs, and mocks. It includes a brief explanation and code example of each component.

Assertions

An assertion is used to verify an expected result. This is an important test component because it validates the expected response from a given function. The following sample assertion validates that the returned ID is between 0 and 1000 when initializing a new game.

```
const { expect }  = require('chai');
const { Game } = require('../src/index');

describe('Game Function Group', () => {
 it('Check that the Game ID is between 0 and 1000', function() {
      const game = new Game();
      expect(game.id).is.above(0).but.below(1000)
 });
});
```

Spies

A spy is used to observe what is happening when a function is running. For example, you might want to validate that the function has been called correctly. The following example shows that start and stop methods are called on a **Game** class object.

```
const { expect }  = require('chai');
const { spy } = require('sinon');
```

```
const { Game } = require('../src/index');

describe('Game Function Group', () => {
   it('should verify that the correct function is called', () => {
      const spyStart = spy(Game.prototype, "start");
      const spyStop = spy(Game.prototype, "stop");

      const game = new Game();
      game.start();
      game.stop();

      expect(spyStart.called).to.be.true
      expect(spyStop.called).to.be.true
    });
});
```

Stubs

A stub is used to override a function’s default response. This is especially useful when the function makes an external request, because you want to avoid making external requests from unit tests. (External requests are better suited for integration tests, which can physically test requests between different components.) In the following example, a stub forces a return ID from the **getId** function.

```
const { expect }  = require('chai');
const {.stub } = require('sinon');

const { Game } = require('../src/index');

describe('Game Function Group', () =>  {
   it('Check that the Game ID is between 0 and 1000', function() {
      let generateIdStub = stub(Game.prototype, 'getId').returns(999999);

      const game = new Game();

      expect(game.getId).is.equal(999999);

      generateIdStub.restore();
    });
});
```

Mocks

A mock is a fake method that has a pre-programmed behavior for testing different scenarios. A mock can be considered an extended form of a stub and can carry out multiple tasks simultaneously. In the following example, a mock is used to validate three scenarios:

- Function is called
- Function is called with arguments
- Function returns the integer 9

```
const { expect }  = require('chai');
const {.mock } = require('sinon');

const { Game } = require('../src/index');

describe('Game Function Group', () =>  {
   it('Check that the Game ID is between 0 and 1000', function() {
      let mock = mock(Game.prototype).expects('getId').withArgs().returns(9);

      const game = new Game();
      const id = get.getId();

      mock.verify();
      expect(id).is.equal(9);
    });
});
```

Structure a Python project in hexagonal architecture using AWS Lambda

Furkan Oruc, Dominik Goby, Darius Kunce, and Michal Ploski, Amazon Web Services

Summary

This pattern shows how to structure a Python project in hexagonal architecture by using AWS Lambda. The pattern uses the AWS Cloud Development Kit (AWS CDK) as the infrastructure as code (IaC) tool, Amazon API Gateway as the REST API, and Amazon DynamoDB as the persistence layer. Hexagonal architecture follows domain-driven design principles. In hexagonal architecture, software consists of three components: domain, ports, and adapters. For detailed information about hexagonal architectures and their benefits, see the guide Building hexagonal architectures on AWS.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Experience in Python
- Familiarity with AWS Lambda, AWS CDK, Amazon API Gateway, and DynamoDB
- A GitHub account (see instructions for signing up)
- Git (see installation instructions)
- A code editor for making changes and pushing your code to GitHub (for example, Visual Studio Code or JetBrains PyCharm)
- Docker installed, and the Docker daemon up and running

Product versions

- Git version 2.24.3 or later
- Python version 3.7 or later
- AWS CDK v2
- Poetry version 1.1.13 or later
- AWS Lambda Powertools for Python version 1.25.6 or later
- pytest version 7.1.1 or later

- Moto version 3.1.9 or later
- pydantic version 1.9.0 or later
- Boto3 version 1.22.4 or later
- mypy-boto3-dynamodb version 1.24.0 or later

Architecture

Target technology stack

The target technology stack consists of a Python service that uses API Gateway, Lambda, and DynamoDB. The service uses a DynamoDB adapter to persist data. It provides a function that uses Lambda as the entry point. The service uses Amazon API Gateway to expose a REST API. The API uses AWS Identity and Access Management (IAM) for the authentication of clients.

Target architecture

To illustrate the implementation, this pattern deploys a serverless target architecture. Clients can send requests to an API Gateway endpoint. API Gateway forwards the request to the target Lambda function that implements the hexagonal architecture pattern. The Lambda function performs create, read, update, and delete (CRUD) operations on a DynamoDB table.

> **Important**
>
> This pattern was tested in a PoC environment. You must conduct a security review to identify the threat model and create a secure code base before you deploy any architect ure to a production environment.

The API supports five operations on a product entity:

- `GET /products` returns all products.
- `POST /products` creates a new product.
- `GET /products/{id}` returns a specific product.
- `PUT /products/{id}` updates a specific product.
- `DELETE /products/{id}` deletes a specific product.

You can use the following folder structure to organize your project to follow the hexagonal architecture pattern:

```
app/  # application code
|--- adapters/  # implementation of the ports defined in the domain
     |--- tests/  # adapter unit tests
|--- entrypoints/  # primary adapters, entry points
     |--- api/  # api entry point
          |--- model/  # api model
          |--- tests/  # end to end api tests
|--- domain/  # domain to implement business logic using hexagonal architecture
     |--- command_handlers/  # handlers used to execute commands on the domain
     |--- commands/  # commands on the domain
     |--- events/  # events triggered via the domain
     |--- exceptions/  # exceptions defined on the domain
     |--- model/  # domain model
     |--- ports/  # abstractions used for external communication
     |--- tests/  # domain tests
|--- libraries/  # List of 3rd party libraries used by the Lambda function
infra/  # infrastructure code
simple-crud-app.py  # AWS CDK v2 app
```

Tools

AWS services

- [Amazon API Gateway](https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html) is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.
- [Amazon DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html) is a fully managed, serverless, key-value NoSQL database that is designed to run high-performance applications at any scale.
- [AWS Lambda](https://docs.aws.amazon.com/lambda/latest/dg/welcome.html) is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can launch

Lambda functions from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

Tools

- Git is used as the version control system for code development in this pattern.
- Python is used as the programming language for this pattern. Python provides high-level data structures and an approach to object-oriented programming. AWS Lambda provides a built-in Python runtime that simplifies the operation of Python services.
- Visual Studio Code is used as the IDE for development and testing for this pattern. You can use any IDE that supports Python development (for example, PyCharm).
- AWS Cloud Development Kit (AWS CDK) is an open-source software development framework that lets you define your cloud application resources by using familiar programming languages. This pattern uses the CDK to write and deploy cloud infrastructure as code.
- Poetry is used to manage dependencies in the pattern.
- Docker is used by the AWS CDK to build the Lambda package and layer.

Code

The code for this pattern is available in the GitHub Lambda hexagonal architecture sample repository.

Best practices

To use this pattern in a production environment, follow these best practices:

- Use customer managed keys in AWS Key Management Service (AWS KMS) to encrypt Amazon CloudWatch log groups and Amazon DynamoDB tables.
- Configure AWS WAF for Amazon API Gateway to allow access only from your organization's network.
- Consider other options for API Gateway authorization if IAM doesn’t meet your needs. For example, you can use Amazon Cognito user pools or API Gateway Lambda authorizers.
- Use DynamoDB backups.
- Configure Lambda functions with a virtual private cloud (VPC) deployment to keep network traffic inside the cloud.

- Update the allowed origin configuration for cross-origin resource sharing (CORS) preflight to restrict access to the requesting origin domain only.
- Use cdk-nag to check the AWS CDK code for security best practices.
- Consider using code scanning tools to find common security issues in the code. For example, Bandit is a tool that's designed to find common security issues in Python code. Pip-audit scans Python environments for packages that have known vulnerabilities.

This pattern uses AWS X-Ray to trace requests through the application's entry point, domain, and adapters. AWS X-Ray helps developers identify bottlenecks and determine high latencies to improve application performance.

Epics

Initialize the project

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create your own repository.</td><td>1. Log in to GitHub.
2. Create a new repository. For instructions, see the GitHub documentation.
3. Clone and push the sample repository for this pattern into the new repository in your account.</td><td>App developer</td></tr>
<tr><td>Install dependencies.</td><td>1. Install Poetry.
<pre>pip install poetry</pre>2. Install packages from the root directory. The following command installs the application and AWS CDK packages. It also installs development packages that are required</td><td>App developer</td></tr>
</table>

Task	Description	Skills required

	for running unit tests. All installed packages are placed in a new virtual environment.
	
	`poetry install`
	
	3. To see a graphical representation of the installed packages, run the following command.
	
	`poetry show --tree`
	
	4. Update all dependencies.
	
	`poetry update`
	
	5. Open a new shell within the newly created virtual environment. It contains all installed dependencies.
	
	`poetry shell`	

Task	Description	Skills required
Configure your IDE.	We recommend Visual Studio Code, but you can use any IDE of your choice that supports Python. The following steps are for Visual Studio Code. 1. Update the `.vscode/settings` file. `{ "python.testing.pytestArgs": [ "app/adapters/tests", "app/entrypoints/api/tests", "app/domain/tests" ], "python.testing.unittestEnabled": false, "python.testing.pytestEnabled": true, "python.envFile": "${workspaceFolder}/.env", }` 2. Create an `.env` file in the root directory of the project. This ensures that the root directory of the project is included in the `PYTHONPATH` so that `pytest` can find it	App developer

Task	Description	Skills required
	and properly discover all packages. `PYTHONPATH=.`	
Run unit tests, option 1: Use Visual Studio Code.	1. Choose the Python interpreter of the virtual environment that's managed by Poetry. 2. Run tests from Test Explorer.	App developer

Run unit tests, option 2: Use shell commands.	1. Start a new shell within the virtual environment.
	`poetry shell`
	2. Run the `pytest` command from the root directory.
	`python -m pytest`
	Alternatively you can run the command directly from Poetry.
	`poetry run python -m pytest`
		App developer

Deploy and test the application

Task	Description	Skills required
Request temporary credentials.	To have AWS credentials on the shell when you run `cdk`	App developer, AWS DevOps

Task	Description	Skills required
	`deploy`, create temporary credentials by using AWS IAM Identity Center (successor to AWS Single Sign-On). For instructions, see the blog post [How to retrieve short-term credentials for CLI use with AWS IAM Identity Center]().	

Deploy the application.	1. Install the AWS CDK v2.
	`npm install -g aws-cdk`
	For more information, see the [AWS CDK documentation]().
	2. Bootstrap the AWS CDK into your account and Region.
	`cdk bootstrap aws://12345678900/us-east-1 --profile aws-profile-name`
	3. Deploy the application as an AWS CloudFormation stack by using an AWS profile.
	`cdk deploy --profile aws-profile-name`
		App developer, AWS DevOps

Task	Description	Skills required
Test the API, option 1: Use the console.	Use the API Gateway console to test the API. For more information about API operations and request/r esponse messages, see the API usage section of the readme file in the GitHub repository.	App developer, AWS DevOps
Test the API, option 2: Use Postman.	If you want to use a tool such as Postman: 1. Install Postman as a standalone application or browser extension. 2. Copy the endpoint URL for the API Gateway. It will be in the following format. `https://{api-id}.execute-api.{region}.amazonaws.com/{stage}/{path}` 3. Configure the AWS signature in the authoriza tion tab. For instructions, see the AWS re:Post article on activating IAM authentic ation for API Gateway REST APIs. 4. Use Postman to send requests to your API endpoint.	App developer, AWS DevOps

Develop the service

Task	Description	Skills required

Write unit tests for the business domain.	1. Create a Python file in the `app/domain/tests` folder by using the `test_` file name prefix.
	2. Create a new test method to test the new business logic by using the following example.
	<pre>def test_create_product_should_store_in_repository():
	    # Arrange
	    command = create_product_command.CreateProductCommand(
	        name="Test Product",
	        description="Test Description",
	    )
	    # Act
	    create_product_command_handler.handle_create_product_command(
	        command=command, unit_of_work=mock_unit_of_work
	    )
	    # Assert</pre>
		App developer

Task	Description	Skills required
	3. Create a command class in the `app/domain/commands` folder. 4. If the functionality is new, create a stub for the command handler in the `app/domain/command_handlers` folder. 5. Run the unit test to see it fail, because there is still no business logic. `python -m pytest`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Implement commands and command handlers.</td><td>

Implement business logic in the newly created command handler file.
For every dependency that interacts with external systems, declare an abstract class in the <code>app/domain/ports</code> folder.

<pre><code>class ProductsRepository(ABC):
    @abstractmethod
    def add(self, product: product.Product) -> None:
        ...

class UnitOfWork(ABC):
    products: ProductsRepository

    @abstractmethod
    def commit(self) -> None:
        ...

    @abstractmethod
    def __enter__(self) -> typing.Any:
        ...

    @abstractmethod
    def __exit__(self, *args) -> None:
        ...
</code></pre>
</td><td>App developer</td></tr>
</table>

Task	Description	Skills required

	3. Update the command handler signature to accept the newly declared dependencies by using the abstract port class as type annotation.
	
	`def handle_create_product_command(`
	`    command: create_product_command.CreateProductCommand,`
	`    unit_of_work: unit_of_work.UnitOfWork,`
	`) -> str:`
	`    ...`
	
	4. Update the unit test to simulate the behavior of all declared dependencies for the command handler.
	
	`    # Arrange`
	`    mock_unit_of_work = unittest.mock.create_autospec(`
	`        spec=unit_of_work.UnitOfWork, instance=True`
	`    )`
	`    mock_unit_of_work.products = unittest.mock.create_autospec(`
	`        spec=unit_of_work.ProductsRepository, instance=True`	

Task	Description	Skills required

	``
	    )
	``
	5. Update the assertion logic in the test to check for the expected dependency invocations.
	``
	  # Assert
	    mock_unit_of_work.commit.assert_called_once()
	    product = mock_unit_of_work.products.add.call_args.args[0]
	
	    assertpy.assert_that(product.name).is_equal_to("Test Product")
	    assertpy.assert_that(product.description).is_equal_to("Test Description")
	``
	6. Run the unit test to see it succeed.
	``
	python -m pytest
	``	

Task	Description	Skills required

Write integration tests for secondary adapters.	1. Create a test file in the `app/adapters/tests` folder by using `test_` as a file name prefix.
	2. Use the Moto library to mock AWS services.
	`@pytest.fixture def mock_dynamodb(): with moto.mock _dynamodb(): yield boto3.res ource("dynamodb", region_name="eu-ce ntral-1")`
	3. Create a new test method for an integration test of the adapter.
	`def test_add_ and_commit_should_ store_product(mock _dynamodb): # Arrange unit_of_work = dynamodb_unit_of_w ork.DynamoDBUnitOf Work( table_nam e=TEST_TABLE_NAME, dynamodb_client=mo ck_dynamodb.meta.c lient ) current_time = datetime.datetime. now(datetime.timez`
		App developer

Task	Description	Skills required

	<pre>one.utc).isoformat
	()
	
	    new_product_id =
	 str(uuid.uuid4())
	    new_product =
	 product.Product(
	        id=new_pr
	oduct_id,
	        name="test-
	name",
	        descripti
	on="test-descripti
	on",
	        createDat
	e=current_time,
	        lastUpdat
	eDate=current_time,
	    )
	
	    # Act
	    with unit_of_w
	ork:
	        unit_of_w
	ork.products.add(n
	ew_product)
	        unit_of_w
	ork.commit()
	
	    # Assert</pre>4. Create an adapter class in the `app/adapters` folder. Use the abstract class from the ports folder as a base class.
	5. Run the unit test to see it fail, because there is still no logic.	

Task	Description	Skills required
	`python -m pytest`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Implement secondary adapters.</td><td>

1. Implement logic in the newly created adapter file.
2. Update test assertions.

```
# Assert
    with unit_of_w
ork_readonly:
        product_f
rom_db = unit_of_w
ork_readonly.produ
cts.get(new_produc
t_id)

    assertpy.
assert_that(produc
t_from_db).is_not_
none()
    assertpy.
assert_that(produc
t_from_db.dict()).
is_equal_to(
        {
            "id":
 new_product_id,
            "name":
 "test-name",
            "descript
ion": "test-des
cription",
            "createDa
te": current_time,
            "lastUpda
teDate": current_t
ime,
        }
    )
```

3. Run the unit test to see it succeed.

</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
	`python -m pytest`	

Task	Description	Skills required

Write end-to-end tests.	1. Create a test file in the `app/entrypoints/api/tests` folder by using `test_` as a file name prefix.
	2. Create a Lambda context fixture that will be used by the test to call Lambda.
	`@pytest.fixture`
	`def lambda_context():`
	`    @dataclass`
	`    class LambdaContext:`
	`        function_name: str = "test"`
	`        memory_limit_in_mb: int = 128`
	`        invoked_function_arn: str = "arn:aws:lambda:eu-west-1:809313241:function:test"`
	`        aws_request_id: str = "52fdfc07-2182-154f-163f-5f0f9a621d72"`
	
	`    return LambdaContext()`
	3. Create a test method for the API invocation.
	`def test_create_product(lambda_context):`
		App developer

Task	Description	Skills required

	``
	    # Arrange
	    name = "TestName"
	    description =
	 "Test description"
	    request =
	 api_model.CreatePr
	oductRequest(name=
	name, descripti
	on=description)
	
	    minimal_event =
	 api_gateway_proxy_
	event.APIGatewayPr
	oxyEvent(
	        {
	            "path": "/
	products",
	            "httpMeth
	od": "POST",
	            "requestC
	ontext": { #
	 correlation ID
	
	  "requestId":
	 "c6af9ac6-7b61-11e
	6-9a41-93e8deadbee
	f"
	            },
	            "body":
	 json.dumps(request
	.dict()),
	        }
	    )
	
	    create_pr
	oduct_func_mock =
	 unittest.mock.crea
	te_autospec(
	        spec=crea
	te_product_command
	_handler.handle_cr
	``	

Task	Description	Skills required

	``
	eate_product_comma
	nd
	    )
	    handler.c
	reate_product_comm
	and_handler.handle
	_create_product_co
	mmand = (
	        create_pr
	oduct_func_mock
	    )
	
	    # Act
	    handler.h
	andler(minimal_eve
	nt, lambda_context)
	``
	4. Run the unit test to see it fail, because there is still no logic.
	``
	python -m pytest
	``	

Task	Description	Skills required

Implement primary adapters.	1. Create a function for API business logic and declare it as an API resource.
	`@tracer.capture_method`
	`@app.post("/products")`
	`@utils.parse_event(model=api_model.CreateProductRequest, app_context=app)`
	`def create_product(`
	`    request: api_model.CreateProductRequest,`
	`) -> api_model.CreateProductResponse:`
	`    """Creates a product."""`
	`...`
	Note
	All decorators you see are features of the AWS Lambda Powertools for Python library. For details, see the AWS Lambda Powertools for Python website.
	2. Implement the API logic.
		App developer

Task	Description	Skills required

	``
	id=create_product_
	command_handler.ha
	ndle_create_produc
	t_command(
	        command=c
	reate_product_comm
	and.CreateProductC
	ommand(
	            name=requ
	est.name,
	            descripti
	on=request.descrip
	tion,
	        ),
	        unit_of_w
	ork=unit_of_work,
	    )
	    response =
	 api_model.CreatePr
	oductResponse(id=i
	d)
	    return response.
	dict()
	``
	3. Run the unit test to see it succeed.
	``
	python -m pytest
	``	

Related resources

APG guide

- Building hexagonal architectures on AWS

AWS References

- AWS Lambda documentation

- AWS CDK documentation
  - Your first AWS CDK app
- API Gateway documentation
  - Control access to an API with IAM permissions
  - Use the API Gateway console to test a REST API method
- Amazon DynamoDB documentation

Tools

- git-scm.com website
- Installing Git
- Creating a new GitHub repository
- Python website
- AWS Lambda Powertools for Python
- Postman website
- Python mock object library
- Poetry website

IDEs

- Visual Studio Code website
- PyCharm website

More patterns

- Access AWS services from an ASP.NET Core app using Amazon Cognito identity pools
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer
- Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer
- Associate an AWS CodeCommit repository in one AWS account with Amazon SageMaker AI Studio Classic in another account
- Automate deletion of AWS CloudFormation stacks and associated resources
- Restrict access based on IP address or geolocation by using AWS WAF
- Build a loosely coupled architecture with microservices using DevOps practices and AWS Cloud9
- Build a serverless React Native mobile app by using AWS Amplify
- Build and test iOS apps with AWS CodeCommit, AWS CodePipeline, and AWS Device Farm
- Configure logging for .NET applications in Amazon CloudWatch Logs by using NLog
- Consolidate Amazon S3 presigned URL generation and object downloads by using an endpoint associated with static IP addresses
- Create a pipeline and AMI using CodePipeline and HashiCorp Packer
- Deploy artifact updates to on-premises Amazon EC2 instances using AWS CodePipeline
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS
- Deploy a gRPC-based application on an Amazon EKS cluster and access it with an Application Load Balancer
- Deploy a ChatOps solution to manage SAST scan results by using Amazon Q Developer in chat applications custom actions and AWS CloudFormation
- Deploy CloudWatch Synthetics canaries by using Terraform
- Deploy Java microservices on Amazon ECS using AWS Fargate
- Deploy a RAG use case on AWS by using Terraform and Amazon Bedrock
- Deploy resources in an AWS Wavelength Zone by using Terraform
- Implement path-based API versioning by using custom domains in Amazon API Gateway
- Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS
- Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk

- Migrate an on-premises Go web application to AWS Elastic Beanstalk by using the binary method
- Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk
- Migrate on-premises Java applications to AWS using AWS App2Container
- Migrate OpenText TeamSite workloads to the AWS Cloud
- Migrate Windows SSL certificates to an Application Load Balancer using ACM
- Modernize ASP.NET Web Forms applications on AWS
- Monitor application activity by using CloudWatch Logs Insights
- Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance
- Send custom attributes to Amazon Cognito and inject them into tokens
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront
- Set up a highly available PeopleSoft architecture on AWS
- Streamline Amazon Lex bot development and deployment by using an automated workflow
- Troubleshoot states in AWS Step Functions by using Amazon Bedrock
- Use Network Firewall to capture the DNS domain names from the Server Name Indication for outbound traffic
- Visualize AI/ML model results using Flask and AWS Elastic Beanstalk

IoT

Topics

- Extract and query AWS IoT SiteWise metadata attributes in a data lake
- Configure logging and monitoring for security events in an AWS IoT environment
- Set up and troubleshoot AWS IoT Greengrass with client devices
- More patterns

Extract and query AWS IoT SiteWise metadata attributes in a data lake

Ambarish Dongaonkar, Amazon Web Services

Summary

AWS IoT SiteWise uses asset models and hierarchies to represent your industrial equipment, processes, and facilities. Each model or asset can have multiple attributes that are specific to your environment. Example metadata attributes include the site or physical location of the asset, plant details, and equipment identifiers. These attribute values complement asset measurement data to maximize the business value. Machine learning (ML) can provide additional insights into this metadata and streamline engineering tasks.

However, metadata attributes can't be queried directly from the AWS IoT SiteWise service. To make the attributes queryable, you must extract and ingest them into a data lake. This pattern uses a Python script to extract the attributes for all AWS IoT SiteWise assets and ingest them into a data lake in an Amazon Simple Storage Service (Amazon S3) bucket. When you have completed this process, you can use SQL queries in Amazon Athena to access the AWS IoT SiteWise metadata attributes and other datasets, such as measurement datasets. The metadata attribute information is also useful when working with AWS IoT SiteWise monitors or dashboards. You can also build an Amazon QuickSight dashboard by using the extracted attributes in the Amazon S3 bucket.

The pattern has reference code, and you can you can implement the code by using the best compute services for your use case, such as AWS Lambda or AWS Glue.

Prerequisites and limitations

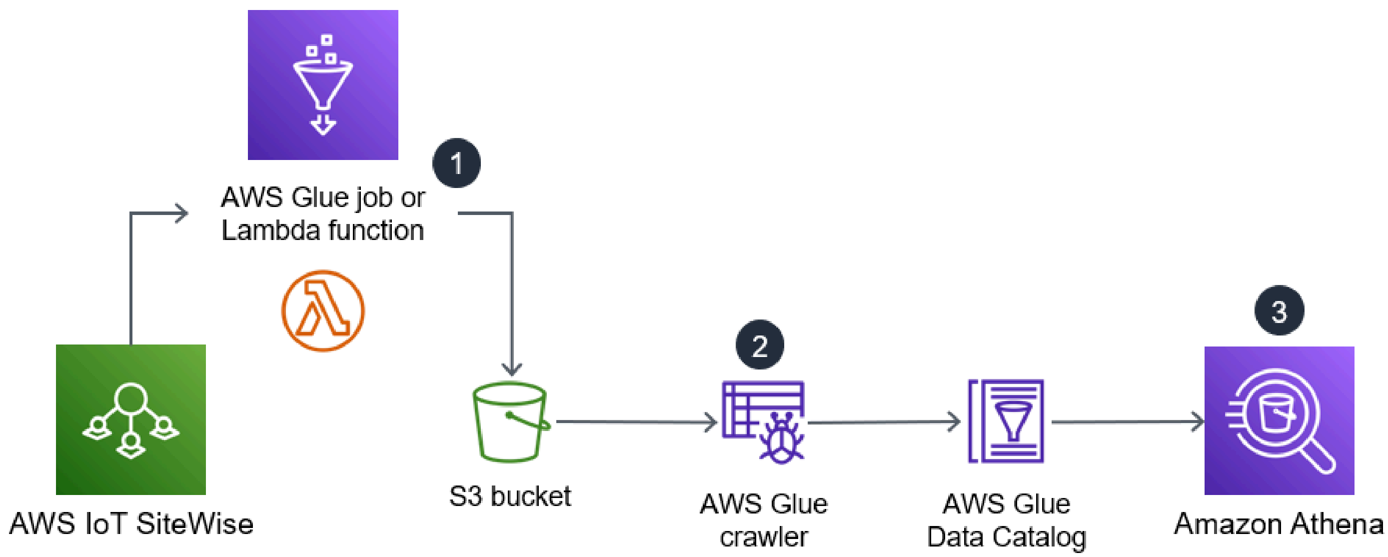
Prerequisites

- An active AWS account.
- Permissions to set up AWS Lambda functions or AWS Glue jobs.
- An Amazon S3 bucket.
- The asset models and hierarchies are set up in AWS IoT SiteWise. For more information, see Creating asset models in the AWS IoT SiteWise documentation.

Architecture

You can use a Lambda function or an AWS Glue job to complete this process. We recommend using Lambda if you have less than 100 models and each model has an average of 15 or fewer attributes. For all other use cases, we recommend using AWS Glue.

The solution architecture and workflow are shown in the following diagram.



1. The scheduled AWS Glue job or Lambda function runs. It extracts the asset metadata attributes from AWS IoT SiteWise and ingests them into an Amazon S3 bucket.
2. An AWS Glue crawler crawls the extracted data in the Amazon S3 bucket and creates tables in an AWS Glue Data Catalog.
3. Using standard SQL, Amazon Athena queries the tables in the AWS Glue Data Catalog.

Automation and scale

You can schedule the Lambda function or AWS Glue job to run daily or weekly, according to the update frequency of your AWS IoT SiteWise asset configurations.

There is no limit to the number of AWS IoT SiteWise assets that the sample code can process, but a large number of assets can increase the amount of time required to complete the process.

Tools

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon S3 by using standard SQL.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS IoT SiteWise helps you collect, model, analyze, and visualize data from industrial equipment at scale.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services.

Epics

Set up the job or function

Task	Description	Skills required
Configure permissions in IAM.	In the IAM console, grant permissions to the IAM role assumed by the Lambda function or AWS Glue job to do the following: • Read from the AWS IoT SiteWise service • Write to the Amazon S3 bucket	General AWS

Task	Description	Skills required
	For more information, see Creating a role for an AWS service in the IAM documenta tion.	
Create the Lambda function or AWS Glue job.	If you are using Lambda, create a new Lambda function. For **Runtime**, choose **Python**. For more information, see Building Lambda functions with Python in the Lambda documentation. If you are using AWS Glue, create a new Python shell job in the AWS Glue console. For more information, see Adding Python shell jobs in the AWS Glue documentation.	General AWS
Update the Lambda function or AWS Glue job.	Modify the new Lambda function or AWS Glue job, and enter the code sample in the Additional information section. Modify the code as needed for your use case. For more information, see Edit code using the console editor in the Lambda documenta tion and see Working with scripts in theAWS Glue documentation.	General AWS

Run the job or function

Task	Description	Skills required
Run the Lambda function or AWS Glue job.	Run the Lambda function or AWS Glue job. For more information, see Invoke the Lambda function in the Lambda documentation or see Starting jobs using triggers in the AWS Glue documentation. This extracts the metadata attributes for the assets and models in the AWS IoT SiteWise hierarchy and stores them in the specified Amazon S3 bucket.	General AWS
Set up an AWS Glue crawler.	Set up an AWS Glue crawler with the necessary format classifier for a CSV-forma tted file. Use the Amazon S3 bucket and prefix details used in the Lambda function or AWS Glue job. For more information, see Defining crawlers in the AWS Glue documentation.	General AWS
Run the AWS Glue crawler.	Run the crawler to process the data file created by the Lambda function or AWS Glue job. The crawler creates a table in the specified AWS Glue Data Catalog. For more information, see or Starting	General AWS

Task	Description	Skills required
	crawlers using triggers in the AWS Glue documentation.	
Query the metadata attribute s.	Using Amazon Athena, use standard SQL to query the AWS Glue Data Catalog as needed for your use case. You can join the metadata attribute table with other databases and tables. For more information, see Getting Started in the Amazon Athena documentation.	General AWS

Related resources

- Amazon Athena documentation
- AWS Glue documentation
- AWS IoT SiteWise API reference
- AWS IoT SiteWise user guide
  - Getting started
  - Modeling industrial assets
  - Defining relationships between asset models (hierarchies)
  - Associating and disassociating assets
  - Creating the AWS IoT SiteWise demo
- IOTSiteWise (SDK for Python documentation)
- Lambda documentation

Additional information

Code

The sample code provided is for reference, and you can customize this code as needed for your use case.

```
# Following code can be used in an AWS Lambda function or in an AWS Glue Python shell
 job.
# IAM roles used for this job need read access to the AWS IoT SiteWise service and
 write access to the S3 bucket.
sw_client = boto3.client('iotsitewise')
s3_client = boto3.client('s3')
output = io.StringIO()

attribute_list=[]
bucket = '{3_bucket name}'
prefix = '{s3_bucket prefix}'
output.write("model_id,model_name,asset_id,asset_name,attribuet_id,attribute_name,attribute_val
\n")

m_resp = sw_client.list_asset_models()
for m_rec in m_resp['assetModelSummaries']:
     model_id = m_rec['id']
     model_name = m_rec['name']

     attribute_list.clear()
     dam_response = sw_client.describe_asset_model(assetModelId=model_id)
     for rec in dam_response['assetModelProperties']:
         if 'attribute' in rec['type']:
            attribute_list.append(rec['name'])

     response = sw_client.list_assets(assetModelId=model_id, filter='ALL')
     for asset in response['assetSummaries']:
         asset_id = asset['id']
         asset_name = asset['name']
         resp = sw_client.describe_asset(assetId=asset_id)
         for rec in resp['assetProperties']:
            if rec['name'] in attribute_list:
                p_resp = sw_client.get_asset_property_value(assetId=asset_id,
 propertyId=rec['id'])
                if 'propertyValue' in p_resp:
                    if p_resp['propertyValue']['value']:
                        if 'stringValue' in p_resp['propertyValue']['value']:
                             output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
```

```
 str(p_resp['propertyValue']['value']['stringValue']) + "\n")

                       if 'doubleValue' in p_resp['propertyValue']['value']:
                            output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
 str(p_resp['propertyValue']['value']['doubleValue']) + "\n")
                       if 'integerValue' in p_resp['propertyValue']['value']:
                            output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
 str(p_resp['propertyValue']['value']['integerValue']) + "\n")
                        if 'booleanValue' in p_resp['propertyValue']['value']:
                            output.write(model_id + "," + model_name + ","
 + asset_id + "," + asset_name + "," + rec['id'] + "," + rec['name'] + "," +
 str(p_resp['propertyValue']['value']['booleanValue']) + "\n")

output.seek(0)
s3_client.put_object(Bucket=bucket, Key= prefix + '/data.csv', Body=output.getvalue())
output.close()
```

Configure logging and monitoring for security events in an AWS IoT environment

Prateek Prakash, Amazon Web Services

Summary

Ensuring that your Internet of Things (IoT) environments are secure is an important priority, particularly because organizations are connecting billions of devices to their IT environments. This pattern provides a reference architecture that you can use to implement logging and monitoring for security events across your IoT environment on the AWS Cloud. Typically, an IoT environment on the AWS Cloud has the following three layers:

- IoT devices that generate relevant telemetry data.
- AWS IoT services (for example, AWS IoT Core, AWS IoT Device Management, or AWS IoT Device Defender) that connect your IoT devices to other devices and AWS services.
- Backend AWS services that help process telemetry data and provide useful insights for your different business use cases.

The best practices provided by the AWS IoT Lens - AWS Well-Architected Framework whitepaper can help you review and improve your cloud-based architecture and better understand the business impact of your design decisions. An important recommendation is that you analyze application logs and metrics on your devices and in the AWS Cloud. You can achieve this by leveraging different approaches and techniques (for example, threat modeling) to identify metrics and events that must be monitored to detect potential security issues.

This pattern describes how to use AWS IoT and security services to design and implement a security logging and monitoring reference architecture for an IoT environment on the AWS Cloud. This architecture builds on existing AWS security best practices and applies them to your IoT environment.

Prerequisites and limitations

Prerequisites

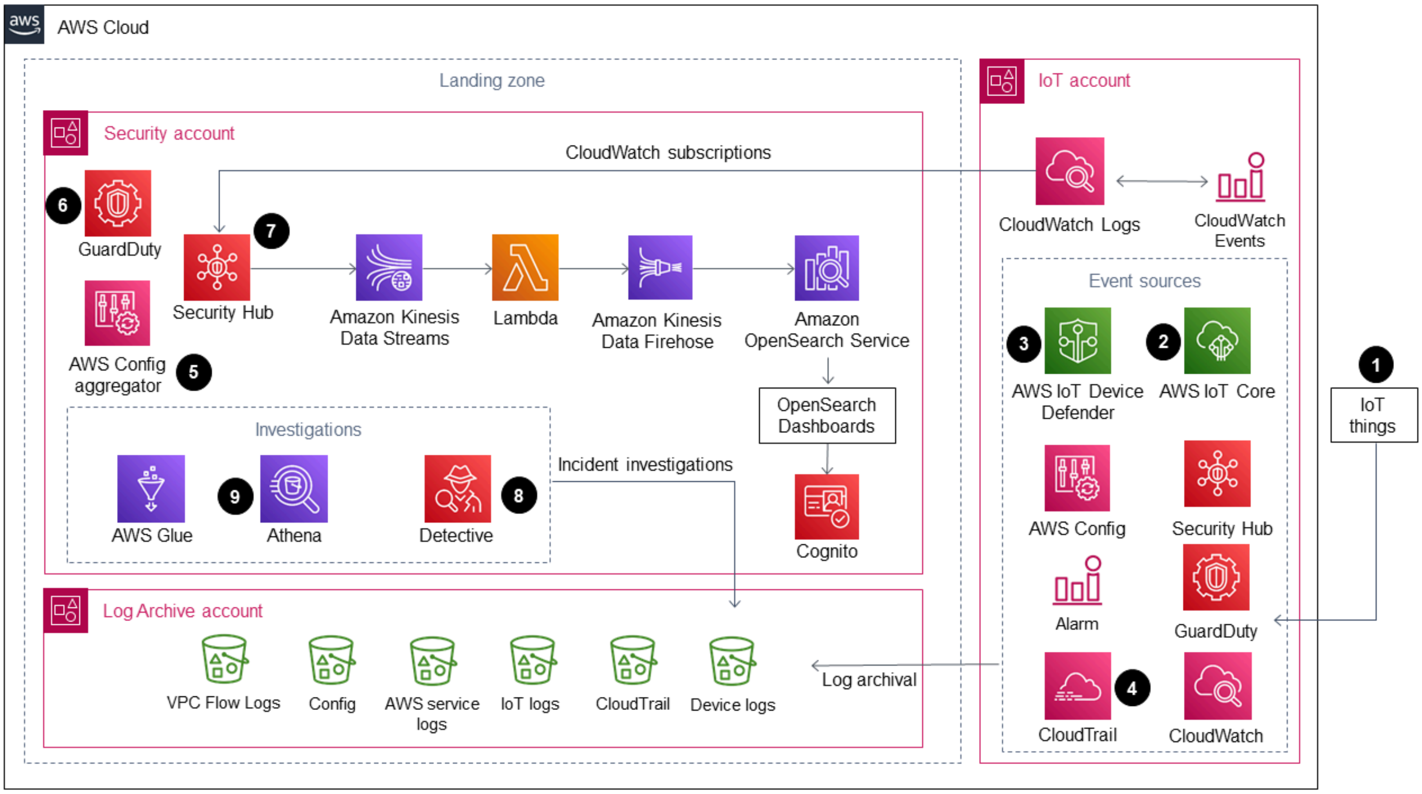
- An existing landing zone environment. For more information about this, see the guide Setting up a secure and scalable multi-account AWS environment on the AWS Prescriptive Guidance website.

- The following accounts must be available in your landing zone:
  - **Log Archive account** – This account is for users that need to access the logging information for accounts in your landing zone's organizational units (OUs). For more information about this, see the Security OU – Log Archive account section of the guide AWS Security Reference Architecture on the AWS Prescriptive Guidance website.
  - **Security account** – Your security and compliance teams use this account for auditing or to perform emergency security operations. This account is also designated as the administrator account for Amazon GuardDuty. Users from the administrator account can configure GuardDuty, in addition to viewing and managing GuardDuty findings for their own account and all member accounts. For more information about this, see Managing multiple accounts in GuardDuty in the GuardDuty documentation.
  - **IoT account** – This account is for your IoT environment.

Architecture

This pattern extends the Centralized Logging solution from the AWS Solutions Library to collect and process security-related IoT events. The Centralized Logging solution is deployed in the Security account and helps collect, analyze, and display Amazon CloudWatch logs in a single dashboard. This solution consolidates, manages, and analyzes log files from multiple sources. Finally, the Centralized Logging solution also uses Amazon OpenSearch Service and OpenSearch Dashboards to show a unified view of all log events.

The following architecture diagram shows the key components of an IoT security logging and reference architecture on the AWS Cloud.



The diagram shows the following workflow:

1. IoT things are the devices that must be monitored for anomalous security events. These devices run an agent to publish security events or metrics to AWS IoT Core and AWS IoT Device Defender.
2. When AWS IoT logging is enabled, AWS IoT sends progress events about each message as it passes from your devices through the message broker and rules engine to Amazon CloudWatch Logs. You can use CloudWatch Logs subscriptions to push events to a Centralized Logging solution. For more information about this, see AWS IoT metrics and dimensions in the AWS IoT Core documentation.
3. AWS IoT Device Defender helps monitor insecure configurations and security metrics for your IoT devices. When an anomaly is detected, alarms notify Amazon Simple Notification Service (Amazon SNS), which has an AWS Lambda function as a subscriber. The Lambda function sends the alarm as a message to CloudWatch Logs. You can use CloudWatch Logs subscriptions to push events to your Centralized Logging solution. For more information about this, see Audit checks, Upload device-side logs to CloudWatch, and Configure AWS IoT logging in the AWS IoT Core documentation.

4. AWS CloudTrail logs AWS IoT Core control plane actions that make changes (for example, creating, updating, or attaching APIs). When CloudTrail is set up as part of a landing zone implementation, it sends events to CloudWatch Logs. You can use subscriptions to push events to your Centralized Logging solution.
5. AWS Config managed rules or custom rules evaluate resources that are part of your IoT environment. Monitor your [compliance change notifications](#) using CloudWatch Events with CloudWatch Logs as the target. After compliance change notifications are sent to CloudWatch Logs, you can use subscriptions to push events to your Centralized Logging solution.
6. Amazon GuardDuty continuously analyzes CloudTrail management events and helps identify API calls made to AWS IoT Core endpoints from known malicious IP addresses, unusual geolocations, or anonymizing proxies. Monitor GuardDuty notifications using CloudWatch Events with log groups in CloudWatch Logs as the target. When GuardDuty notifications are sent to CloudWatch Logs, you can use subscriptions to push events to your Centralized Monitoring solution or use the GuardDuty console in your Security account to view the notifications.
7. AWS Security Hub monitors your IoT account by using security best practices. Monitor Security Hub notifications by using CloudWatch Events with log groups in CloudWatch Logs as the target. When Security Hub notifications are sent to CloudWatch Logs, use subscriptions to push events to your Centralized Monitoring solution or use the Security Hub console in your Security account to view the notifications.
8. Amazon Detective evaluates and analyzes information to isolate the root cause and take action on security findings for unusual calls to AWS IoT endpoints or other services in your IoT architecture.
9. Amazon Athena queries logs stored in your Log Archive account to enhance your understanding of security findings and identify trends and malicious activities.

Tools

- Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL.
- AWS CloudTrail helps you enable governance, compliance, and operational and risk auditing of your AWS account.
- Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

- Amazon CloudWatch Logs centralizes the logs from all your systems, applications, and AWS services that you use. You can view and monitor the logs, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis.
- AWS Config provides a detailed view of the configuration of AWS resources in your AWS account.
- Amazon Detective makes it easy to analyze, investigate, and quickly identify the root cause of security findings or suspicious activities.
- AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it simple and cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams.
- Amazon GuardDuty is a continuous security monitoring service.
- AWS IoT Core provides secure, bi-directional communication for Internet-connected devices (such as sensors, actuators, embedded devices, wireless devices, and smart appliances) to connect to the AWS Cloud over MQTT, HTTPS, and LoRaWAN.
- AWS IoT Device Defender is a security service that allows you to audit the configuration of your devices, monitor connected devices to detect abnormal behavior, and mitigate security risks.
- Amazon OpenSearch Service is a managed service that makes it easy to deploy, operate, and scale OpenSearch clusters in the AWS Cloud.
- AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Security Hub provides a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.
- Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Set up an IoT account in your landing zone environment

Task	Description	Skills required
Validate the security guardrail s in the IoT account.	Validate that the guardrails for CloudTrail, AWS Config,	AWS administrator

Task	Description	Skills required
	GuardDuty, and Security Hub are enabled in your IoT account.	
Validate that your IoT account is configured as a member account of your Security account.	Validate that your IoT account is configured and associate d as a member account for GuardDuty and Security Hub in your Security account. For more information about this, see Managing GuardDuty accounts with AWS Organizat ions in the GuardDuty documentation and Managing administrator and member accounts in the Security Hub documentation.	AWS administrator
Validate log archiving.	Validate that CloudTrail, AWS Config, and VPC Flow Logs are stored in the Log Archive account.	AWS administrator

Set up the Centralized Logging solution

Task	Description	Skills required
Set up the Centralized Logging solution in your Security account.	Sign in to the AWS Management Console for your Security account and set up the Centralized Logging solution from the AWS Solutions Library to collect, analyze, and display	AWS administrator

Task	Description	Skills required
	CloudWatch Logs in Amazon OpenSearch Service and OpenSearch Dashboards. For more information about this, see Collect, analyze, and display Amazon CloudWatch Logs in a single dashboard with the Centralized Logging solution from the Centralized Logging implementation guide in the AWS Solutions Library.	

Set up and configure AWS resources in your IoT account

Task	Description	Skills required
Set up AWS IoT logging.	Sign in to the AWS Management Console for your IoT account. Set up and configure AWS IoT Core to send logs to CloudWatch Logs. For more information about this, see Configure AWS IoT logging and Monitor AWS IoT using CloudWatch Logs in the AWS IoT Core documentation.	AWS administrator
Set up AWS IoT Device Defender.	Set up AWS IoT Device Defender to audit your IoT resources and detect anomalies.	AWS administrator

Task	Description	Skills required
	For more information about this, see Getting started with AWS IoT Device Defender in the AWS IoT Core documenta tion.	
Set up CloudTrail.	Set up CloudTrail to send events to CloudWatch Logs. For more information about this, see Sending events to CloudWatch Logs in the CloudTrail documentation.	AWS administrator
Set up AWS Config and AWS Config rules.	Set up AWS Config and the required AWS Config rules. For more information about this, see Setting up AWS Config with the console and Adding AWS Config rules in the AWS Config documenta tion.	AWS administrator
Set up GuardDuty.	Set up and configure GuardDuty to send findings to Amazon CloudWatch Events with log groups in CloudWatc h Logs as the target. For more information about this, see Creating custom responses to GuardDuty findings with Amazon CloudWatch Events in the GuardDuty documentation.	AWS administrator

Task	Description	Skills required
Set up Security Hub.	Set up Security Hub and enable the CIS AWS Foundations Benchmark and AWS Foundational Security Best Practices standards. For more information about this, see Automated response and remediation in the Security Hub documentation.	AWS administrator
Set up Amazon Detective.	Set up Detective to facilitate analysis of security findings. For more information about this, see Getting started with Amazon Detective in the Amazon Detective documenta tion.	AWS administrator
Set up Amazon Athena and AWS Glue.	Set up Athena and AWS Glue to query the AWS service logs that conduct security incident investigations. For more information about this, see Querying AWS service logs in the Amazon Athena documentation.	AWS administrator

Related resources

- What is a landing zone?

Set up and troubleshoot AWS IoT Greengrass with client devices

Marouane Sefiani and Akalanka De Silva, Amazon Web Services

Summary

AWS IoT Greengrass is an open-source edge runtime and cloud service for building, deploying, and managing Internet of Things (IoT) software on edge devices. Use cases for AWS IoT Greengrass include:

- Smart homes where an AWS IoT Greengrass gateway is used as a hub for home automation
- Smart factories where AWS IoT Greengrass can facilitate ingestion and local processing of data from the shop floor

AWS IoT Greengrass can act as a secure, authenticated, MQTT connection endpoint for other edge devices (also known as *client devices*), which otherwise would typically connect directly to AWS IoT Core. This capability is useful when client devices do not have direct network access to the AWS IoT Core endpoint.

You can set up AWS IoT Greengrass for use with client devices for the following use cases:

- For client devices to send data to AWS IoT Greengrass
- For AWS IoT Greengrass to forward data to AWS IoT Core
- To take advantage of advanced AWS IoT Core rules engine features

These capabilities require installing and configuring the following components on the AWS IoT Greengrass device:

- MQTT broker
- MQTT bridge
- Client device authentication
- IP detector

In addition, published messages from client devices must be in JSON format or Protocol Buffers (protobuf) format.

This pattern describes how to install and configure these required components, and provides troubleshooting tips and best practices.

Prerequisites and limitations

Prerequisites

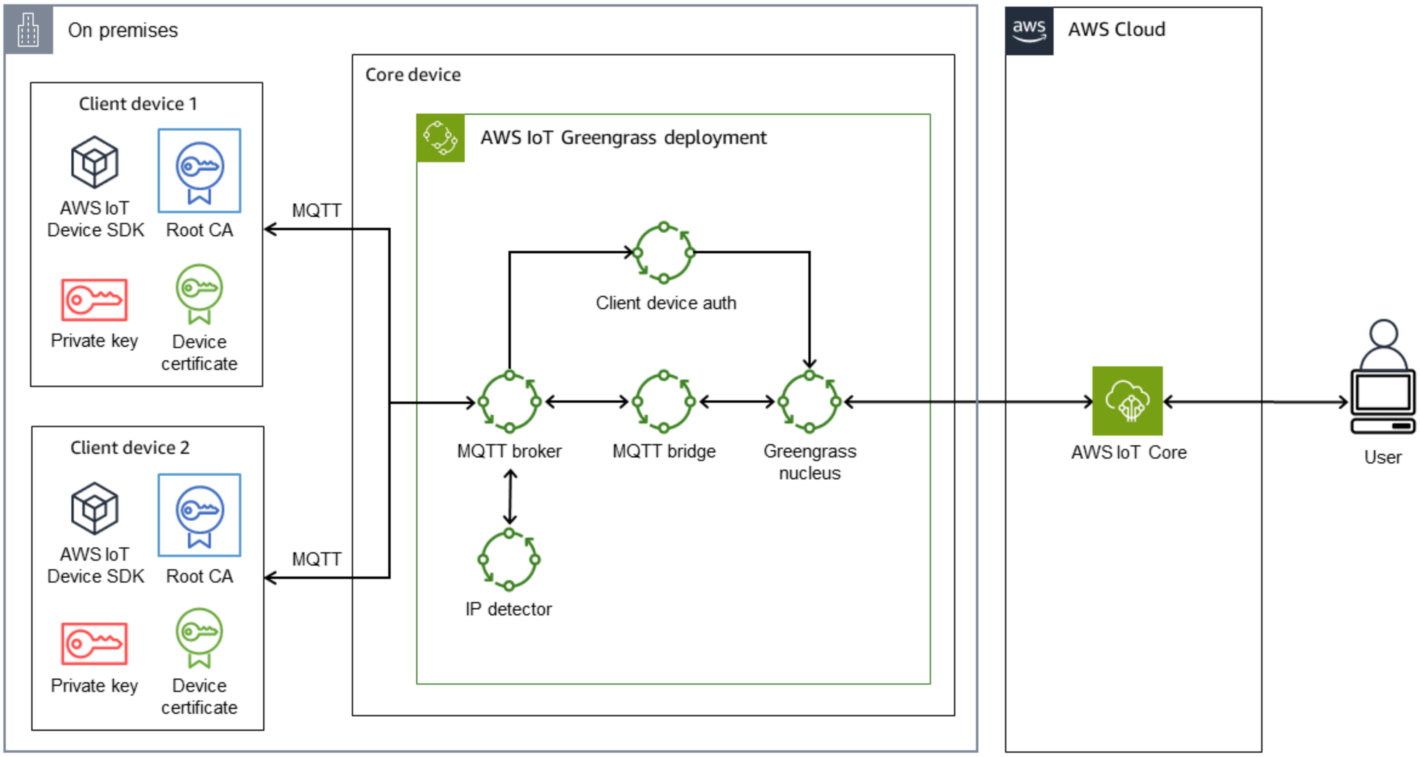
- An active AWS account
- AWS Command Line Interface (AWS CLI) version 2
- Two client devices running Python 3.7 or later
- One core device running Java Runtime Environment (JRE) version 8 or later, and Amazon Corretto 11 or OpenJDK 11

Limitations

- You must choose an AWS Region where AWS IoT Core is available. For the current list of Regions for AWS IoT Core, see AWS Services by Region.
- The core device must have at least 172 MB RAM and 512 MB of disk space.

Architecture

The following diagram shows the solution architecture for this pattern.



The architecture includes:

- Two client devices. Each device contains a private key, a device certificate, and a root certificate authority (CA) certificate. The AWS IoT Device SDK, which contains an MQTT client, is also installed on each client device.
- A core device that has AWS IoT Greengrass deployed with the following components:
  - MQTT broker
  - MQTT bridge
  - Client device authentication
  - IP detector

This architecture supports the following scenarios:

- Client devices can use their MQTT client to communicate with one another through the core device's MQTT broker.
- Client devices can also communicate with AWS IoT Core in the cloud through the core device's MQTT broker and the MQTT bridge.

- AWS IoT Core in the cloud can send messages to client devices through the MQTT test client and the core device's MQTT bridge and MQTT broker.

For more information about the communications between client devices and the core device, see the Additional information section.

Tools

AWS services

- AWS IoT Greengrass is an open source Internet of Things (IoT) edge runtime and cloud service that helps you build, deploy, and manage IoT applications on your devices.
- AWS IoT Core provides secure, bidirectional communication for internet-connected devices to connect to the AWS Cloud.
- AWS IoT Device SDK is a software development kit that includes open-source libraries, developer guides with samples, and porting guides so that you can build innovative IoT products or solutions on your choice of hardware platforms.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Best practices

- The payload of the messages from client devices should be in either JSON or Protobuf format in order to take advantage of the advanced features of the AWS IoT Core rules engine, such as transformation and conditional actions.
- Configure the MQTT bridge to allow bidirectional communication.
- Configure and deploy the IP detector component in AWS IoT Greengrass to ensure that the core device's IP addresses are included in the subject alternative name (SAN) field of the MQTT broker certificate.

Epics

Set up the core device

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Set up AWS IoT Greengrass on your core device.</td><td>Install the AWS IoT Greengrass Core software by following the instructions in the developer guide.</td><td>AWS IoT Greengrass</td></tr>
<tr><td>Check the status of your installation.</td><td>Use the following command to check the status of the AWS IoT Greengrass service on your core device:
<pre>sudo systemctl status
 greengrass.service</pre>The expected output of the command is:
<pre>Launched Nucleus
 successfully</pre></td><td>General AWS</td></tr>
<tr><td>Set up an IAM policy and attach it to the Greengrass service role.</td><td>1. Create an IAM policy to allow communications to and from the MQTT bridge. Here’s an example policy:
<pre>{
    "Version":
 "2012-10-17",
    "Statement": [

        {
            "Effect":
 "Allow",</pre></td><td>General AWS</td></tr>
</table>

Task	Description	Skills required

	``
	              "Action":
	 [
	
	 "iot:*"
	              ],
	              "Resource
	": "*"
	        },
	        {
	              "Sid":
	 "GreengrassActions
	",
	              "Effect":
	 "Allow",
	              "Action":
	 [
	
	 "greengrass:*"
	              ],
	              "Resource
	": "*"
	        }
	    ]
	}
	``
	2. Attach the policy to the Greengrass service role. To get the service role, use the command:
	``
	aws greengrassv2
	 get-service-role-f
	or-account --region
	 <region>
	``
	where `<region>` refers to your AWS Region.	

Task	Description	Skills required
Configure and deploy required components in the AWS IoT Greengrass core device.	Configure and deploy the following components: • `greengrass.clientdevices.mqtt.Moquette` (see configuration details) • `greengrass.clientdevices.mqtt.Bridge` (see configuration details and the next task) • `greengrass.clientdevices.Auth` (see configuration details and the task after the next one) • `aws.greengrass.clientdevices.IPDetector` (see configuration details)	AWS IoT Greengrass

Task	Description	Skills required

Confirm that the MQTT bridge allows bidirectional communication.	To relay MQTT messages between client devices and AWS IoT Core, configure and deploy the MQTT bridge component and specify the topics to relay. Here's an example:
	``
	{
	  "mqttTopicMapping":
	 {
	    "ClientDevicesToCl
	oud": {
	      "topic": "dt/#",
	      "source":
	 "LocalMqtt",
	      "target":
	 "IotCore"
	    },
	    "CloudToClientDevi
	ces": {
	      "topic": "cmd/#",
	      "source":
	 "IotCore",
	      "target":
	 "LocalMqtt"
	    }
	  }
	}
	``
		AWS IoT Greengrass

Task	Description	Skills required
Confirm that the auth component allows client devices to connect and publish or subscribe to topics.	The following `aws.greengrass.clientdevices.Auth` configuration allows all client devices to connect, publish messages, and subscribe to all topics. `{ "deviceGroups": { "formatVersion": "2021-03-05", "definitions": { "MyPermissiveDeviceGroup": { "selectionRule": "thingName: *", "policyName": "MyPermissivePolicy" } }, "policies": { "MyPermissivePolicy": { "AllowAll": { "statementDescription": "Allow client devices to perform all actions.", "operations": ["*"], "resources": ["*"] } } }`	AWS IoT Greengrass

Task	Description	Skills required

	``
	  }
	}
	``	

Set up client devices

Task	Description	Skills required
Install the AWS IoT Device SDK.	Install the AWS IoT Device SDK on client devices. For a full list of supported languages and the associated SDKs, see the [AWS IoT Core documentation]. For example, the AWS IoT Device SDK for Python SDK is [located on GitHub]. To install this SDK: 1. Confirm that Python 3.7 or later is installed, as instructed on the [Prerequisites page] of the GitHub repository. 2. Use the **pip** command to install the SDK. For MacOS and Linux:: `python3 -m pip install awsiotsdk` For Windows:	General AWS IoT

Task	Description	Skills required

	```python -m pip install awsiotsdk```
	
	Alternatively, you can install the SDK from the source repository:
	
	```# Create a workspace directory to hold all the SDK files mkdir sdk-workspace cd sdk-workspace # Clone the repository git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git # Install using Pip (use 'python' instead of 'python3' on Windows) python3 -m pip install ./aws-iot-device-sdk-python-v2```	

Task	Description	Skills required

Create a thing.	1. In the [AWS IoT console](https://console.aws.amazon.com/iot/home), if a **Get started** button appears, choose it. Otherwise, in the navigation pane, choose **Security**, **Policies**.
	2. If the **You don't have any policies yet** dialog box appears, choose **Create a policy**. Otherwise, choose **Create**.
	3. Enter a name for the AWS IoT policy (for example, `ClientDevicePolicy`).
	4. In the **Add statements** section, replace the existing policy with the following JSON code. Replace `<region>` and `<account>` with your AWS Region and AWS account number.
	
	`{`
	`  "Version": "2012-10-17",`
	`  "Statement": [{`
	`      "Effect": "Allow",`
	`      "Action": "iot:Connect",`
	`      "Resource": "arn:aws:iot:region:account:client/*"`
	`    },`
		AWS IoT Core

Task	Description	Skills required

	``
	        {
	            "Effect":
	 "Allow",
	            "Action":
	 "iot:Publish",
	            "Resource":
	 "*"
	        },
	        {
	            "Effect":
	 "Allow",
	            "Action":
	 "iot:Receive",
	            "Resource":
	 "*"
	        },
	        {
	            "Effect":
	 "Allow",
	            "Action":
	 "iot:Subscribe",
	            "Resource":
	 "*"
	        },
	        {
	            "Effect":
	 "Allow",
	            "Action": [
	                "iot:GetT
	hingShadow",
	                "iot:Upda
	teThingShadow",
	                "iot:Dele
	teThingShadow"
	            ],
	            "Resource":
	 "arn:aws:iot:regio
	n:account:thing/*"
	
	        }
	    ]
	``	

Task	Description	Skills required

	``
	}
	``
	5. Choose **Create**.
	6. On the [AWS IoT console,](https://console.aws.amazon.com/iot/home) in the navigation pane, choose **Manage**, **Things**.
	7. If the **You don't have any things yet** dialog box is displayed, choose **Register a thing**. Otherwise, choose **Create**.
	8. On the **Creating AWS IoT things** page, choose **Create a single thing**.
	9. On the **Add your device to the device registry** page, enter a name for your IoT thing (for example, `ClientDevice1`), and then choose **Next**.
	
	Note
	You can't change the name of a thing after you create it. To change the name, you must create a new thing, give it the new name, and then delete the old thing.	

Task	Description	Skills required
	10. On the **Add a certificate for your thing** page, choose **Create certificate**. 11. Choose the **Download** links to download the certificate, private key, and root CA certificate. **Important** This is your only opportunity to download your certificate and private key. 12. To activate the certificate, choose **Activate**. The certificate must be active for a device to connect to AWS IoT. 13. Choose **Attach a policy**. 14. For **Add a policy for your thing**, choose **ClientDevicePolicy**, **Register Thing**.	

Task	Description	Skills required
Download the CA certificate from the Greengrass core device.	If you expect the Greengrass core device to work in offline environments, you have to make the Greengrass core CA certificate available to the client device so it can verify the MQTT broker's certificate (which is issued by the Greengrass core CA). Therefore, it is important to obtain a copy of this certificate. Use one of the following approaches to download the CA certificate: • If you have network access to the AWS IoT Greengrass device from your PC, enter `https://<device IP>:8883` in your web browser and view the MQTT broker certificate and the CA certificate. You can also save the CA certificate to the client device. • Alternatively, you can use the OpenSSL command line: `openssl s_client -showcerts -connect <device IP>:8883`	General AWS

Task	Description	Skills required
Copy credentials in the client devices.	Copy the Greengrass core CA certificate, the device certifica te, and the private key in the client devices.	General AWS

Task	Description	Skills required
Associate client devices with the core device.	Associate client devices with a core device so that they can discover the core device. The client devices can then use the [Greengrass discovery API](#) to retrieve connectivity information and certificates for their associated core devices. For more information, see [Associate client devices](#) in the AWS IoT Greengrass documentation. 1. On the [AWS IoT Greengrass console](#), choose **Core devices**. 2. Choose the core device to manage. 3. On the core device's details page, choose the **Client devices** tab. 4. In the **Associated client devices** section, choose **Associate client devices**. 5. In the **Associate client devices with core device** modal, do the following for each client device to associate: a. Enter the name of the AWS IoT thing to associate as a client device. b. Choose **Add**.	AWS IoT Greengrass

Task	Description	Skills required
	6. Choose **Associate**. The client devices that you associated can now use the Greengrass discovery API to discover this core device.	

Send and receive data

Task	Description	Skills required
Send data from one client device to another client device.	Use the MQTT client in your device to publish a message on the `dt/client1/sensor` topic.	General AWS
Send data from the client device to AWS IoT Core.	Use the MQTT client in your device to publish a message on the `dt/client1/sensor` topic. In the MQTT test client, subscribe to the topic that the device is sending messages on, or subscribe to **#** for all topics (see details).	General AWS
Send messages from AWS IoT Core to client devices.	On the MQTT test client page, in the **Publish to a topic** tab, in the **Topic name** field, enter the topic name of your message. In this example, use `cmd/client1` for the topic.	General AWS

Troubleshooting

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td>

Unable to verify server certificate error

</td><td>

This error occurs when the MQTT client cannot verify the certificate that's presented by the MQTT broker during the TLS handshake. The most common reason is that the MQTT client doesn't have the CA certificate. Follow these steps to make sure that the CA certificate is provided to the MQTT client.

1. If you have network access to the AWS IoT Greengrass device from your PC, enter `https://<device IP>:8883` in a browser window to view the MQTT broker certificate and the CA certificate. You can also save the CA certificate to the client device.

   Alternatively use the OpenSSL command line:

   ```
   openssl s_client -showcerts -connect
    <device IP>:8883
   ```

2. Save the contents of the Moquette CA and Greengrass Core CA certificates into files, and then view the decoded contents by using the command:

   ```
   openssl x509 -in <Name of CA>.pem -
   text
   ```

   The Moquette CA certificate should show the SAN field as in this example:

</td></tr>
</table>

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td></td><td><pre><code>X509v3 Subject Alternative Name:
 IP Address:XXX.XXX.XXX.XXX, IP
Address:127.0.0.1, DNS:localhost</code></pre></td></tr>
<tr><td>Unable to verify server name error</td><td> This errors occurs when the MQTT client can’t verify that it’s connecting to the correct server. The most common reason is that the IP address of the Greengrass device isn’t listed in the SAN field of the certificate. Follow the instructions in the previous solution to obtain the MQTT broker certifica te and verify that the SAN field contains the IP address of the AWS IoT Greengrass device, as explained in the Additional information section. If not, confirm that the IP detector component is installed correctly and restart the core device. </td></tr>
<tr><td>Unable to verify server name only when connecting from an embedded client device</td><td>Mbed TLS, which is a popular TLS library used in embedded devices, currently supports DNS name verification only in the SAN field of the certificate, as shown in the Mbed TLS library code. Because the core device doesn’t have its own domain name and depends on the IP address, TLS clients that use Mbed TLS will fail the server name verification during the TLS handshake, causing a connection failure. We recommend that you add the SAN IP address verification to your Mbed TLS library at the x509_crt_check_san function.</td></tr>
</table>

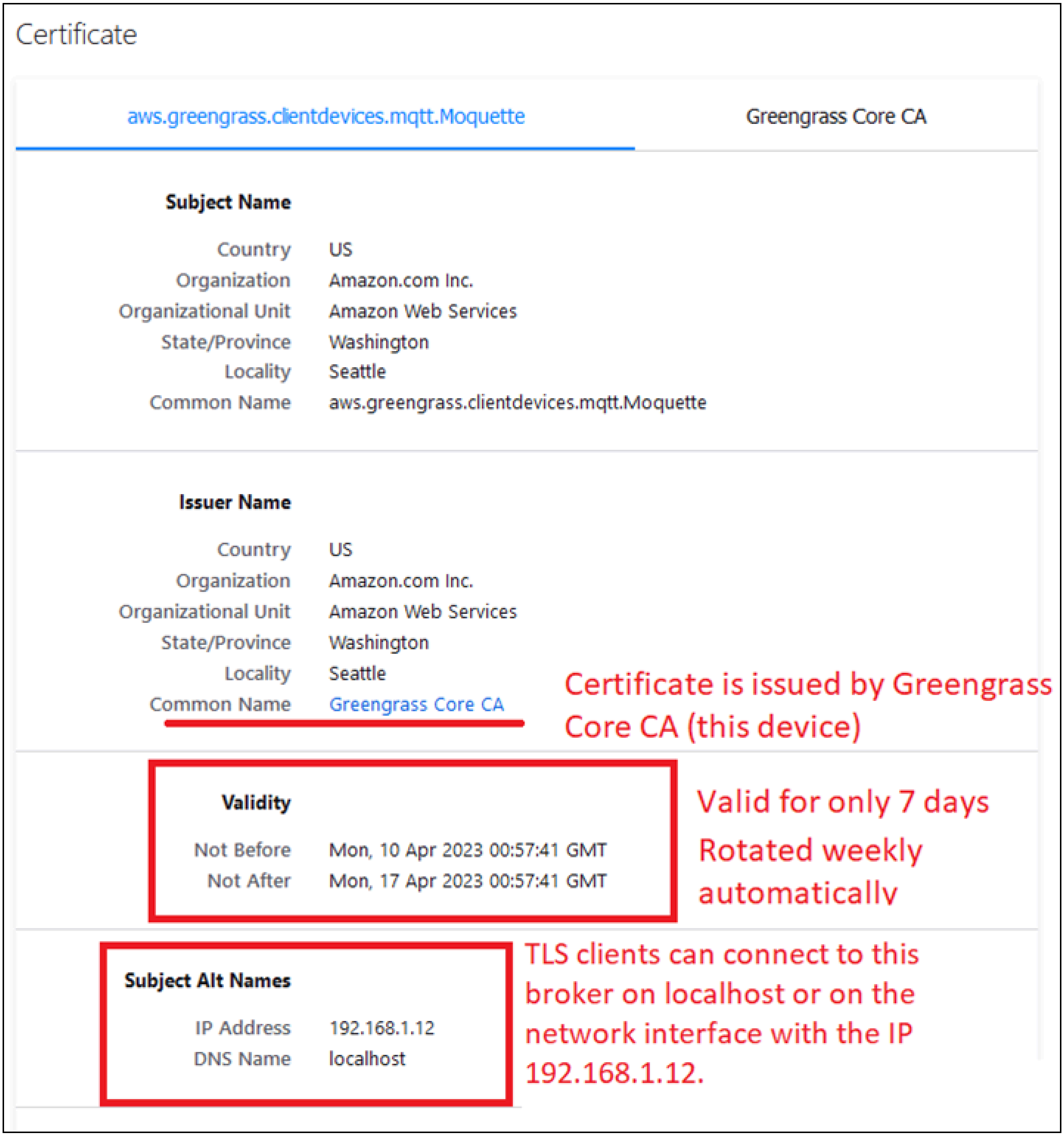
Related resources

- AWS IoT Greengrass documentation
- AWS IoT Core documentation
- MQTT broker component
- MQTT bridge component
- Client device auth component
- IP detector component
- AWS IoT Device SDKs
- Implementing Local Client Devices with AWS IoT Greengrass (AWS blog post)
- RFC 5280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

Additional information

This section provides additional information about the communications between the client devices and the core device.

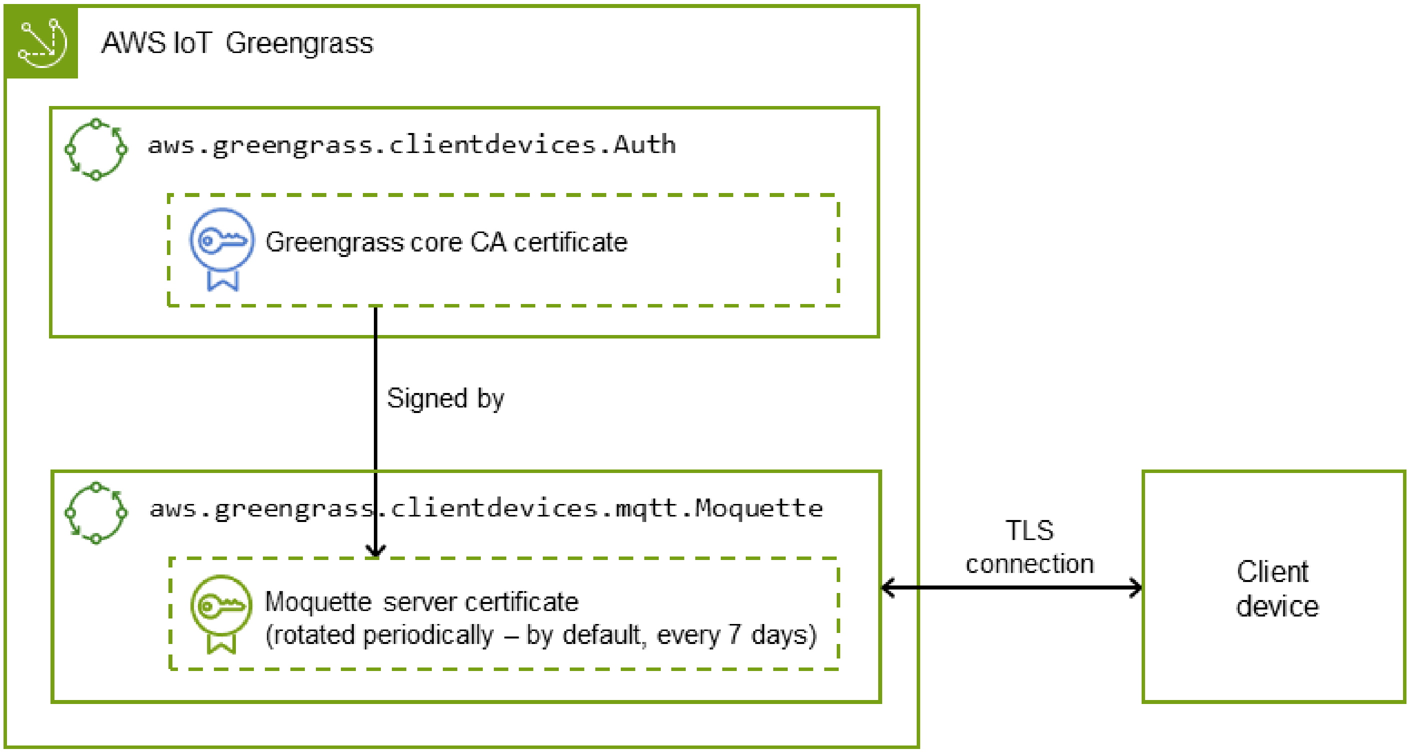
The MQTT broker listens on port 8883 in the core device for a TLS client connection attempt. The following illustration shows an example MQTT broker's server certificate.



The example certificate displays the following details:

- The certificate is issued by the AWS IoT Greengrass Core CA, which is local and specific to the core device; that is, it acts as a local CA.

- This certificate is automatically rotated every week by the client auth component as shown in the following illustration. You can set this interval in the client auth component configuration.

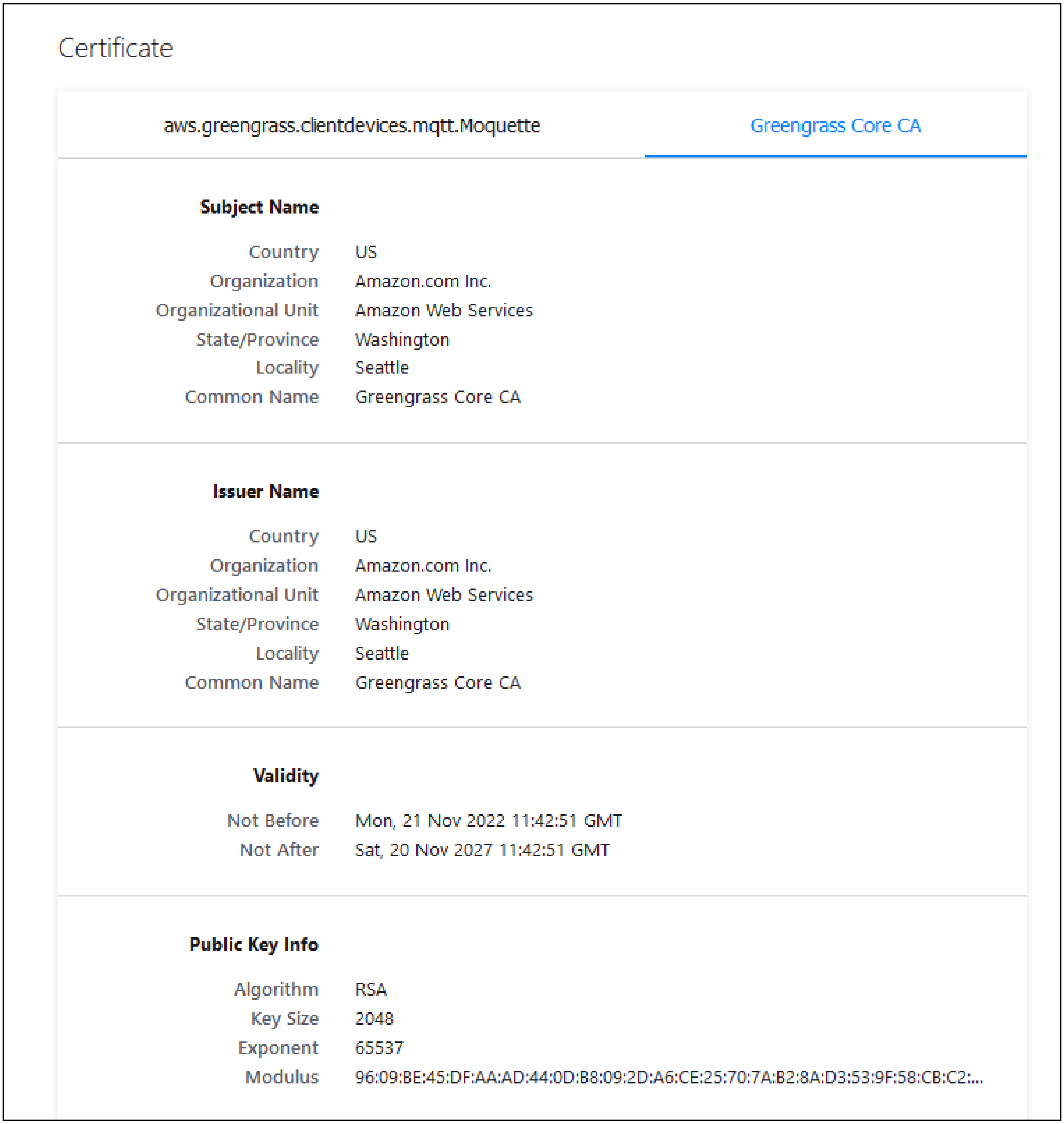


- The subject alternative name (SAN) plays a critical role in the server name verification on the TLS client end. It helps the TLS client ensure that it connects to the correct server and helps avoid man-in-the-middle attacks during TLS session setup. In the example certificate, the SAN field indicates that this server is listening on localhost (the local Unix domain socket), and the network interface has the IP address 192.168.1.12.

The TLS client uses the SAN field in the certificate to verify that it's connecting to a legitimate server during server verification. In contrast, during a typical TLS handshake between an HTTP server and a browser, the domain name in the common name (CN) field or SAN field is used to cross-check the domain that the browser is actually connecting to during the server verification process. If the core device doesn't have a domain name, the IP address included in the SAN field serves the same purpose. For more information, see the [Subject Alternative Name section](#) of *RFC 5280 – Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*.

Th IP detector component in AWS IoT Greengrass ensures that the correct IP addresses are included in the SAN field of the certificate.

The certificate in the example is signed by the AWS IoT Greengrass device acting as a local CA. The TLS client (MQTT client) isn't aware of this CA, so we must provide a CA certificate that looks like the following.


Certificate
aws.greengrass.clientdevices.mqtt.Moquette
Greengrass Core CA
Subject Name
Country US
Organization Amazon.com Inc.
Organizational Unit Amazon Web Services
State/Province Washington
Locality Seattle
Common Name Greengrass Core CA
Issuer Name
Country US
Organization Amazon.com Inc.
Organizational Unit Amazon Web Services
State/Province Washington
Locality Seattle
Common Name Greengrass Core CA
Validity
Not Before Mon, 21 Nov 2022 11:42:51 GMT
Not After Sat, 20 Nov 2027 11:42:51 GMT
Public Key Info
Algorithm RSA
Key Size 2048
Exponent 65537
Modulus 96:09:BE:45:DF:AA:AD:44:0D:B8:09:2D:A6:CE:25:70:7A:B2:8A:D3:53:9F:58:CB:C2:...

More patterns

- Cost-effectively ingest IoT data directly into Amazon S3 using AWS IoT Greengrass
- Deploy agentic systems on Amazon Bedrock with the CrewAI framework by using Terraform

Migration & modernization

Topics

- Migration
- Modernization
- Mainframes

Migration

Topics

- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python
- Get started with automated portfolio discovery
- Migrate on-premises Cloudera workloads to Cloudera Data Platform on AWS
- Resolve connection errors after migrating Microsoft SQL Server to the AWS Cloud
- Restart the AWS Replication Agent automatically without disabling SELinux after rebooting a RHEL source server
- Re-architect
- Rehost
- Relocate
- Replatform
- Migration patterns by workload
- More patterns

Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python

Venkata Naveen Koppula, Amazon Web Services

Summary

This pattern outlines steps for automatically creating AWS CloudFormation templates for AWS Database Migration Service (AWS DMS) using Microsoft Excel and Python.

Migrating databases using AWS DMS often involves creation of AWS CloudFormation templates to provision AWS DMS tasks. Previously, creating AWS CloudFormation templates required knowledge of the JSON or YAML programming language. With this tool, you only need basic knowledge of Excel and how to run a Python script using a terminal or command window.

As input, the tool takes an Excel workbook that includes the names of the tables to be migrated, Amazon Resource Names (ARNs) of AWS DMS endpoints, and AWS DMS replication instances. The tool then generates AWS CloudFormation templates for the required AWS DMS tasks.

For detailed steps and background information, see the blog post Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel in the AWS Database blog.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Microsoft Excel version 2016 or later
- Python version 2.7 or later
- The **xlrd** Python module (installed at a command prompt with the command: **pip install xlrd**)
- AWS DMS source and target endpoints and AWS DMS replication instance

Limitations

- The names of schemas, tables, and associated columns are transformed into lowercase characters at the destination endpoints.
- This tool doesn’t address the creation of AWS DMS endpoints and replication instances.

- Currently, the tool supports only one schema for each AWS DMS task.

Architecture

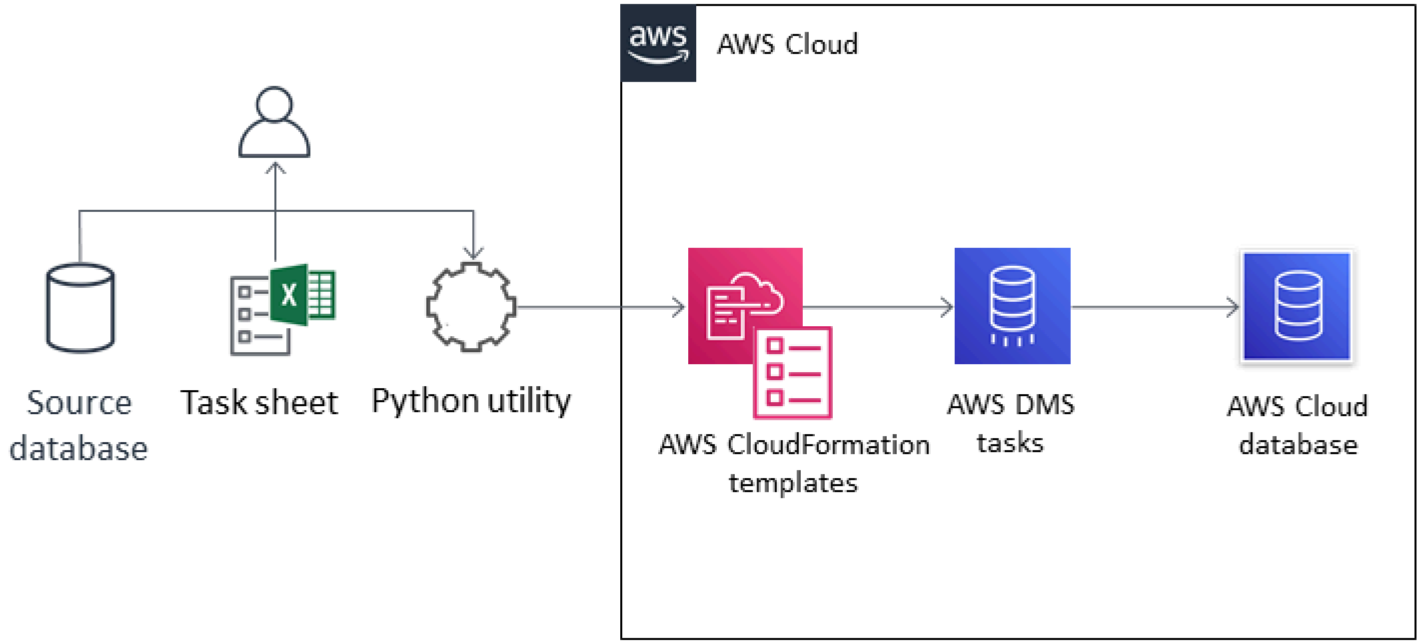
Source technology stack

- An on-premises database
- Microsoft Excel

Target technology stack

- AWS CloudFormation templates
- A database in the AWS Cloud

Architecture



Tools

- [Pycharm IDE](), or any integrated development environment (IDE) that supports Python version 3.6
- Microsoft Office 2016 (for Microsoft Excel)

Epics

Configure the network, AWS DMS replication instance, and endpoints

Task	Description	Skills required
If necessary, request a service quota increase.	Request a service quota increase for the AWS DMS tasks if needed.	General AWS
Configure the AWS Region, virtual private clouds (VPCs), CIDR ranges, Availability Zones, and subnets.		General AWS
Configure the AWS DMS replication instance.	The AWS DMS replication instance can connect to both on-premises and AWS databases.	General AWS
Configure AWS DMS endpoints.	Configure endpoints for both the source and target databases.	General AWS

Prepare the worksheets for AWS DMS tasks and tags

Task	Description	Skills required
Configure the tables list.	List all tables involved in the migration.	Database
Prepare the tasks worksheet.	Prepare the Excel worksheet using the tables list you configured.	General AWS, Microsoft Excel
Prepare the tags worksheet.	Detail the AWS resource tags to attach to the AWS DMS tasks.	General AWS, Microsoft Excel

Download and run the tool

Task	Description	Skills required
Download and extract the template generation tool from the GitHub repository.	GitHub repository: https://github.com/aws-samples/dms-cloudformation-templates-generator/	
Run the tool.	Follow the detailed instructions in the blog post listed under "References and help."	

Related resources

- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel (blog post)
- DMS CloudFormation Templates Generator (GitHub repository)
- Python documentation
- xlrd description and download
- AWS DMS documentation
- AWS CloudFormation documentation

Get started with automated portfolio discovery

Pratik Chunawala and Rodolfo Jr. Cerrada, Amazon Web Services

Summary

Assessing the portfolio and collecting metadata is a critical challenge when migrating applications and servers to the Amazon Web Services (AWS) Cloud, especially for large migrations that have more than 300 servers. Using an automated portfolio discovery tool can help you collect information about your applications, such as the number of users, frequency of use, dependencies, and information about the application's infrastructure. This information is essential when planning migration waves so that you can properly prioritize and group applications with similar traits. Using a discovery tool streamlines communication between the portfolio team and the application owners because the portfolio team can validate the results of the discovery tool rather than manually collecting the metadata. This pattern discusses key considerations for selecting an automated discovery tool and information about how to deploy and test one in your environment.

This pattern includes a template, which is a starting point for building your own checklist of high-level activities. Next to the checklist is template for a responsible, accountable, consulted, informed (RACI) matrix. You can use this RACI matrix to determine who is responsible for each task in your checklist.

Epics

Select a discovery tool

Task	Description	Skills required
Determine whether a discovery tool is appropriate for your use case.	A discovery tool might not be the best solution for your use case. Consider the amount of time required to select, procure, prepare, and deploy a discovery tool. It can take 4–8 weeks to set up the scanning appliance for an agentless discovery tool in your environment or to	Migration lead, Migration engineer

Task	Description	Skills required
	install agents to all in-scope workloads. Once deployed, you must allow 4–12 weeks for the discovery tool to collect metadata by scanning the application workloads and performing application stack analysis. If you are migrating fewer than 100 servers, you might be able to manually collect the metadata and analyze dependencies faster than the time required to deploy and collect metadata with an automated discovery tool.	
Select a discovery tool.	Review the **Considerations for selecting an automated discovery tool** in the Additional information section. Determine the appropriate criteria for selecting a discovery tool for your use case, and then evaluate each tool against those criteria. For a comprehensive list of automated discovery tools, see Discovery, Planning, and Recommendation migration tools.	Migration lead, Migration engineer

Prepare for installation

Task	Description	Skills required
Prepare the pre-deployment checklist.	Create a checklist of the tasks you must complete before deploying the tool. For an example, see Predeployment Checklist on the Flexera documentation website.	Build lead, Migration engineer, Migration lead, Network administrator
Prepare the network requirements.	Provision the ports, protocols , IP addresses, and routing necessary for the tool to run and access the target servers. For more information, see the installation guide for your discovery tool. For an example, see Deployment Requirements on the Flexera documentation website.	Migration engineer, Network administrator, Cloud architect
Prepare the account and credential requirements.	Identify the credentials you need to access the target servers and to install all of the tool's components.	Cloud administrator, General AWS, Migration engineer, Migration lead, Network administrator, AWS administrator
Prepare the appliances on which you will install the tool.	Ensure that the appliances on which you will install the tool components meet the specifications and platform requirements for the tool.	Migration engineer, Migration lead, Network administrator
Prepare the change orders.	According to the change management process in your organization, prepare the any change orders needed, and	Build lead, Migration lead

Task	Description	Skills required
	ensure these change orders are approved.	
Send requirements to stakeholders.	Send the pre-deployment checklist and network requirements to the stakeholders. Stakeholders should review, evaluate, and prepare the necessary requirements before proceeding with the deployment.	Build lead, Migration lead

Deploy the tool

Task	Description	Skills required
Download the installer.	Download the installer or the virtual machine image. Virtual machine images typically come in Open Virtualization Format (OVF).	Build lead, Migration lead
Extract the files.	If you are using an installer, you must download and run the installer on an on-premises server.	Build lead, Migration lead
Deploy the tool on the servers.	Deploy the discovery tool on the target, on-premises servers as follows: • If your source file is a virtual machine image, deploy it into your virtual	Build lead, Migration lead, Network administrator

Task	Description	Skills required
	machine environment, such as VMware. • If your source file is an installer, run the installer to install and set up the tool.	
Log in to the discovery tool.	Follow the on-screen prompts, and log in to get started with the tool.	Migration lead, Build lead
Activate the product.	Enter your license key.	Build lead, Migration lead
Configure the tool.	Enter any credentials necessary to access the target servers, such as credentia ls for Windows, VMware, Simple Network Managemen t Protocol (SNMP), and Secure Shell Protocol (SSH), or databases.	Build lead, Migration lead

Test the tool

Task	Description	Skills required
Select test servers.	Identify a small set of non-production subnets or IP addresses that you can use to test the discovery tool. This helps you validate the scans quickly, identify and troubleshoot any errors quickly, and isolate your tests from production environme nts.	Build lead, Migration lead, Network administrator

Task	Description	Skills required
Start scanning the selected test servers.	For an agentless discovery tool, enter the subnets or IP addresses for the selected test servers in the discovery tool console, and start the scan. For an agent-based discovery tool, install the agent on the selected test servers.	Build lead, Migration lead, Network administrator
Review the scan results.	Review the scan results for the test servers. If any errors are found, troubleshoot and fix the errors. Document the errors and solutions. You reference this information in the future, and you can add this information to your portfolio runbook.	Build lead, Migration lead, Network administrator
Rescan the test servers.	Once the rescan is complete, repeat the scan until there are no errors.	Build lead, Migration lead, Network administrator

Related resources

AWS resources

- Application portfolio assessment guide for AWS Cloud migration
- Discovery, Planning, and Recommendation migration tools

Deployment guides for commonly selected discovery tools

- Deploy the RN150 virtual appliance (Flexera documentation)

- [Gatherer Installation](#) (modelizeIT documentation)
- [On-Prem Analysis Server Installation](#) (modelizeIT documentation)

Additional information

Considerations for selecting an automated discovery tool

Each discovery tool has benefits and limitations. When selecting the appropriate tool for your use case, consider the following:

- Select a discovery tool that can collect most, if not all, of the metadata you need to achieve your portfolio assessment goal.
- Identify any metadata you need to gather manually because the tool doesn't support it.
- Provide the discovery tool requirements to stakeholders so they can review and assess the tool based on their internal security and compliance requirements, such as server, network, and credential requirements.
  - Does the tool require that you install an agent in the in-scope workload?
  - Does the tool require that you set up a virtual appliance in your environment?
- Determine your data residency requirements. Some organizations don't want to store their data outside of their environment. To address this, you might need to install some components of the tool in the on-premises environment.
- Make sure the tool supports the operating system (OS) and OS version of the in-scope workload.
- Determine whether your portfolio includes mainframe, mid-range, and legacy servers. Most of the discovery tools can detect these workloads as dependencies, but some tools might not be able to get device details, such as utilization and server dependencies. Device42 and modernizeIT discovery tools both support mainframe and mid-range servers.

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip](#)

Migrate on-premises Cloudera workloads to Cloudera Data Platform on AWS

Battulga Purevragchaa and Nidhi Gupta, Amazon Web Services

Nijjwol Lamsal, Partner

Summary

This pattern describes the high-level steps for migrating your on-premises Cloudera Distributed Hadoop (CDH), Hortonworks Data Platform (HDP), and Cloudera Data Platform (CDP) workloads to CDP Public Cloud on AWS. We recommend that you partner with Cloudera Professional Services and a systems integrator (SI) to implement these steps.

There are many reasons Cloudera customers want to move their on-premises CDH, HDP, and CDP workloads to the cloud. Some typical reasons include:

- Streamline adoption of new data platform paradigms such as data lakehouse or data mesh
- Increase business agility, democratize access and inference on existing data assets
- Lower the total cost of ownership (TCO)
- Enhance workload elasticity
- Enable greater scalability; drastically reduce time to provision data services compared with legacy, on-premises install base
- Retire legacy hardware; significantly reduce hardware refresh cycles
- Take advantage of pay-as-you-go pricing, which is extended to Cloudera workloads on AWS with the Cloudera licensing model (CCU)
- Take advantage of faster deployment and improved integration with continuous integration and continuous delivery (CI/CD) platforms
- Use a single unified platform (CDP) for multiple workloads

Cloudera supports all major workloads, including Machine Learning, Data Engineering, Data Warehouse, Operational Database, Stream Processing (CSP), and data security and governance. Cloudera has offered these workloads for many years on premises, and you can migrate these workloads to the AWS Cloud by using CDP Public Cloud with Workload Manager and Replication Manager.

Cloudera Shared Data Experience (SDX) provides a shared metadata catalog across these workloads to facilitate consistent data management and operations. SDX also includes comprehensive, granular security to protect against threats, and unified governance for audit and search capabilities for compliance with standards such as Payment Card Industry Data Security Standard (PCI DSS) and GDPR.

CDP migration at a glance

Workload	Source workload	CDH, HDP, and CDP Private Cloud
	Source environment	• Windows, Linux • On-premises, colocation, or any non-AWS environment
	Destination workload	CDP Public Cloud on AWS
	Destination environment	• Deployment model: customer account • Operating model: customer/Cloudera control plane
Migration	Migration strategy (7Rs)	Rehost, replatform, or refactor
	Is this an upgrade in the workload version?	Yes
	Migration duration	• Deployment: About 1 week to create customer account, virtual private cloud (VPC), and CDP Public Cloud customer-managed environment. • Migration duration: 1-4 months, depending on the

complexity and size of the workload.

Cost	Cost of running the workload on AWS	• At a high level, the cost of a CDH workload migration to AWS assumes that you will establish a new environment on AWS. It includes accounting for personnel time and effort as well as provisioning computing resources and licensing software for the new environment. • The Cloudera cloud consumption-based pricing model gives you the flexibility to take advantage of bursting and automatic scaling capabilities. For more information, see CDP Public Cloud service rates on the Cloudera website. • Cloudera Enterprise Data Hub is based on Amazon Elastic Compute Cloud (Amazon EC2) and closely models traditional clusters. Data Hub can be customized, but this will affect costs. • CDP Public Cloud Data Warehouse, Cloudera Machine Learning, and Cloudera Data Engineering (CDE) are container-based and can be configured to scale automatically.

Infrastructure agreements and framework	System requirements	See the Prerequisites section.
	SLA	See Cloudera Service Level Agreement for CDP Public Cloud.
	DR	See Disaster Recovery in the Cloudera documentation.
	Licensing and operating model (for target AWS account)	Bring Your Own License (BYOL) model
Compliance	Security requirements	See Cloudera Security Overview in the Cloudera documentation.
	Other compliance certifications	See the information on the Cloudera website about General Data Protection Regulation (GDPR) compliance and the CDP Trust Center.

Prerequisites and limitations

Prerequisites

- AWS account requirements, including accounts, resources, services, and permissions, such as AWS Identity and Access Management (IAM) roles and policies setup
- Prerequisites for deploying CDP from the Cloudera website

The migration requires the following roles and expertise:

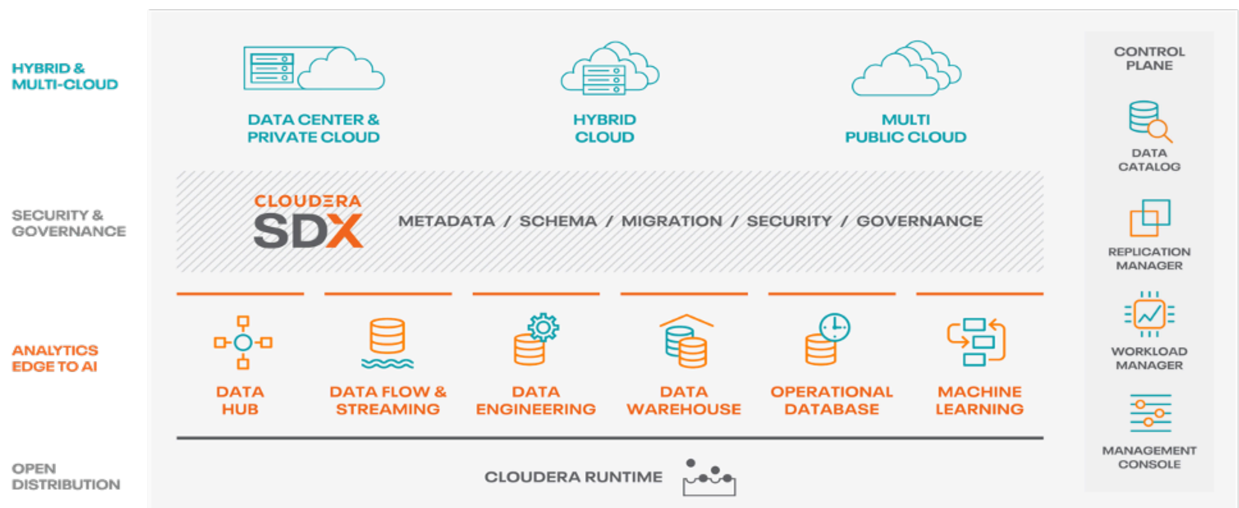
Role	Skills and responsibilities
Migration lead	Ensures executive support, team collaboration, planning, implementation, and assessment

Cloudera SME	Expert skills in CDH, HDP, and CDP administration, system administration, and architecture
AWS architect	Skills in AWS services, networking, security, and architectures

Architecture

Building to the appropriate architecture is a critical step to ensure that migration and performance meet your expectations. For your migration effort to meet this playbook's assumptions, your target data environment in the AWS Cloud, either on virtual private cloud (VPC) hosted instances or CDP, must be an equivalent match to your source environment in terms of operating system and software versions as well as major machine specifications.

The following diagram (reproduced with permission from the Cloudera Shared Data Experience data sheet) shows the infrastructure components for the CDP environment and how the tiers or infrastructure components interact.



The architecture includes the following CDP components:

- Data Hub is a service for launching and managing workload clusters powered by Cloudera Runtime. You can use the cluster definitions in Data Hub to provision and access workload clusters for custom use cases and define custom cluster configurations. For more information, see the Cloudera website.
- Data Flow and Streaming addresses the key challenges enterprises face with data in motion. It manages the following:

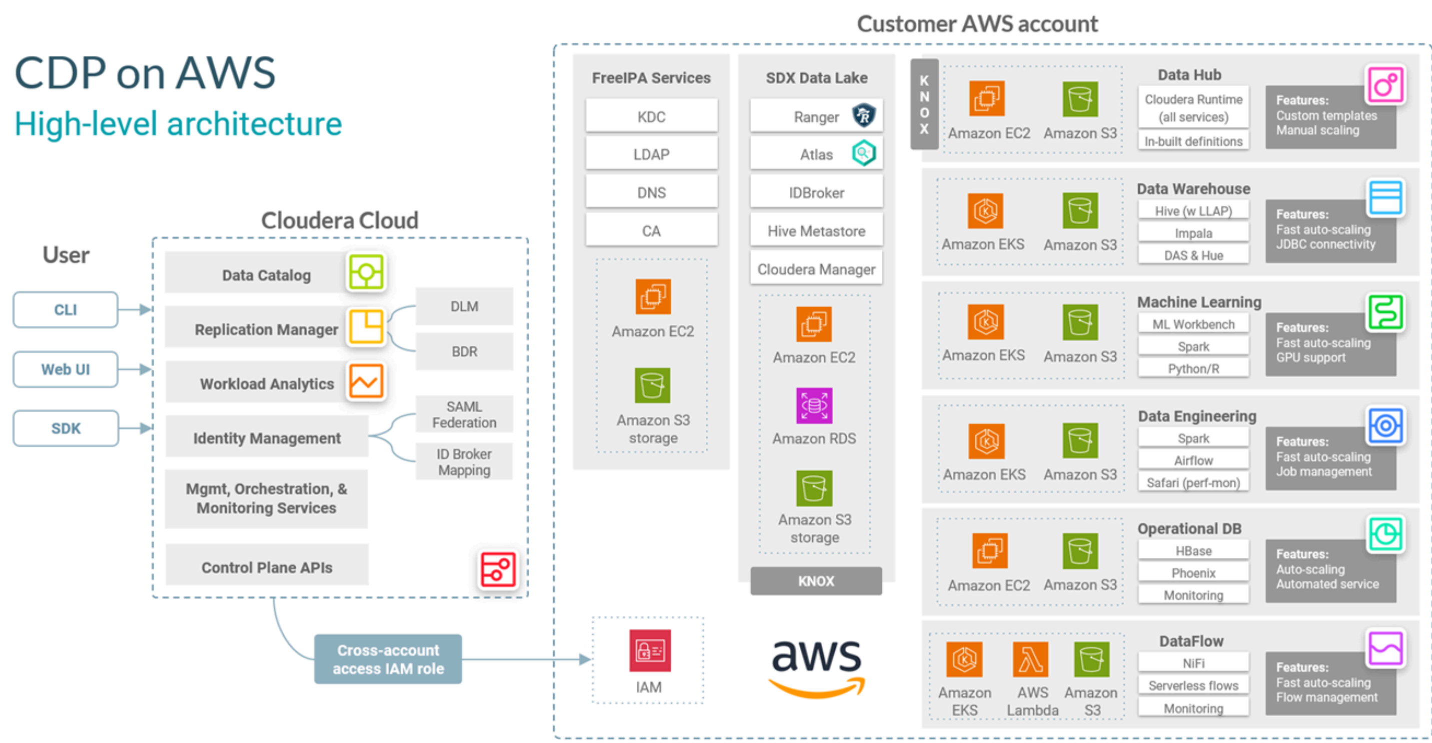
  - Processing real-time data streaming at high volume and high scale
  - Tracking data provenance and lineage of streaming data
  - Managing and monitoring edge applications and streaming sources

  For more information, see Cloudera DataFlow and CSP on the Cloudera website.

- Data Engineering includes data integration, data quality, and data governance, which help organizations build and maintain data pipelines and workflows. For more information, see the Cloudera website. Learn about support for spot instances to facilitate cost savings on AWS for Cloudera Data Engineering workloads.
- Data Warehouse enables you to create independent data warehouses and data marts that automatically scale to meet workload demands. This service provides isolated compute instances and automated optimization for each data warehouse and data mart, and helps you save costs while meeting SLAs. For more information, see the Cloudera website. Learn about managing costs and auto-scaling for Cloudera Data Warehouse on AWS.
- Operational Database in CDP provides a reliable and flexible foundation for scalable, high-performance applications. It delivers a real-time, always available, scalable database that serves traditional structured data alongside new, unstructured data within a unified operational and warehousing platform. For more information, see the Cloudera website.
- Machine Learning is a cloud-native machine learning platform that merges self-service data science and data engineering capabilities into a single, portable service within an enterprise data cloud. It enables scalable deployment of machine learning and artificial intelligence (AI) on data anywhere. For more information, see the Cloudera website.

CDP on AWS

The following diagram (adapted with permission from the Cloudera website) shows the high-level architecture of CDP on AWS. CDP implements its own security model to manage both accounts and data flow. These are integrated with IAM through the use of cross-account roles.



The CDP control plane resides in a Cloudera master account in its own VPC. Each customer account has its own sub-account and unique VPC. Cross-account IAM roles and SSL technologies route management traffic to and from the control plane to customer services that reside on internet-routable public subnets within each customer VPC. On the customer's VPC, the Cloudera Shared Data Experience (SDX) provides enterprise-strength security with unified governance and compliance so you can get insights from your data faster. SDX is a design philosophy incorporated into all Cloudera products. For more information about SDX and the CDP Public Cloud network architecture for AWS, see the Cloudera documentation.

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- [Amazon Relational Database Service (Amazon RDS)](#) helps you set up, operate, and scale a relational database in the AWS Cloud.
- [Amazon Simple Storage Service (Amazon S3)](#) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Automation and tooling

- For additional tooling, you can use Cloudera Backup Data Recovery (BDR), AWS Snowball, and AWS Snowmobile to help migrate data from on-premises CDH, HDP, and CDP to AWS-hosted CDP.
- For new deployments, we recommend that you use the AWS Partner Solution for CDP.

Epics

Prepare for migration

Task	Description	Skills required
Engage the Cloudera team.	Cloudera pursues a standardized engagement model with its customers and can work with your systems integrator (SI) to promote the same approach. Contact the Cloudera customer team so they can provide guidance and the necessary technical resources to get the project started. Contacting the Cloudera team ensures that all necessary teams can prepare for the migration as its date approaches. You can contact Cloudera Professional Services to move your	Migration lead

Task	Description	Skills required
	Cloudera deployment from pilot to production quickly, at lower cost, and with peak performance. For a complete list of offerings, see the Cloudera website.	
Create a CDP Public Cloud environment on AWS for your VPC.	Work with Cloudera Professional Services or your SI to plan and deploy CDP Public Cloud into a VPC on AWS.	Cloud architect, Cloudera SME

Task	Description	Skills required
Prioritize and assess workloads for migration.	Evaluate all your on-premises workloads to determine the workloads that are the easiest to migrate. Applications that aren't mission-critical are the best to move first, because they will have minimal impact on your customers. Save the mission-critical workloads for last, after you successfully migrate other workloads. **Note** Transient (CDP Data Engineering) workloads are easier to migrate than persistent (CDP Data Warehouse) workloads. It's also important to consider data volume and locations when migrating. Challenge s can include replicati ng data continuously from an on-premises environment to the cloud, and changing the data ingestion pipelines to import data directly to the cloud.	Migration lead

Task	Description	Skills required
Discuss CDH, HDP, CDP, and legacy application migration activities.	Consider and start planning for the following activitie s with Cloudera Workload Manager: • Data and workloads to copy to your AWS environment • Cloud-ready data • Noisy neighbors, which use up resources and create issues for other tenants • Elastic workloads • Small clusters with high operational overhead	Migration lead

Task	Description	Skills required
Complete the Cloudera Replication Manager requirements and recommendations.	Work with Cloudera Professional Services and your SI to prepare to migrate workloads to your CDP Public Cloud environment on AWS. Understanding the following requirements and recommendations can help you avoid common issues during and after you install the Replication Manager service. • Review Replication Manager supporting documents to confirm that you meet the environment and system requirements. For more information, see [Support matrix for CDP Public Cloud Replication Manager]() on the Cloudera website. • You don't need root access to the nodes on which the Replication Manager App and Data Lifecycle Manager (DLM) engine will be installed. • Install Apache Hive during the initial installation of Replication Manager, unless you are certain that you won't use Hive replication	Migration lead

Task	Description	Skills required
	in the future. If you decide to install Hive after creating HDFS replication policies in Replication Manager, you have to delete and then recreate all HDFS replicati on policies after you add Hive. • Clusters used in Replicati on Manager must have symmetrical configurations. Each cluster in a replicati on relationship must be configured exactly the same for security (Kerberos), user management (LDAP/AD), and Knox Proxy. Cluster services such as Hadoop Distribut ed File System (HDFS), Apache Hive, Apache Knox, Apache Ranger, and Apache Atlas can have different configurations for high availability (HA). For example, source and target clusters might have separate HA and non-HA configurations.	

Migrate CDP to AWS

Task	Description	Skills required
Migrate the first workload for dev/test environments by using Cloudera Workload Manager.	Your SI can help you migrate your first workload to the AWS Cloud. This should be an application that isn't customer-facing or mission-critical. Ideal candidates for dev/test migration are applications that have data that the cloud can easily ingest, such as CDP Data Engineering workloads. This is a transient workload that usually has fewer users accessing it, compared with a persistent workload such as a CDP Data Warehouse workload that could have many users who need uninterrupted access. Data Engineering workloads aren't persistent, which minimizes the business impact if something goes wrong. However, these jobs could be critical for productio n reporting, so prioritize low-impact Data Engineering workloads first.	Migration lead
Repeat migration steps as necessary.	Cloudera Workload Manager helps identify workloads that are best suited for the cloud. It provides metrics	Cloudera SME

Task	Description	Skills required
	such as cloud performance ratings, sizing/capacity plans for the target environment, and replication plans. The best candidates for migration are seasonal workloads, ad hoc reporting, and intermittent jobs that don't consume many resources. Cloudera Replication Manager moves data from on premises to the cloud, and from the cloud to on premises. Proactively optimize workloads, applications, performance, and infrastructure capacity for data warehousing, data engineering, and machine learning by using Workload Manager. For a complete guide on how to modernize a data warehouse, see the Cloudera website.	

Related resources

Cloudera documentation:

- Registering classic clusters with CDP, Cloudera Manager, and Replication Manager:
  - Management Console
  - Replication Manager hive replication
- Sentry replication
- Sentry permissions

- Data Hub cluster planning checklist
- Workload Manager architecture
- Replication Manager requirements
- Cloudera Data Platform Observability
- AWS requirements

AWS documentation:

- Cloud Data Migration

Resolve connection errors after migrating Microsoft SQL Server to the AWS Cloud

Premkumar Chelladurai, Amazon Web Services

Summary

After you migrate Microsoft SQL Server running on Windows Server 2008 R2, 2012, or 2012 R2 to Amazon Elastic Compute Cloud (Amazon EC2) instances on the Amazon Web Services (AWS) Cloud, the connection to SQL Server fails and the following errors appear:

- `[Microsoft][ODBC SQL Server Driver][DBNETLIB] General Network error`
- `ERROR [08S01] [Microsoft][SQL Native Client]Communication link failure. System.Data.SqlClient.SqlException: A transport-level error has occurred when sending the request to the server. (provider: TCP Provider, error: 0 - An existing connection was forcibly closed by the remote host.)`
- `TCP Provider: The semaphore timeout period has expired`

This pattern describes how you can resolve these errors by turning off the Windows Scalable Networking Pack (SNP) features at the operating system (OS) and network interface level for SQL Server running on Windows Server 2008 R2, 2012, or 2012 R2.

Prerequisites and limitations

Prerequisites

- Administrator privileges for Windows Server.
- If you used AWS Application Migration Service as your migration tool, you require one of the following Windows Server versions:
  - Windows Server 2008 R2 Service Pack 1, 2012, or 2012 R2
- If you used CloudEndure Migration as your migration tool, you require one of the following Windows Server versions:
  - Windows Server 2003 R2 Service Pack 3, 2008, 2008 R2 Service Pack 1, 2012, or 2012 R2

Tools

- [Amazon EC2](#) – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- [Windows Server](#) – Windows Server is a platform for building an infrastructure of connected applications, networks, and web services.

Epics

Turn off SNP features at the OS and elastic network interface levels

Task	Description	Skills required
Turn off SNP features at the OS level.	1. Sign in to Windows Server and open a command prompt as an administr ator. 2. Run the `netsh int tcp show global` command. 3. In the output, check if either `Receive-Side Scaling` or `Chimney Offload` is in `enabled` mode. If either of them is `enabled`, run the following commands: • `netsh int tcp set global chimney=d isabled` • `netsh int tcp set global rss=disab led`	AWS administrator, AWS systems administrator, Migration engineer, Cloud administrator

Task	Description	Skills required
Turn off SNP features at the elastic network interface level.	1. Choose **Start**, enter `ncpa.cpl`, and then press **Enter**. 2. Right-click **Elastic Network Adapter**. 3. In the popup menu, choose **Properties**. 4. In the **Ethernet Adapter Properties** window, choose **Configure**. 5. In the **Amazon Elastic Network Adapter Properties** popup window, choose the **Advanced** tab. 6. In the **Property** section, turn off all offloads and RSS.	AWS administrator, Cloud administrator, AWS systems administrator

Related resources

- How to troubleshoot advanced network performance features such as RSS and NetDMA

Restart the AWS Replication Agent automatically without disabling SELinux after rebooting a RHEL source server

Anil Kunapareddy, Venkatramana Chintha, and Shanmugam Shanker, Amazon Web Services

Summary

AWS Application Migration Service helps simplify, expedite, and automate the migration of your Red Hat Enterprise Linux (RHEL) workload to the Amazon Web Services (AWS) Cloud. To add source servers to Application Migration Service, you install the AWS Replication Agent on the servers.

Application Migration Service provides real-time, asynchronous, block-level replication. This means that you can continue normal IT operations during the entire replication process. These IT operations might require that you reboot or restart your RHEL source server during the migration. If this happens, the AWS Replication Agent will not restart automatically, and your data replication will stop. Typically, you can set Security-Enhanced Linux (SELinux) to **disabled** or **permissive** mode to automatically restart AWS Replication Agent. However, your organization's security policies might prohibit disabling SELinux, and you might also have to relabel your files.

This pattern describes how to automatically restart the AWS Replication Agent without turning off SELinux when your RHEL source server reboots or restarts during a migration.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An on-premises RHEL workload that you want to migrate to the AWS Cloud.
- Application Migration Service initialized from the Application Migration Service console. Initialization is required only the first time you use this service. For instructions, see the Application Migration Service documentation.
- An existing AWS Identity and Access Management (IAM) policy for Application Migration Service. For more information, see the Application Migration Service documentation.

Versions

- RHEL version 7 or later

Tools

AWS services

- AWS Application Migration Service is a highly automated lift-and-shift (rehost) solution that simplifies, expedites, and reduces the cost of migrating applications to AWS.

Linux commands

The following table provides a list of Linux commands that you will run on your RHEL source server. These are also described in the epics and stories for this pattern.

Command	Description
`#systemctl –version`	Identifies the system version.
`#systemctl list-units --type=service`	Lists all active services that are available on the RHEL server.
`#systemctl list-units --type=service | grep running`	Lists all services that are currently running on the RHEL server.
`#systemctl list-units --type=service | grep failed`	Lists all services that failed to load after the RHEL server rebooted or restarted.
`restorecon -Rv /etc/rc.d/init.d/aws-replication-service`	Changes the context to `aws-replication-service` .
`yum install policycoreutils*`	Installs the policy core utilities that are required for the operation of the SELinux system.
`ausearch -c "insmod" --raw | audit2allow -M my-modprobe`	Searches the audit log and creates a module for policies.
`semodule -i my-modprobe.pp`	Activates the policy.
`cat my-modprobe.te`	Displays the contents of the `my-modprobe.te` file.

`semodule -l | grep my-modprobe`	Checks whether the policy has been loaded to the SELinux module.

Epics

Install the AWS Replication Agent and reboot the RHEL source server

Task	Description	Skills required
Create an Application Migration Service user with an access key and a secret access key.	To install the AWS Replicati on Agent, you must create an Application Migration Service user with the required AWS credentials. For instructions, see the [Application Migration Service documentation]().	Migration engineer
Install the AWS Replication Agent.	1. Sign in to the AWS Management Console and open the AWS Migration Service console at https://console.aws.amazon.com/mgn/home. 2. Configure replication settings by following the instructions in the [Applicati on Migration Service documentation](). 3. Install the AWS Replicati on Agent by following the instructions in the [Application Migration Service documentation](). 4. On the **Source Servers** page, choose the RHEL source server, and then	Migration engineer

Task	Description	Skills required
	choose **Replication** to start the initial replication. For more information, see the [Application Migration Service documentation]().	
Restart or reboot the RHEL source server.	Restart or reboot your RHEL source server when its **Data replication status** displays **Stalled** on the [Migration dashboard]().	Migration engineer
Check data replication status.	Wait for one hour and then check the **Data replication status** again on the Migration dashboard. It should be in the **Healthy** state.	Migration engineer

Check AWS Replication Agent status on the RHEL source server

Task	Description	Skills required
Identify the system version.	Open the command line interface for your RHEL source server and run the following command to identify the system version: `#systemctl -version`	Migration engineer
List all active services.	To list all active services available on the RHEL server, run the command:	Migration engineer

Task	Description	Skills required
	#systemctl list-unit s --type=service	
List all running services.	To list all services that are currently running on the RHEL server, use the command: #systemctl list-unit s --type=service | grep running	Migration engineer
List all services that failed to load.	To list all services that failed to load after the RHEL server rebooted or restarted, run the command: #systemctl list-unit s --type=service | grep failed	Migration engineer

Create and run the SELinux module

Task	Description	Skills required
Change the security context.	In the command line interface for your RHEL source server, run the following command to change the security context to the AWS replicati on service: restorecon -Rv /etc/ rc.d/init.d/aws- replication-service	Migration engineer

Task	Description	Skills required
Install core utilities.	To install the core utilities required for the operation of the SELinux system and its policies, run the command: `yum install policycoreutils*`	Migration engineer
Search the audit log and create a module for policies.	Run the command: `ausearch -c "insmod" --raw | audit2allow -M my-modprobe`	Migration engineer
Display the contents of the my-modprobe-te file.	The `my-modprobe.te` file is generated by the **audit2allow** command. It includes the SELinux domains, policy source directory, and subdirectories, and specifies the access vector rules and transitions associated with the domains. To display the contents of the file, run the command: `cat my modprobe.te`	Migration engineer
Activate the policy.	To insert the module and make the policy package active, run the command: `semodule -i my-modprobe.pp`	Migration engineer

Task	Description	Skills required
Check whether the module has been loaded.	Run the command: `semodule -l | grep my-modprobe` After the SELinux module is loaded, you will no longer have to set SELinux to **disabled** or **permissive** mode during your migration.	Migration engineer
Reboot or restart the RHEL source server and verify the data replication status.	Open the AWS Migration Service console, navigate to **Data replication progress**, and then reboot or restart your RHEL source server. Data replication should now resume automatically after the RHEL source server reboots.	Migration engineer

Related resources

- Application Migration service documentation
- Technical training materials
- Troubleshooting AWS Replication Agent issues
- Application Migration Service policies

Re-architect

Topics

- Convert VARCHAR2(1) data type for Oracle to Boolean data type for Amazon Aurora PostgreSQL
- Create application users and roles in Aurora PostgreSQL-Compatible
- Emulate Oracle DR by using a PostgreSQL-compatible Aurora global database
- Implement SHA1 hashing for PII data when migrating from SQL Server to PostgreSQL
- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT
- Load BLOB files into TEXT by using file encoding in Aurora PostgreSQL-Compatible
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL in SSL mode by using AWS DMS
- Migrate Oracle SERIALLY_REUSABLE pragma packages into PostgreSQL
- Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible
- Migrate function-based indexes from Oracle to PostgreSQL
- Migrate Oracle native functions to PostgreSQL using extensions
- Migrate a Db2 database from Amazon EC2 to Aurora MySQL-Compatible by using AWS DMS
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS
- Migrate an on-premises ThoughtSpot Falcon database to Amazon Redshift
- Migrate an Oracle database to Amazon DynamoDB using AWS DMS
- Migrate an Oracle partitioned table to PostgreSQL by using AWS DMS
- Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL
- Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL-Compatible using AWS DMS and AWS SCT
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS
- Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate from Oracle to Amazon DocumentDB using AWS DMS

- Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT
- Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL by using an Oracle bystander and AWS DMS
- Migrate from Oracle Database to Amazon RDS for PostgreSQL by using Oracle GoldenGate
- Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT
- Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT
- Migrate data from an on-premises Oracle database to Aurora PostgreSQL
- Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents
- Migrate legacy applications from Oracle Pro*C to ECPG
- Migrate virtual generated columns from Oracle to PostgreSQL
- Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible
- Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL

Convert VARCHAR2(1) data type for Oracle to Boolean data type for Amazon Aurora PostgreSQL

Naresh Damera, Amazon Web Services

Summary

During a migration from Amazon Relational Database Service (Amazon RDS) for Oracle to Amazon Aurora PostgreSQL-Compatible Edition, you might encounter a data mismatch when validating the migration in AWS Database Migration Service (AWS DMS). To prevent this mismatch, you can convert VARCHAR2(1) data type to Boolean data type.

VARCHAR2 data type stores variable-length text strings, and VARCHAR2(1) indicates that the string is 1 character in length or 1 byte. For more information about VARCHAR2, see Oracle built-in data types (Oracle documentation).

In this pattern, in the sample source data table column, the VARCHAR2(1) data is either a **Y**, for *Yes*, or **N**, for *No*. This pattern includes instructions for using AWS DMS and AWS Schema Conversion Tool (AWS SCT) to convert this data type from the **Y** and **N** values in VARCHAR2(1) to **true** or **false** values in Boolean.

Intended audience

This pattern is recommended for those who have experience migrating Oracle databases to Aurora PostgreSQL-Compatible by using AWS DMS. As you complete the migration, adhere to the recommendations in Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL (AWS SCT documentation).

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Confirm that your environment is prepared for Aurora, including setting up credentials, permissions, and a security group. For more information, see Setting up your environment for Amazon Aurora (Aurora documentation).
- A source Amazon RDS for Oracle database that contains a table column with VARCHAR2(1) data.
- A target Amazon Aurora PostgreSQL-Compatible database instance. For more information, see Creating a database cluster and connecting to a database on an Aurora PostgreSQL database cluster (Aurora documentation).

Product versions

- Amazon RDS for Oracle version 12.1.0.2 or later.
- AWS DMS version 3.1.4 or later. For more information, see Using an Oracle database as a source for AWS DMS and Using a PostgreSQL database as a target for AWS DMS (AWS DMS documentation). We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.
- AWS Schema Conversion Tool (AWS SCT) version 1.0.632 or later. We recommend that you use the latest version of AWS SCT for the most comprehensive version and feature support.
- Aurora supports the PostgreSQL versions listed in Database Engine Versions for Aurora PostgreSQL-Compatible (Aurora documentation).

Architecture

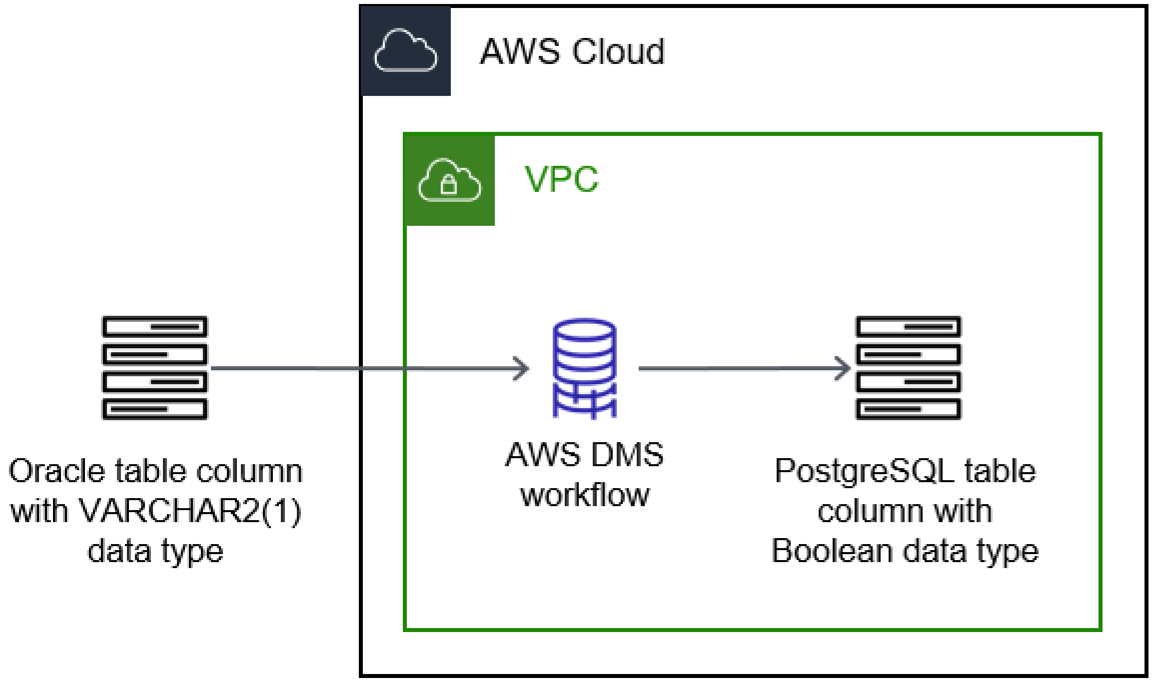
Source technology stack

Amazon RDS for Oracle database instance

Target technology stack

Amazon Aurora PostgreSQL-Compatible database instance

Source and target architecture



Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Amazon Relational Database Service (Amazon RDS) for Oracle helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format compatible with the target database.

Other services

- Oracle SQL Developer is an integrated development environment that simplifies the development and management of Oracle databases in both traditional and cloud-based deployments. In this pattern, you use this tool to connect to the Amazon RDS for Oracle database instance and query the data.

- pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects. In this pattern, you use this tool to connect to the Aurora database instance and query the data.

Epics

Prepare for the migration

Task	Description	Skills required
Create database migration report.	1. In AWS SCT, create a database migration assessment report. For more information, see Creating migration assessment reports. 2. Review and perform the action items in the migration assessment report. For more informati on, see Assessment report action items.	DBA, Developer
Disable foreign key constrain ts on the target database.	In PostgreSQL, foreign keys are implemented by using triggers. During the full load phase, AWS DMS loads each table one at a time. We strongly recommend that you disable foreign key constraints during a full load by using one of the following methods: • Temporarily disable all triggers from the instance, and finish the full load.	DBA, Developer

Task	Description	Skills required
	• Use the `session_replication_role` parameter in PostgreSQL. If disabling foreign key constraints is not feasible, create an AWS DMS migration task for the primary data that is specific to the parent table and child table.	

Disable the primary keys and unique keys on the target database.	Using the following commands, disable the primary keys and constraints on the target database. This helps improve the performance of the initial load task.
	
	`ALTER TABLE <table> DISABLE PRIMARY KEY;`
	
	`ALTER TABLE <table> DISABLE CONSTRAINT <constraint_name>;`
		DBA, Developer

Create the initial load task.	In AWS DMS, create the migration task for the initial load. For instructions, see Creating a task. For the migration method, choose **Migrate existing data**. This migration method is called `Full Load` in the API. Do not start this task yet.	DBA, Developer

Task	Description	Skills required
Edit task settings for the initial load task.	Edit the task settings to add data validation. These validation settings must be created in a JSON file. For instructions and examples, see Specifying task settings. Add the following validations: • To validate that the VARCHAR2(1) data is accurately converted to Boolean in the target database, add the code in *Data validation script* in the Additional information section of this pattern. The validation script converts the Boolean values of **1** to **Y** and **0** to **N** in the target table, and then it compares the values in the target table to the source table. To validate the rest of the data migration, enable data validation in the task. For more information, see Data validation task settings.	AWS administrator, DBA

Task	Description	Skills required
Create the ongoing replicati on task.	In AWS DMS, create the migration task that keeps the target database in sync with the source database. For instructions, see [Creating a task](). For the migration method, choose **Replicate data changes only**. Do not start this task yet.	DBA

Test the migration tasks

Task	Description	Skills required
Create sample data for testing.	In the source database, create a sample table with data for testing purposes.	Developer
Confirm there are no conflicti ng activities.	Use the `pg_stat_a ctivity` to check for any activity on the server that might affect the migration. For more information, see [The Statistics Collector]() (PostgreS QL documentation).	AWS administrator
Start the AWS DMS migration tasks.	In the AWS DMS console, on the **Dashboard** page, start the initial load and ongoing replication tasks that you created in the previous epic.	AWS administrator
Monitor the tasks and table load states.	During the migration, monitor the [task status]() and the [table states](). When the initial load	AWS administrator

Task	Description	Skills required
	task is complete, on the **Table statistics** tab: • The **Load state** should be **Table completed**. • The **Validation state** should be **Validated**.	
Verify the migration results.	Using pgAdmin, query the table on target database. A successful query indicates that the data was migrated successfully.	Developer
Add primary keys and foreign keys on the target database.	Create the primary key and foreign key on the target database. For more information, see [ALTER TABLE](#) (PostgreSQL website).	DBA
Clean up the test data.	On the source and target databases, clean up data that was created for unit testing.	Developer

Cut over

Task	Description	Skills required
Complete the migration.	Repeat the previous epic, *Test the migration tasks*, using the real source data. This migrates the data from the source to the target database.	Developer
Validate that the source and target databases are in sync.	Validate that the source and target databases are in sync.	Developer

Task	Description	Skills required
	For more information and instructions, see AWS DMS data validation.	
Stop the source database.	Stop the Amazon RDS for Oracle database. For instructi ons, see Stopping an Amazon RDS DB instance temporari ly. When you stop the source database, the initial load and ongoing replication tasks in AWS DMS are automatic ally stopped. No additiona l action is required to stop these tasks.	Developer

Related resources

AWS references

- Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT (AWS Prescriptive Guidance)
- Converting Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL (AWS SCT documentation)
- How AWS DMS Works (AWS DMS documentation)

Other references

- Boolean data type (PostgreSQL documentation)
- Oracle built-in data types (Oracle documentation)
- pgAdmin (pgAdmin website)
- SQL Developer (Oracle website)

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started With Amazon RDS
- Introduction to AWS DMS (Video)
- Understanding Amazon RDS (Video)

Additional information

Data validation script

The following data validation script converts **1** to **Y** and **0** to **N**. This helps the AWS DMS task successfully complete and pass the table validation.

```
{
"rule-type": "validation",
"rule-id": "5",
"rule-name": "5",
"rule-target": "column",
"object-locator": {
"schema-name": "ADMIN",
"table-name": "TEMP_CHRA_BOOL",
"column-name": "GRADE"
},
"rule-action": "override-validation-function",
"target-function": "case grade when '1' then 'Y' else 'N' end"
        }
```

The `case` statement in the script performs the validation. If validation fails, AWS DMS inserts a record in the **public.awsdms_validation_failures_v1** table on the target database instance. This record includes the table name, error time, and details about the mismatched values in the source and target tables.

If you do not add this data validation script to the AWS DMS task and the data is inserted in the target table, the AWS DMS task shows validation state as **Mismatched Records**.

During AWS SCT conversion, the AWS DMS migration task changes the data type of VARCHAR2(1) data type to Boolean and adds a primary key constraint on the `"NO"` column.

Create application users and roles in Aurora PostgreSQL-Compatible

Abhishek Verma, Amazon Web Services

Summary

When you migrate to Amazon Aurora PostgreSQL-Compatible Edition, the database users and roles that exist on the source database must be created in the Aurora PostgreSQL-Compatible database. You can create the users and roles in Aurora PostgreSQL-Compatible by using two different approaches:

- Use similar users and roles in the target as in the source database. In this approach, the data definition languages (DDLs) are extracted for users and roles from the source database. Then they are transformed and applied to the target Aurora PostgreSQL-Compatible database. For example, the blog post Use SQL to map users, roles, and grants from Oracle to PostgreSQL covers using extraction from an Oracle source database engine.
- Use standardized users and roles that are commonly used during development, administration, and for performing other related operations in the database. This includes read-only, read/write, development, administration, and deployment operations performed by the respective users.

This pattern contains the grants required for users and roles creation in Aurora PostgreSQL-Compatible required for the standardized users and roles approach. The user and role creation steps are aligned to the security policy of granting least privilege to the database users. The following table lists the users, their corresponding roles, and their details on the database.

Users	Roles	Purpose
`APP_read`	`APP_RO`	Used for read-only access on the schema APP
`APP_WRITE`	`APP_RW`	Used for the write and read operations on the schema APP
`APP_dev_user`	`APP_DEV`	Used for the developme nt purpose on the schema APP_DEV, with read-only access on the schema APP

`Admin_User`	`rds_superuser`	Used for performing administrator operations on the database
`APP`	`APP_DEP`	Used for creating the objects under the APP schema and for deployment of objects in the APP schema

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- A PostgreSQL database, Amazon Aurora PostgreSQL-Compatible Edition database, or Amazon Relational Database Service (Amazon RDS) for PostgreSQL database

Product versions

- All versions of PostgreSQL

Architecture

Source technology stack

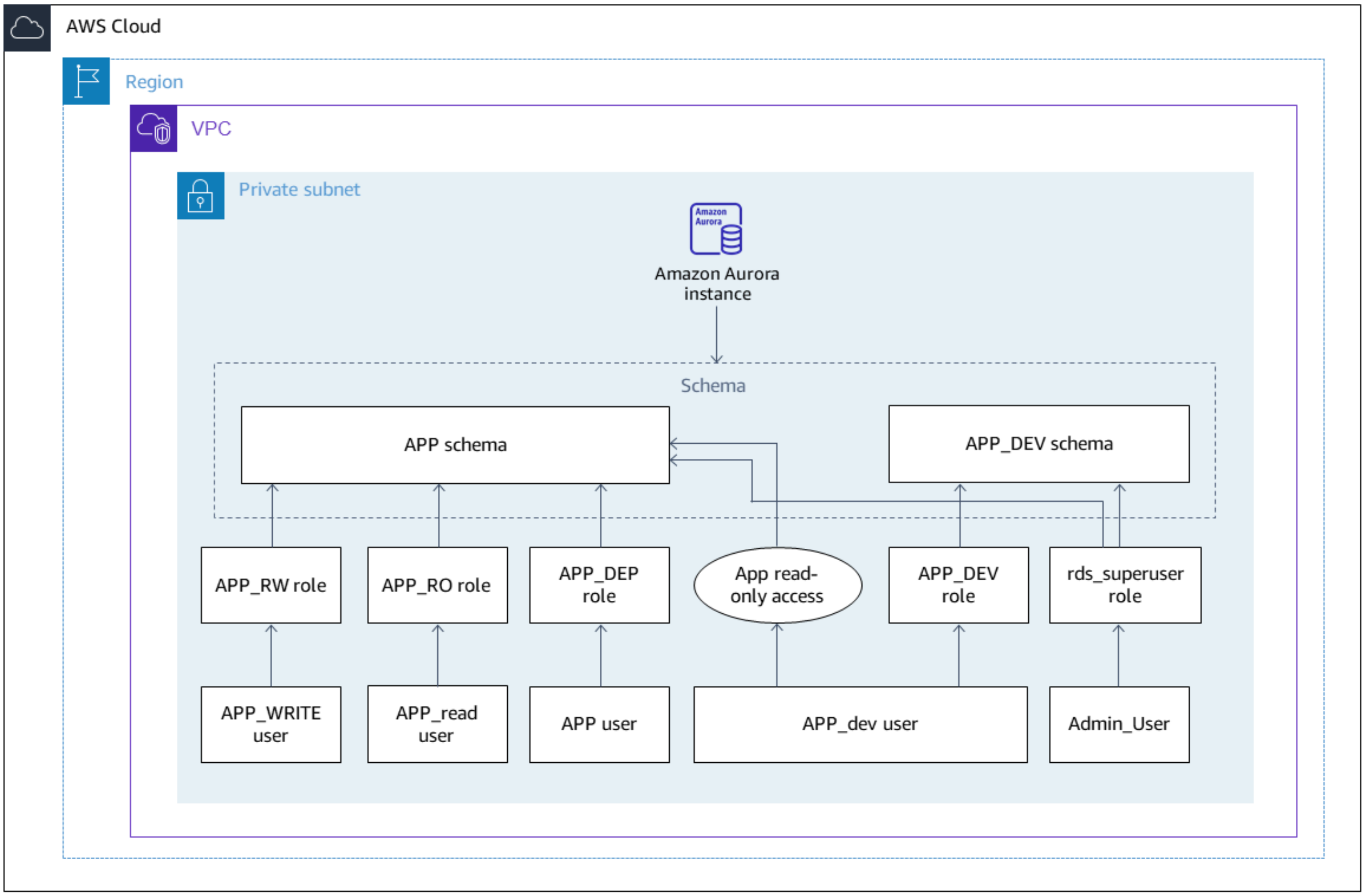
- Any database

Target technology stack

- Amazon Aurora PostgreSQL-Compatible

Target architecture

The following diagram shows user roles and the schema architecture in the Aurora PostgreSQL-Compatible database.



Automation and scale

This pattern contains the users, roles, and schema creation script, which you can run multiple times without any impact to existing users of the source or target database.

Tools

AWS services

- [Amazon Aurora PostgreSQL-Compatible Edition](https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.AuroraPostgreSQL.html) is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.

Other services

- psql is a terminal-based front-end tool that is installed with every PostgreSQL Database installation. It has a command line interface for running SQL, PL-PGSQL, and operating system commands.
- pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.

Epics

Create the users and roles

Task	Description	Skills required

Create the deployment user.	The deployment user APP will be used to create and modify the database objects during deployments. Use the following scripts to create the deployment user role APP_DEP in the schema APP. Validate access rights to make sure this user has only the privilege to create objects in the required schema APP.
	
	1. Connect to the admin user, and create the schema.
	`CREATE SCHEMA APP;`
	2. Create the user.
	`CREATE USER APP WITH PASSWORD <password > ;`
	3. Create the role.
	`CREATE ROLE APP_DEP ;`
		DBA

Task	Description	Skills required

	<pre>GRANT all on schema
	 APP to APP_DEP ;
	GRANT USAGE ON SCHEMA
	 APP to APP_DEP ;
	GRANT connect on
	 database <db_name>
	 to APP_DEP ;
	GRANT APP_DEP to APP;</pre>
	4. To test the privileges, connect to APP and create the tables.
	<pre>set search_path to
	 APP;
	SET
	CREATE TABLE test(id
	 integer) ;
	CREATE TABLE</pre>
	5. Check the privileges.
	<pre>select schemaname ,
	 tablename , tableowne
	r from pg_tables
	 where tablename like
	 'test' ;
	schemaname
	 | tablename |
	 tableowner
	APP |
	 test | APP</pre>	

Task	Description	Skills required

Create the read-only user.	The read-only user `APP_read` will be used for performing the read-only operation in the schema APP. Use the following scripts to create the read-only user. Validate access rights to make sure that this user has privilege only to read the objects in the schema APP and for automatically granting read access for any new objects created in the schema APP.
	
	1. Create the user `APP_read`.
	`create user APP_read ;`
	`alter user APP_read with password 'your_password' ;`
	2. Create the role.
	`CREATE ROLE APP_ro ;`
	`GRANT SELECT ON ALL TABLES IN SCHEMA  APP  TO  APP_RO   ;`
	`GRANT USAGE ON SCHEMA APP   TO APP_RO`
	`GRANT CONNECT ON DATABASE  testdb  TO APP_RO ;`
	`GRANT APP_RO TO APP_read;`
	3. To test the privileges, log in using the `APP_read` user.
		DBA

Task	Description	Skills required
	``` set search_path to APP ; create table test1(id integer) ; ERROR: permission denied for schema APP LINE 1: create table test1(id integer) ; insert into test values (34) ; ERROR: permission denied for table test SQL state: 42501 select from test no rows selected ```	

Task	Description	Skills required

Create the read/write user.	The read/write user `APP_WRITE` will be used to perform read and write operations on the schema APP. Use the following scripts to create the read/write user and grant it the `APP_RW` role. Validate access rights to make sure that this user has read and write privileges only on the objects in the schema APP and for automatically granting read and write access for any new object created in schema APP.
	
	1. Create the user.
	
	`CREATE USER APP_WRITE ; alter user APP_WRITE with password 'your_password' ;`
	
	2. Create the role.
	
	`CREATE ROLE APP_RW; GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA APP TO APP_RW ; GRANT CONNECT ON DATABASE postgres to APP_RW ; GRANT USAGE ON SCHEMA APP to APP_RW ;`	

Task	Description	Skills required
	``` ALTER DEFAULT PRIVILEGES IN SCHEMA APP GRANT SELECT, INSERT, UPDATE, DELETE ON TABLES TO APP_RW ; GRANT APP_RW to APP_WRITE ``` 3. To test the privileges, log in using the `APP_WRITE` user. ``` SET SEARCH_PATH to APP; CREATE TABLE test1(id integer) ; ERROR: permission denied for schema APP LINE 1: create table test1(id integer) ; SELECT * FROM test ; id ---- 12 INSERT INTO test values (31) ; INSERT 0 1 ```	

Task	Description	Skills required

Create the admin user.	The admin user `Admin_User` will be used to perform admin operations on the database. Examples of these operations are `CREATE ROLE` and `CREATE DATABASE`. `Admin_User` uses the built-in role `rds_superuser` to perform admin operations on the database. Use the following scripts to create and test the privilege for the admin user `Admin_User` in the database.
	
	1. Create the user and grant the role.
	
	`create user Admin_User WITH PASSWORD 'Your password'`
	`ALTER user Admin_user CREATEDB;`
	`ALTER user Admin_user CREATEROLE;`
	
	2. To test the privilege, log in from the `Admin_User` user.
	
	`SELECT * FROM APP.test ;`
	`id`
	`----`
	`31`
	`CREATE ROLE TEST ;`
		DBA

Task	Description	Skills required
	``` CREATE DATABASE test123 ; ```	

Task	Description	Skills required

Create the development user.	The development user `APP_dev_user` will have rights to create the objects in its local schema APP_DEV and read access in the schema APP. Use the following scripts to create and test the privileges of the user `APP_dev_user` in the database.
	
	1. Create the user.
	`CREATE USER APP1_dev_user with password 'your password';`
	2. Create the APP_DEV schema for the `App_dev_user` .
	`CREATE SCHEMA APP1_DEV ;`
	3. Create the APP_DEV role.
	`CREATE ROLE APP1_DEV ;`
	`GRANT APP1_RO to APP1_DEV ;`
	`GRANT SELECT ON ALL TABLES IN SCHEMA APP1_DEV to APP1_dev_user`
	`GRANT USAGE, CREATE ON SCHEMA APP1_DEV to APP1_DEV_USER`
	`GRANT APP1_DEV to APP1_DEV_USER ;`
		DBA

Task	Description	Skills required

	4. To test the privileges, log in from `APP_dev_user` .
	
	`CREATE TABLE APP1_dev.test1( id integer ) ;`
	`CREATE TABLE`
	`INSERT into APP1_dev.test1 ( select * from APP1.test );`
	`INSERT 0 1`
	`CREATE TABLE APP1.test4 ( id int) ;`
	`ERROR:   permission denied for schema APP1`
	`LINE 1: create table APP1.test4 ( id int) ;`	

Related resources

PostgreSQL documentation

- CREATE ROLE
- CREATE USER
- Predefined Roles

Additional information

PostgreSQL 14 enhancement

PostgreSQL 14 provides a set of predefined roles that give access to certain commonly needed, privileged capabilities and information. Administrators (including roles that have the `CREATE ROLE` privilege) can grant these roles or other roles in their environment to users, providing them with access to the specified capabilities and information.

Administrators can grant users access to these roles using the GRANT command. For example, to grant the pg_signal_backend role to Admin_User, you can run the following command.

```
GRANT pg_signal_backend TO Admin_User;
```

The pg_signal_backend role is intended to allow administrators to enable trusted, non-superuser roles to send signals to other backends. For more information, see PostgreSQL 14 enhancement.

Fine-tuning access

In some cases, it might be necessary to provide more granular access to the users (for example, table-based access or column-based access). In such cases, additional roles can be created to grant those privileges to the users. For more information, see PostgreSQL Grants.

Emulate Oracle DR by using a PostgreSQL-compatible Aurora global database

HariKrishna Boorgadda, Amazon Web Services

Summary

Best practices for Enterprise disaster recovery (DR) basically consist of designing and implementing fault-tolerant hardware and software systems that can survive a disaster (*business continuance*) and resume normal operations (*business resumption*), with minimal intervention and, ideally, with no data loss. Building fault-tolerant environments to satisfy Enterprise DR objectives can be expensive and time consuming and requires a strong commitment by the business.

Oracle Database provides three different approaches to DR that provide the highest level of data protection and availability compared to any other approach for protecting Oracle data.

- Oracle Zero Data Loss Recovery Appliance
- Oracle Active Data Guard
- Oracle GoldenGate

This pattern provides a way to emulate the Oracle GoldenGate DR by using an Amazon Aurora global database. The reference architecture uses Oracle GoldenGate for DR across three AWS Regions. The pattern walks through the replatforming of the source architecture to the cloud-native Aurora global database based on Amazon Aurora PostgreSQL–Compatible Edition.

Aurora global databases are designed for applications with a global footprint. A single Aurora database spans multiple AWS Regions with as many as five secondary Regions. Aurora global databases provide the following features:

- Physical storage-level replication
- Low-latency global reads
- Fast disaster recovery from Region-wide outages
- Fast cross-Region migrations
- Low replication lag across Regions
- Little-to-no performance impact on your database

For more information about Aurora global database features and advantages, see Using Amazon Aurora global databases. For more information about unplanned and managed failovers, see Using failover in an Amazon Aurora global database.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Java Database Connectivity (JDBC) PostgreSQL driver for application connectivity
- An Aurora global database based on Amazon Aurora PostgreSQL-Compatible Edition
- An Oracle Real Application Clusters (RAC) database migrated to the Aurora global database based on Aurora PostgreSQL–Compatible

Limitations of Aurora global databases

- Aurora global databases aren't available in all AWS Regions. For a list of supported Regions, see Aurora global databases with Aurora PostgreSQL.
- For information about features that aren't supported and other limitations of Aurora global databases, see the Limitations of Amazon Aurora global databases.

Product versions

- Amazon Aurora PostgreSQL–Compatible Edition version 10.14 or later

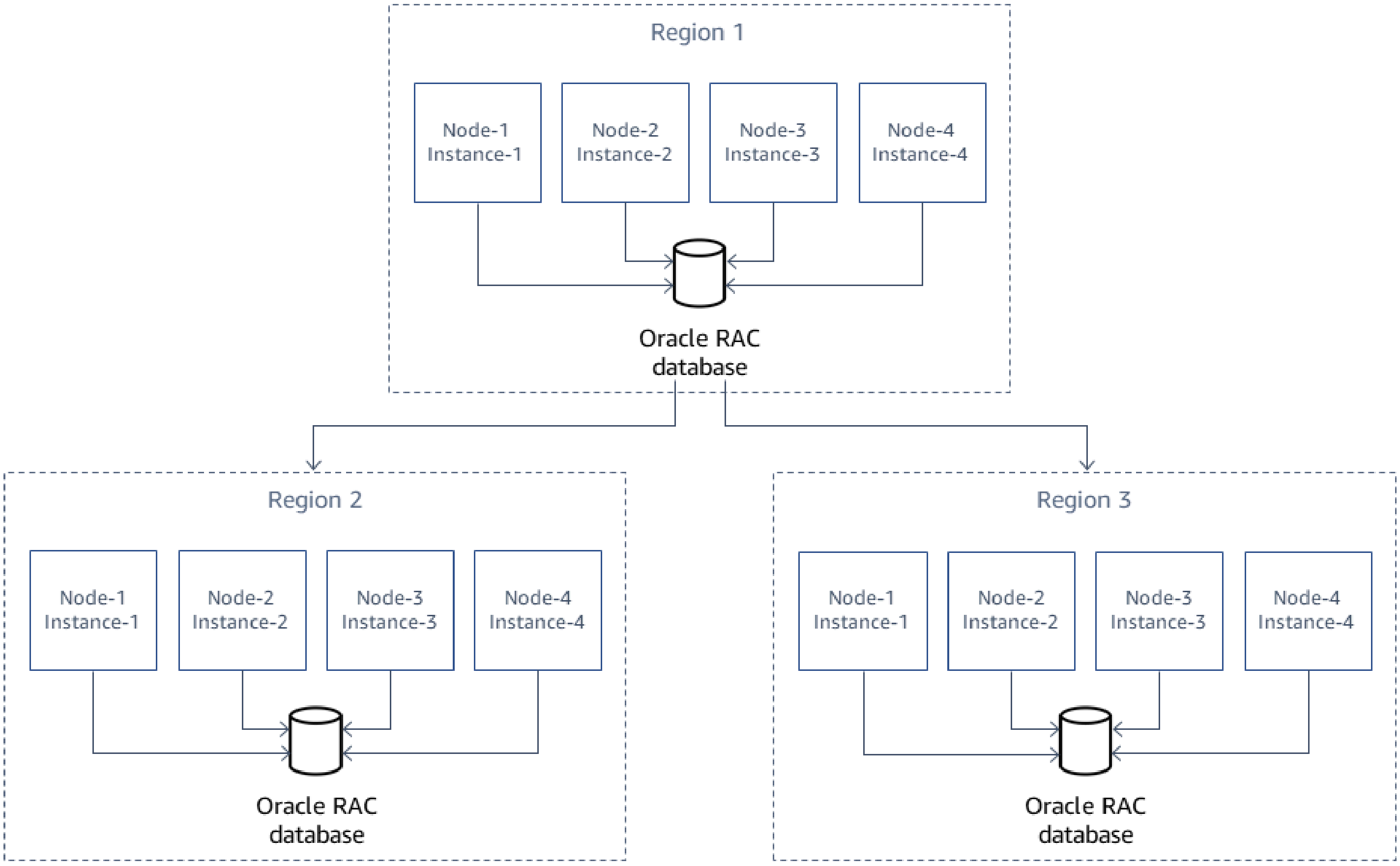
Architecture

Source technology stack

- Oracle RAC four-node database
- Oracle GoldenGate

Source architecture

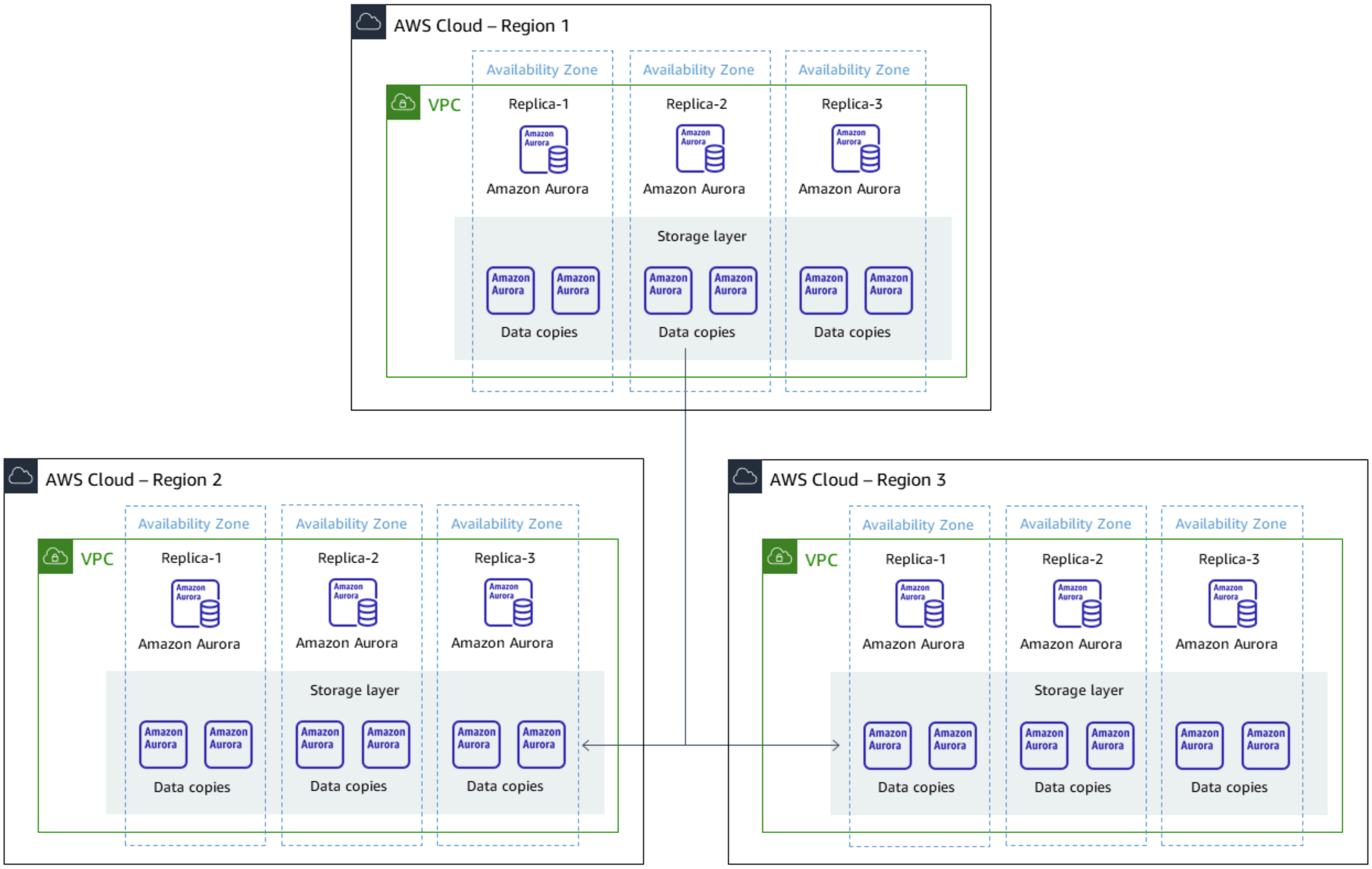
The following diagram shows three clusters with four-node Oracle RAC in different AWS Regions replicated using Oracle GoldenGate.



Target technology stack

- A three-cluster Amazon Aurora global database based on Aurora PostgreSQL–Compatible, with one cluster in the primary Region, two clusters in different secondary Regions

Target architecture



Tools

AWS services

- [Amazon Aurora PostgreSQL-Compatible Edition](#) is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- [Amazon Aurora global databases](#) span multiple AWS Regions, providing low latency global reads and fast recovery from the rare outage that might affect an entire AWS Region.

Epics

Add Regions with reader DB instances

Task	Description	Skills required
Attach one or more secondary Aurora clusters.	On the AWS Managemen t Console, choose Amazon	DBA

Task	Description	Skills required
	Aurora. Select the primary cluster, choose **Actions**, and choose **Add region** from the dropdown list.	
Select the instance class.	You can change the instance class of the secondary cluster. However, we recommend keeping it the same as the primary cluster instance class.	DBA
Add the third Region.	Repeat the steps in this epic to add a cluster in the third Region.	DBA

Fail over the Aurora global database

Task	Description	Skills required
Remove the primary cluster from the Aurora global database.	1. On the Databases page, choose the primary cluster. 2. Choose **Remove from Global** to fail over to a secondary cluster.	DBA
Reconfigure your application to divert write traffic to the newly promoted cluster.	Modify the endpoint in the application with that of the newly promoted cluster.	DBA
Stop issuing any write operations to the unavailable cluster.	Stop the application and any data manipulation language (DML) activity to the cluster that you removed.	DBA
Create a new Aurora global database.	Now you can create an Aurora global database with the	DBA

Task	Description	Skills required
	newly promoted cluster as the primary cluster.	

Start the primary cluster

Task	Description	Skills required
Select the primary cluster to be started from the global database.	On the Amazon Aurora console, in the Global Database setup, choose the primary cluster.	DBA
Start the cluster.	On the **Actions** dropdown list, choose **Start**. This process might take some time. Refresh the screen to see the status, or check the **Status** column for the current state of the cluster after the operation is completed.	DBA

Clean up the resources

Task	Description	Skills required
Delete the remaining secondary clusters.	After the failover pilot is completed, remove the secondary clusters from the global database.	DBA
Delete the primary cluster.	Remove the cluster.	DBA

Related resources

- Using Amazon Aurora global databases
- Aurora PostgreSQL Disaster Recovery solutions using Amazon Aurora Global Database (blog post)

Implement SHA1 hashing for PII data when migrating from SQL Server to PostgreSQL

Rajkumar Raghuwanshi and Jagadish Kantubugata, Amazon Web Services

Summary

This pattern describes how to implement Secure Hash Algorithm 1 (SHA1) hashing for email addresses when migrating from SQL Server to either Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible. An email address is an example of *personally identifiable information* (PII). PII is information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual.

This pattern covers the challenges of maintaining consistent hash values across different database collations and character encodings, and provides a solution using PostgreSQL functions and triggers. Although this pattern focuses on SHA1 hashing, it can be adapted for other hashing algorithms supported by PostgreSQL's `pgcrypto` module. Always consider the security implications of your hashing strategy and consult with security experts if handling sensitive data.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Source SQL Server database
- Target PostgreSQL database (Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible)
- PL/pgSQL coding expertise

Limitations

- This pattern requires database-level collation changes based on use cases.
- The performance impact on large datasets has not been evaluated.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS Services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

- Microsoft SQL Server 2012 or later

Architecture

Source technology stack

- SQL Server
- .NET Framework

Target technology stack

- PostgreSQL
- `pgcrypto` extension

Automation and scale

- Consider implementing the hashing function as a stored procedure for easier maintenance.
- For large datasets, evaluate performance and consider batch processing or indexing strategies.

Tools

AWS services

- [Amazon Aurora PostgreSQL-Compatible] is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- [AWS Database Migration Service (AWS DMS)] helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- [Amazon Relational Database Service Amazon RDS for PostgreSQL] helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.
- [AWS Schema Conversion Tool (AWS SCT)] supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.

Other tools

- pgAdmin is an open source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.
- SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure.

Best practices

- Use appropriate collation settings for handling special characters on the target database side.
- Test thoroughly with a variety of email addresses, including addresses with non-ASCII characters.
- Maintain consistency in uppercase and lowercase handling between the application and database layers.
- Benchmark performance of queries using the hashed values.

Epics

Analyze source hashing implementation

Task	Description	Skills required
Review SQL Server code.	To review SQL Server code that generates SHA1 hashes, do the following: • Analyze the existing SQL Server implementation of SHA1 hashing. • Identify the exact methods used for hash generation. • Document the input parameters and output format. • Review any data type conversions or transform ations.	Data engineer, DBA, App developer

Task	Description	Skills required
	• Examine collation settings and their impact.	
Document the hashing algorithm and data transform ations.	To document the exact hashing algorithm and data transformations, do the following: • Create detailed technical documentation of the hashing process. • Document the step-by-step transformation logic. • Specify input and output formats and data types. • Include edge cases and special character handling.	App developer, Data engineer, DBA

Create PostgreSQL hashing function

Task	Description	Skills required
Create `pgcrypto` extension.	To create the `pgcrypto` extension, use `pgAdmin/psql` to run the following command: `CREATE EXTENSION pgcrypto;`	DBA, Data engineer
Implement a PostgreSQL function.	Implement the following PostgreSQL function to replicate the SQL Server hashing logic. At a high	Data engineer, DBA, App developer

Task	Description	Skills required

	level, this function uses the following steps:
	
	1. Optionally converts the input to uppercase.
	2. Creates an SHA1 hash of the input.
	3. Takes the last 10 bytes (80 bits) of this hash.
	4. Converts these bytes to a 64-bit integer.
	
	`CREATE OR REPLACE FUNCTION utility.hex_to_bigint (`
	`     par_val character varying,`
	`     par_upper character varying DEFAULT 'lower'::character varying)`
	`RETURNS bigint`
	`LANGUAGE 'plpgsql'`
	`AS $BODY$`
	`DECLARE`
	`    retnumber bigint;`
	`    digest_bytes bytea;`
	`BEGIN`
	`    if lower(par_upper) = 'upper'`
	`    then`
	`        digest_bytes := digest(upper(par_val), 'sha1');`
	`    else`
	`        digest_bytes := digest((par_val), 'sha1');`	

Task	Description	Skills required

	``
	    end if;
	    retnumber := ('x'
	 || encode(substring(d
	igest_bytes, length(di
	gest_bytes)-10+1),
	 'hex'))::bit(64)::
	bigint;
	    RETURN retnumber;
	END;
	$BODY$;
	``	

Test the function.	To test the function, use sample data from SQL Server to verify matching hash values. Run the following command: ``` select 'alejandr o_rosalez@example. com' as Email, utility.hex_to_big int('alejandro_ros alez@example.com', 'upper') as HashValue; --OUTPUT /* email hashvalue "alejandro_rosale z@example.com" 451 397011176045063 */ ```	App developer, DBA, Data engineer

Implement triggers for automatic hashing

Task	Description	Skills required

Create triggers on relevant tables.	To create triggers on relevant tables to automatically generate hash values on insert or update, run the following command:
	`CREATE OR REPLACE FUNCTION update_email_hash() RETURNS TRIGGER AS $$ BEGIN NEW.email_hash = utility.hex_to_bigint(NEW.email, 'upper'); RETURN NEW; END; $$ LANGUAGE plpgsql;`
	`CREATE TRIGGER email_hash_trigger BEFORE INSERT OR UPDATE ON users FOR EACH ROW EXECUTE FUNCTION update_email_hash();`
		App developer, Data engineer, DBA

Migrate existing data

Task	Description	Skills required
Develop a migration script or use AWS DMS.	Develop a migration script or use AWS DMS to populate hash values for existing	Data engineer, App developer , DBA

Task	Description	Skills required
	data (including hash values stored as `BIGINT` in the source system.) Complete the following tasks: • Create migration scripts for data transfer with hash values. • Configure AWS DMS tasks with appropriate transformation rules. • Set up source and target endpoints in AWS DMS. • Implement error handling and logging mechanisms. • Design a batch processing strategy for large datasets. • Create validation queries for data verification.	
Use the new PostgreSQL hashing function.	To use the new PostgreSQL hashing function to ensure consistency, do the following: • Implement validation procedures to verify hash consistency. • Create comparison scripts between the source and target systems. • Set up automated testing for hash value verification. • Document any discrepancies and resolution steps.	App developer, DBA, DevOps engineer

Update application queries

Task	Description	Skills required
Identify application queries.	To identify application queries that use hashed values, do the following: • Analyze the application codebase for queries using hash values. • Review stored procedure s and functions that reference hash operations. • Document query performan ce metrics and execution plans. • Identify dependencies on hash-based lookups. • Map out affected applicati on components.	App developer, DBA, Data engineer
Modify queries.	If necessary, modify queries to use the new PostgreSQ L hashing function. Do the following: • Refactor existing queries to use the PostgreSQL hashing function. • Update stored procedures and functions. • Implement and test new query patterns. • Optimize modified queries for performance.	App developer, DBA, Data engineer

Test and validate

Task	Description	Skills required
Perform testing.	To perform thorough testing with a subset of production data, do the following: • Create a test plan for data subset validation. • Extract representative sample of production data. • Set up a test environment with proper configurations. • Execute data loading and transformation tests. • Perform volume and stress testing.	App developer, Data engineer, DBA
Validate that hash values match.	To validate that hash values match between SQL Server and PostgreSQL, do the following: • Develop comparison scripts for hash values. • Create validation reports for hash matching. • Implement automated verification procedures. • Document any discrepan cies found. • Analyze and resolve hash mismatches.	App developer, Data engineer, DBA

Task	Description	Skills required
Verify application functiona lity.	To verify application functionality by using the migrated data and the new hashing implementation, do the following: • Execute end-to-end application testing. • Validate all application features using hashed data. • Test application performan ce with the new implement ation. • Verify API integrations and dependencies.	App developer, DBA, Data engineer

Troubleshooting

Issue	Solution
Hash values don't match.	Verify character encodings and collation s between source and target. For more information, see Manage collation changes in PostgreSQL on Amazon Aurora and Amazon RDS (AWS Blog).

Related resources

AWS Blogs

- Manage collation changes in PostgreSQL on Amazon Aurora and Amazon RDS
- Migrate SQL Server to Amazon Aurora PostgreSQL using best practices and lessons learned from the field

Other resources

- PostgreSQL pgcrypto module (PostgreSQL documentation)
- PostgreSQL trigger functions (PostgreSQL documentation)
- SQL Server HASHBYTES function (Microsoft documentation)

Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT

Pinesh Singal, Amazon Web Services

Summary

Many migration strategies and approaches run in multiple phases that can last from a few weeks to several months. During this time, you can experience delays because of patching or upgrades in the source Oracle DB instances that you want to migrate to PostgreSQL DB instances. To avoid this situation, we recommend that you incrementally migrate the remaining Oracle database code to PostgreSQL database code.

This pattern provides an incremental migration strategy with no downtime for a multi-terabyte Oracle DB instance that has a high number of transactions performed after your initial migration and that must be migrated to a PostgreSQL database. You can use this pattern's step-by-step approach to incrementally migrate an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance without signing in to the Amazon Web Services (AWS) Management Console.

The pattern uses Oracle SQL Developer to find the differences between two schemas in the source Oracle database. You then use AWS Schema Conversion Tool (AWS SCT) to convert the Amazon RDS for Oracle database schema objects to Amazon RDS for PostgreSQL database schema objects. You can then run a Python script in the Windows Command Prompt to create AWS SCT objects for the incremental changes to the source database objects.

> **Note**
>
> Before you migrate your production workloads, we recommend that you run a proof of concept (PoC) for this pattern's approach in a testing or non-production environment.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing Amazon RDS for Oracle DB instance.
- An existing Amazon RDS for PostgreSQL DB instance.

- AWS SCT, installed and configured with JDBC drivers for Oracle and PostgreSQL database engines. For more information about this, see Installing AWS SCT and Installing the required database drivers in the AWS SCT documentation.
- Oracle SQL Developer, installed and configured. For more information about this, see the Oracle SQL Developer documentation.
- The `incremental-migration-sct-sql.zip` file (attached), downloaded to your local computer.

Limitations

- The minimum requirements for your source Amazon RDS for Oracle DB instance are:
  - Oracle versions 10.2 and later (for versions 10.x), 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c for the Enterprise, Standard, Standard One, and Standard Two editions
- The minimum requirements for your target Amazon RDS for PostgreSQL DB instance are:
  - PostgreSQL versions 9.4 and later (for versions 9.x), 10.x, and 11.x
- This pattern uses Oracle SQL Developer. Your results might vary if you use other tools to find and export schema differences.
- The SQL scripts generated by Oracle SQL Developer can raise transformation errors, which means that you need to perform a manual migration.
- If the AWS SCT source and target test connections fail, make sure that you configure the JDBC driver versions and inbound rules for the virtual private cloud (VPC) security group to accept incoming traffic.

Product versions

- Amazon RDS for Oracle DB instance version 12.1.0.2 (version 10.2 and later)
- Amazon RDS for PostgreSQL DB instance version 11.5 (version 9.4 and later)
- Oracle SQL Developer version 19.1 and later
- AWS SCT version 1.0.632 and later

Architecture

Source technology stack

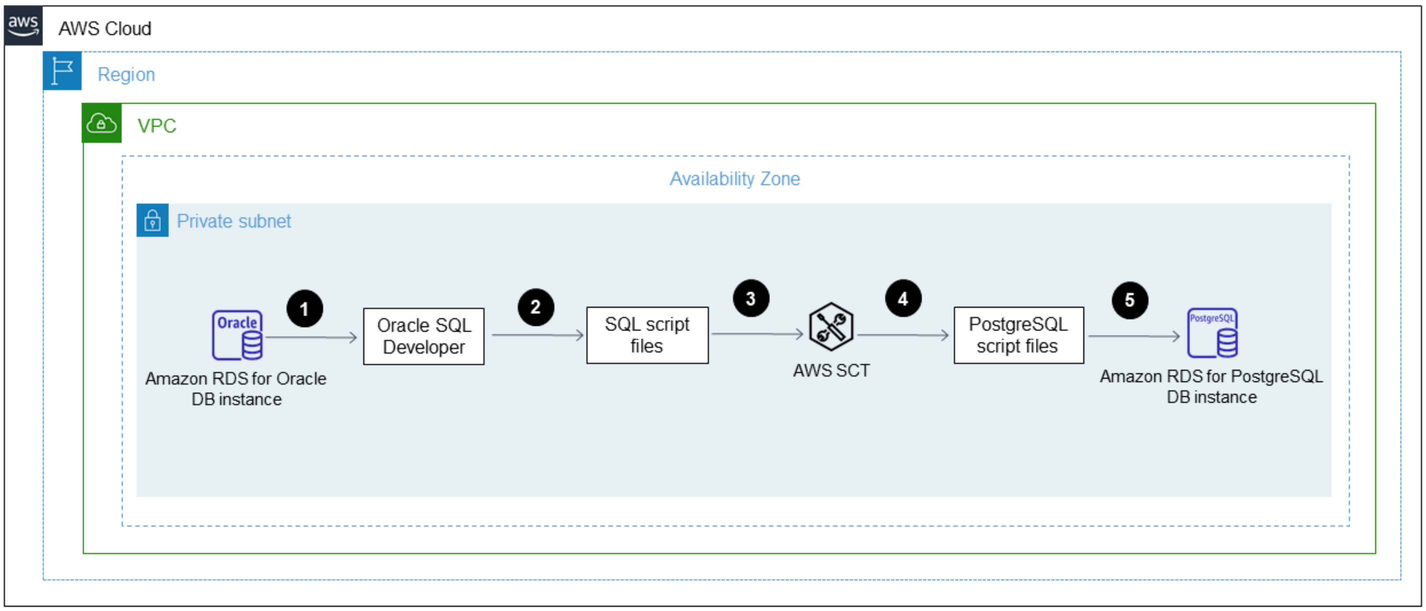
- Amazon RDS for Oracle DB instance

Target technology stack

- Amazon RDS for PostgreSQL DB instance

Source and target architecture

The following diagram shows the migration of an Amazon RDS for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance.



The diagram shows the following migration workflow:

1. Open Oracle SQL Developer and connect to the source and target databases.
2. Generate a diff report and then generate the SQL scripts file for the schema difference objects. For more information about diff reports, see Detailed diff reports in the Oracle documentation.
3. Configure AWS SCT and run the Python code.
4. The SQL scripts file converts from Oracle to PostgreSQL.
5. Run the SQL scripts file on the target PostgreSQL DB instance.

Automation and scale

You can automate this migration by adding additional parameters and security-related changes for multiple functionalities in a single program to your Python script.

Tools

- [AWS SCT](#) – AWS Schema Conversion Tool (AWS SCT) converts your existing database schema from one database engine to another.
- [Oracle SQL Developer](#) – Oracle SQL Developer is an integrated development environment (IDE) that simplifies the development and management of Oracle databases in both traditional and cloud-based deployments.

Code

The `incremental-migration-sct-sql.zip` file (attached) contains the complete source code for this pattern.

Epics

Create the SQL scripts file for the source database schema differences

Task	Description	Skills required
Run Database Diff in Oracle SQL Developer.	1. Sign in to your source Oracle DB instance, choose **Tools**, and then choose **Database Diff**. 2. Choose your source database in **Source Connection**. 3. Choose the updated or patched source database in **Destination Connection**. 4. Configure the remaining options according to your requirements, choose **Next**, and then choose **Finish** to generate the diff report.	DBA
Generate the SQL scripts file.	Choose **Generate Script** to generate the differences in the SQL files.	DBA

Task	Description	Skills required
	This generates the SQL scripts file that AWS SCT uses to convert your database from Oracle to PostgreSQL.	

Use the Python script to create the target DB objects in AWS SCT

Task	Description	Skills required
Configure AWS SCT with the Windows Command Prompt.	1. Copy the `AWSSchemaConversionToolBatch.jar` file from your pre-installed AWS SCT folder and paste it into your working directory. 2. Deploy the Python code from the `run_aws_sct_sql.py` file from the `incremental-migration-sct-sql.zip` folder (attached). This creates .xml files and .sct files in the `projects` directory with your source and target database environment configuration details. It also reads the SQL scripts file that you generated in Oracle SQL Developer. Finally, it creates .sql file objects in the `output` directory.	DBA

Task	Description	Skills required

	3. Configure the source and target environment configuration details in the `database_migration.txt` file by using the following format:
	
	`#source_vendor,source_hostname,source_dbname,source_user,source_pwd,source_schema,source_port,source_sid,target_vendor,target_hostname,target_user,target_pwd,target_dbname,target_port`
	
	`ORACLE,myoracledb.cokmvis0v46q.us-east-1.rds.amazonaws.com,ORCL,orcl,orcl1234,orcl,1521,ORCL,POSTGRESQL,mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com,pguser,pgpassword,pgdb,5432`
	
	4. Modify the AWS SCT configuration parameters according to your requirements and then copy the SQL scripts file into your working directory in the `input` subdirectory.	

Task	Description	Skills required
Run the Python script.	1. Run the Python script by using the following command: `$ python run_aws_sct_sql.py database_migration .txt` 2. This creates the DB objects SQL file. Non-converted codes with transformation errors can be manually converted.	DBA
Create the objects in Amazon RDS for PostgreSQL	Run the SQL files and create objects in your Amazon RDS for PostgreSQL DB instance.	DBA

Related resources

- Oracle on Amazon RDS
- PostgreSQL on Amazon RDS
- Using the AWS SCT user interface
- Using Oracle as a source for AWS SCT

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Load BLOB files into TEXT by using file encoding in Aurora PostgreSQL-Compatible

Bhanu Ganesh Gudivada and Jeevan Shetty, Amazon Web Services

Summary

Often during migration, there are cases where you have to process unstructured and structured data that is loaded from files on a local file system. The data might also be in a character set that differs from the database character set.

These files hold the following types of data:

- **Metadata –** This data describes the file structure.
- **Semi-structured data –** These are textual strings in a specific format, such as JSON or XML. You might be able to make assertions about such data, such as "will always start with '<' " or "does not contain any newline characters."
- **Full text –** This data usually contains all types of characters, including newline and quote characters. It might also consist of multibyte characters in UTF-8.
- **Binary data –** This data might contain bytes or combinations of bytes including, nulls and end-of-file markers.

Loading a mixture of these types of data can be a challenge.

The pattern can be used with on-premises Oracle databases , Oracle databases that are on Amazon Elastic Compute Cloud (Amazon EC2) instances on the Amazon Web Services (AWS) Cloud, and Amazon Relational Database Service (Amazon RDS) for Oracle databases. As an example, this pattern is using Amazon Aurora PostgreSQL-Compatible Edition.

In Oracle Database, with the help of a `BFILE` (binary file) pointer, the `DBMS_LOB` package, and Oracle system functions, you can load from file and convert to CLOB with character encoding. Because PostgreSQL does not support the BLOB data type when migrating to an Amazon Aurora PostgreSQL-Compatible Edition database, these functions must be converted to PostgreSQL-compatible scripts.

This pattern provides two approaches for loading a file into a single database column in an Amazon Aurora PostgreSQL-Compatible database:

- Approach 1 – You import data from your Amazon Simple Storage Service (Amazon S3) bucket by using the `table_import_from_s3` function of the `aws_s3` extension with the encode option.
- Approach 2 – You encode to hexadecimal outside of the database, and then you decode to view `TEXT` inside the database.

We recommend using Approach 1 because Aurora PostgreSQL-Compatible has direct integration with the `aws_s3` extension.

This pattern uses the example of loading a flat file that contains an email template, which has multibyte characters and distinct formatting, into an Amazon Aurora PostgreSQL-Compatible database.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS instance or an Aurora PostgreSQL-Compatible instance
- A basic understanding of SQL and Relational Database Management System (RDBMS)
- An Amazon Simple Storage Service (Amazon S3) bucket.
- Knowledge of system functions in Oracle and PostgreSQL
- RPM Package HexDump-XXD-0.1.1 (included with Amazon Linux 2)

> **Note**
>
> Amazon Linux 2 is nearing end of support. For more information, see the [Amazon Linux 2 FAQs](#).

Limitations

- For the `TEXT` data type, the longest possible character string that can be stored is about 1 GB.

Product versions

- Aurora supports the PostgreSQL versions listed in [Amazon Aurora PostgreSQL updates](#).

Architecture

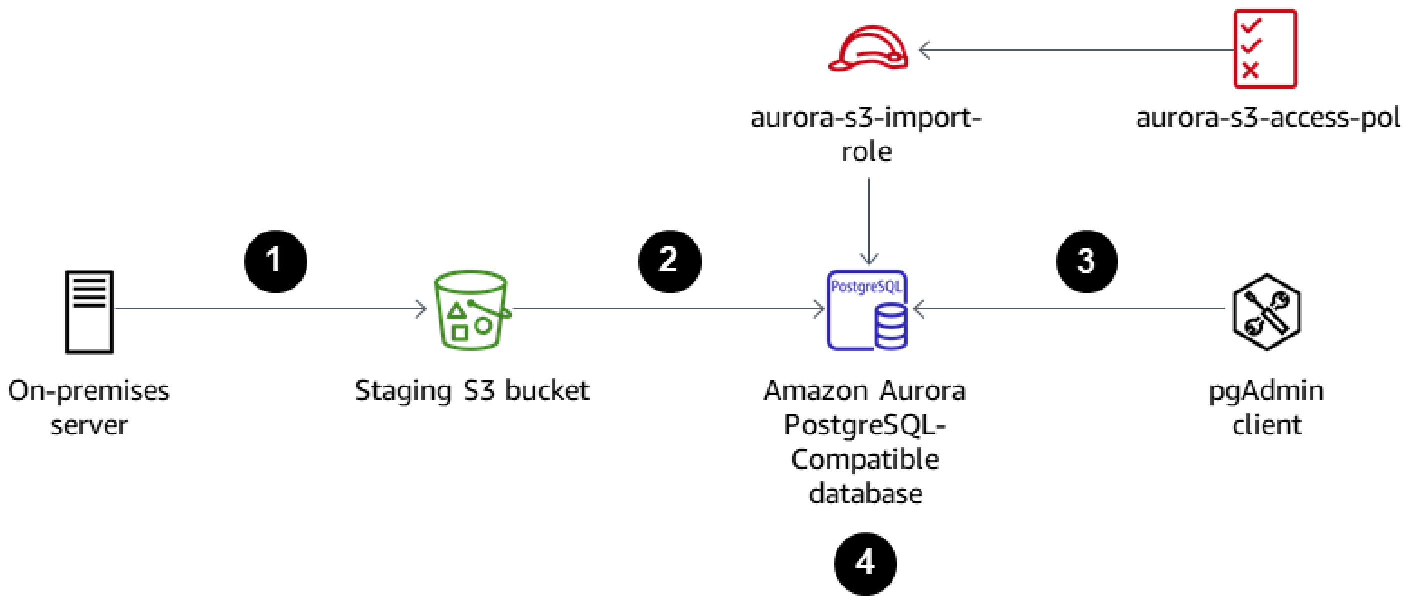
Target technology stack

- Aurora PostgreSQL-Compatible

Target architecture

Approach 1 – Using aws_s3.table_import_from_s3

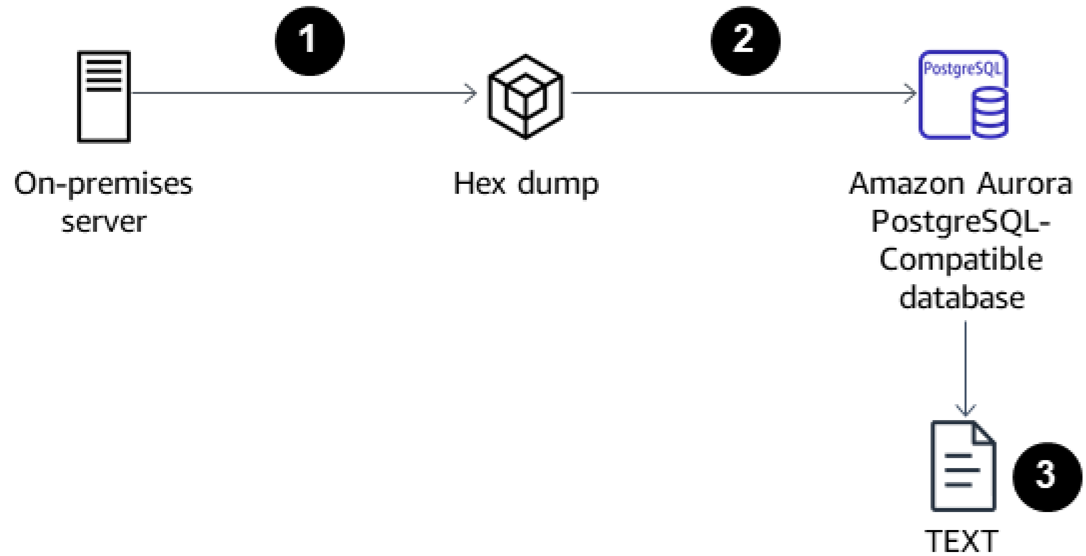
From an on-premises server, a file containing an email template with multibyte characters and custom formatting is transferred to Amazon S3. The custom database function provided by this pattern uses the `aws_s3.table_import_from_s3` function with `file_encoding` to load files into the database and return query results as the TEXT data type.



1. Files are transferred to the staging S3 bucket.
2. Files are uploaded to the Amazon Aurora PostgreSQL-Compatible database.
3. Using the pgAdmin client, the custom function `load_file_into_clob` is deployed to the Aurora database.
4. The custom function internally uses `table_import_from_s3` with file_encoding. The output from the function is obtained by using `array_to_string` and `array_agg` as TEXT output.

Approach 2 – Encoding to hexadecimal outside of the database and decoding to view TEXT inside the database

A file from an on-premises server or a local file system is converted into a hex dump. Then the file is imported into PostgreSQL as a TEXT field.



1. Convert the file to a hex dump in the command line by using the `xxd -p` option.
2. Upload the hex dump files into Aurora PostgreSQL-Compatible by using the `\copy` option, and then decode the hex dump files to binary.
3. Encode the binary data to return as TEXT.

Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.

Other tools

- pgAdmin4 is an open source administration and development platform for PostgreSQL. pgAdmin4 can be used on Linux, Unix, mac OS, and Windows to manage PostgreSQL.

Epics

Approach 1: Import data from Amazon S3 to Aurora PostgreSQL-Compatible

Task	Description	Skills required
Launch an EC2 instance.	For instructions on launching an instance, see [Launch your instance]().	DBA
Install the PostgreSQL client pgAdmin tool.	Download and install [pgAdmin]().	DBA

Create an IAM policy.	Create an AWS Identity and Access Management (IAM) policy named `aurora-s3-access-pol` that grants access to the S3 bucket where the files will be stored. Use the following code, replacing <bucket-name> with the name of your S3 bucket.
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Effect": "Allow",`
	`            "Action": [`
	`                "s3:GetObject",`
	`                "s3:AbortMultipartUpload",`
	`                "s3:DeleteObject",`
	`                "s3:ListMultipartUploadParts",`
		DBA

Task	Description	Skills required

	<pre>"s3:PutObject",
	
	"s3:ListBucket"
	            ],
	            "Resource":
	[
	
	"arn:aws:s3:::<buc
	ket-name>/*",
	
	"arn:aws:s3:::<buc
	ket-name>"
	            ]
	        }
	    ]
	}</pre>	

Task	Description	Skills required
Create an IAM role for object import from Amazon S3 to Aurora PostgreSQL-Compatible.	Use the following code to create an IAM role named `aurora-s3-import-role` with the [AssumeRole](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html) trust relationship. `AssumeRole` allows Aurora to access other AWS services on your behalf. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow","Principal": { "Service": "rds.amazonaws.com" },"Action": "sts:AssumeRole" } ] }`	DBA

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Associate the IAM role to the cluster.</td><td>To associate the IAM role with the Aurora PostgreSQL-Compatible database cluster, run the following AWS CLI command. Change <code><Account-ID></code> to the ID of the AWS account that hosts the Aurora PostgreSQL-Compatible database. This enables the Aurora PostgreSQL-Compatible database to access the S3 bucket.

<pre><code>aws rds add-role-to-db-cluster --db-cluster-identifier aurora-postgres-cl
--feature-name s3Import
 --role-arn arn:aws:iam::<Account-ID>:role/aurora-s3-import-role</code></pre></td><td>DBA</td></tr>
<tr><td>Upload the example to Amazon S3.</td><td>1. In the Additional information section of this pattern, copy the email template code into a file named <code>salary.event.notification.email.vm</code>.
2. Upload to the file to the S3 bucket.</td><td>DBA, App owner</td></tr>
</table>

Task	Description	Skills required
Deploy the custom function.	1. From the *Additional information* section, copy the custom function `load_file_into_clob` SQL file content into a temporary table. 2. Log in to the Aurora PostgreSQL-Compatible database and deploy it into the database schema by using the pgAdmin client.	App owner, DBA

Task	Description	Skills required

Run the custom function for importing the data into the database.	Run the following SQL command, replacing the items in angle brackets with the appropriate values.
	`select load_file _into_clob('aws-s3 -import-test'::tex t,'us-west-1'::tex t,'employee.salary .event.notificatio n.email.vm'::text);`
	Replace the items in angle brackets with the appropria te values, as shown in the following example, before running the command.
	`Select load_file _into_clob('aws-s3 -import-test'::tex t,'us-west-1'::tex t,'employee.salary .event.notificatio n.email.vm'::text);`
	The command loads the file from Amazon S3 and returns the output as TEXT.
		App owner, DBA

Approach 2: Convert the template file into a hex dump in a local Linux system

Task	Description	Skills required
Convert the template file into a hex dump.	**Note** The Hexdump utility displays the contents of binary files in hexadecimal, decimal, octal, or ASCII. The `hexdump` command is part of the `util-linux` package and comes pre-installed in Linux distributions. The Hexdump RPM package is part of Amazon Linux 2 as well. (: Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs.) To convert the file contents into a hex dump, run the following shell command. `xxd -p </path/file.vm> | tr -d '\n' > </path/file.hex>` Replace the path and file with the appropriate values,	DBA

Task	Description	Skills required
	as shown in the following example. ``` xxd -p employee.salary.event.notification.email.vm | tr -d '\n' > employee.salary.event.notification.email.vm.hex ```	

Task	Description	Skills required

Load the hexdump file into the database schema.	Use the following commands to load the hexdump file into the Aurora PostgreSQL-Compatible database.
	
	1. Log in to Aurora PostgreSQL database, and create a new table called `email_template_hex` .
	`CREATE TABLE email_template_hex(hex_data TEXT);`
	2. Load the files from the local file system into the DB schema by using the following command.
	`\copy email_template_hex FROM '/path/file.hex';`
	Replace the path with the location on your local file system.
	`\copy email_template_hex FROM '/tmp/employee.salary.event.notification.email.vm.hex';`
	3. Create one more table called `email_template_bytea` .
		DBA

Task	Description	Skills required
	``` CREATE TABLE email_template_bytea(hex_data bytea); ``` 4. Insert the data from `email_template_hex` into `email_template_bytea` . ``` INSERT INTO email_template_bytea (hex_data) (SELECT decode(hex_data, 'hex') FROM email_template_hex limit 1); ``` 5. To return hex bytea code as TEXT data, run the following command. ``` SELECT encode(hex_data::bytea, 'escape') FROM email_template_bytea; ```	

Related resources

References

- Using a PostgreSQL database as a target for AWS Database Migration Service
- Oracle Database 19c to Amazon Aurora with PostgreSQL Compatibility (12.4) Migration Playbook
- Creating IAM policies

- [Associating an IAM role with an Amazon Aurora MySQL DB cluster]
- [pgAdmin]

Tutorials

- [Getting Started with Amazon RDS]
- [Migrate from Oracle to Amazon Aurora]

Additional information

load_file_into_clob custom function

```
CREATE OR REPLACE FUNCTION load_file_into_clob(
    s3_bucket_name text,
    s3_bucket_region text,
    file_name text,
    file_delimiter character DEFAULT '&'::bpchar,
    file_encoding text DEFAULT 'UTF8'::text)
    RETURNS text
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
DECLARE
    blob_data BYTEA;
    clob_data TEXT;
    l_table_name CHARACTER VARYING(50) := 'file_upload_hex';
    l_column_name CHARACTER VARYING(50) := 'template';
    l_return_text TEXT;
    l_option_text CHARACTER VARYING(150);
    l_sql_stmt CHARACTER VARYING(500);

BEGIN

    EXECUTE format ('CREATE TEMPORARY TABLE %I (%I text, id_serial serial)',
 l_table_name, l_column_name);

    l_sql_stmt := 'select ''(format text, delimiter ''''' || file_delimiter || ''''',
 encoding ''''' || file_encoding ||  ''''')'' ';

    EXECUTE FORMAT(l_sql_stmt)
```

```
    INTO l_option_text;

    EXECUTE FORMAT('SELECT aws_s3.table_import_from_s3($1,$2,$6,
 aws_commons.create_s3_uri($3,$4,$5))')
    INTO l_return_text
    USING l_table_name, l_column_name, s3_bucket_name,
 file_name,s3_bucket_region,l_option_text;

    EXECUTE format('select array_to_string(array_agg(%I order by id_serial),E''\n'')
 from %I', l_column_name, l_table_name)
    INTO clob_data;

    drop table file_upload_hex;

    RETURN clob_data;
END;
$BODY$;
```

Email template

```
#####################################################################################
##
 ##
##    johndoe Template Type: email
 ##
##    File: johndoe.salary.event.notification.email.vm
 ##
##    Author: Aimée Étienne    Date 1/10/2021
      ##
##  Purpose: Email template used by EmplmanagerEJB to inform a johndoe they         ##
##        have been given access to a salary event
 ##
##    Template Attributes:
 ##
##        invitedUser - PersonDetails object for the invited user
  ##
##        salaryEvent - OfferDetails object for the event the user was given access
 ##
##        buyercollege - CompDetails object for the college owning the salary event
 ##
##        salaryCoordinator - PersonDetails of the salary coordinator for the event
 ##
```

```
##        idp - Identity Provider of the email recipient
 ##
##        httpWebRoot - HTTP address of the server
 ##
##
 ##
#####################################################################################

$!invitedUser.firstname $!invitedUser.lastname,

Ce courriel confirme que vous avez ete invite par $!salaryCoordinator.firstname $!
salaryCoordinator.lastname de $buyercollege.collegeName a participer a l'evenement
 "$salaryEvent.offeringtitle" sur johndoeMaster Sourcing Intelligence.

Votre nom d'utilisateur est $!invitedUser.username

Veuillez suivre le lien ci-dessous pour acceder a l'evenement.

${httpWebRoot}/myDashboard.do?idp=$!{idp}

Si vous avez oublie votre mot de passe, utilisez le lien "Mot de passe oublie" situe
 sur l'ecran de connexion et entrez votre nom d'utilisateur ci-dessus.

Si vous avez des questions ou des preoccupations, nous vous invitons a
 communiquer avec le coordonnateur de l'evenement $!salaryCoordinator.firstname $!
salaryCoordinator.lastname au ${salaryCoordinator.workphone}.

*******

johndoeMaster Sourcing Intelligence est une plateforme de soumission en ligne pour les
 equipements, les materiaux et les services.

Si vous avez des difficultes ou des questions, envoyez un courriel a
 support@johndoeMaster.com pour obtenir de l'aide.
```

Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation

Pinesh Singal, Amazon Web Services

Summary

This pattern shows how to migrate a multi-terabyte Amazon Relational Database Service (Amazon RDS) for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance by using the AWS Command Line Interface (AWS CLI). The approach provides minimal downtime and doesn't require signing in to the AWS Management Console.

This pattern helps avoid manual configurations and individual migrations by using the AWS Schema Conversion Tool (AWS SCT) and AWS Database Migration Service (AWS DMS) consoles. The solution sets up a one-time configuration for multiple databases and performs the migrations by using AWS SCT and AWS DMS in the AWS CLI.

The pattern uses AWS SCT to convert database schema objects from Amazon RDS for Oracle to Amazon RDS for PostgreSQL, and then uses AWS DMS to migrate the data. Using Python scripts in AWS CLI, you create AWS SCT objects and AWS DMS tasks with an AWS CloudFormation template.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing Amazon RDS for Oracle DB instance.
- An existing Amazon RDS for PostgreSQL DB instance.
- An Amazon Elastic Compute Cloud (Amazon EC2) instance or local machine with Windows or Linux OS for running scripts.
- An understanding of the following AWS DMS migration task types: `full-load`, `cdc`, `full-load-and-cdc`. For more information, see Creating a task in the AWS DMS documentation.
- AWS SCT, installed and configured with Java Database Connectivity (JDBC) drivers for Oracle and PostgreSQL database engines. For more information, see Installing and configuring AWS SCT in the AWS SCT documentation.
- The `AWSSchemaConversionToolBatch.jar` file from the installed AWS SCT folder, copied to your working directory.

- The `cli-sct-dms-cft.zip` file (attached), downloaded and extracted in your working directory.
- The most recent AWS DMS replication instance engine version. For more information, see [How do I create an AWS DMS replication instance](#) in the AWS Support documentation and [AWS DMS release notes](#).
- AWS CLI version 2, installed and configured with your access key ID, secret access key, and default AWS Region name for the EC2 instance or OS where the scripts are run. For more information, see [Installing or updating to the latest version of the AWS CLI](#) and [Configuring settings for the AWS CLI](#) in the AWS CLI documentation.
- Familiarity with AWS CloudFormation templates. For more information, see [How AWS CloudFormation works](#) in the AWS CloudFormation documentation.
- Python version 3, installed and configured on the EC2 instance or OS where the scripts are run. For more information, see the [Python documentation](#).

Limitations

- The minimum requirements for your source Amazon RDS for Oracle DB instance are:
  - Oracle versions 12c (12.1.0.2, 12.2.0.1), 18c (18.0.0.0), and 19c (19.0.0.0) for the Enterprise, Standard, Standard One, and Standard Two editions.
  - Although Amazon RDS supports Oracle 18c (18.0.0.0), this version is on a deprecation path because Oracle no longer provides patches for 18c after the end-of-support date. For more information, see [Amazon RDS for Oracle](#) in the Amazon RDS documentation.
  - Amazon RDS for Oracle 11g is no longer supported.
- The minimum requirements for your target Amazon RDS for PostgreSQL DB instance are:
  - PostgreSQL versions 9 (9.5 and 9.6), 10.x, 11.x, 12.x, and 13.x

Product versions

- Amazon RDS for Oracle DB instance version 12.1.0.2 and later
- Amazon RDS for PostgreSQL DB instance version 11.5 and later
- AWS CLI version 2
- The latest version of AWS SCT
- The latest version of Python 3

Architecture

Source technology stack

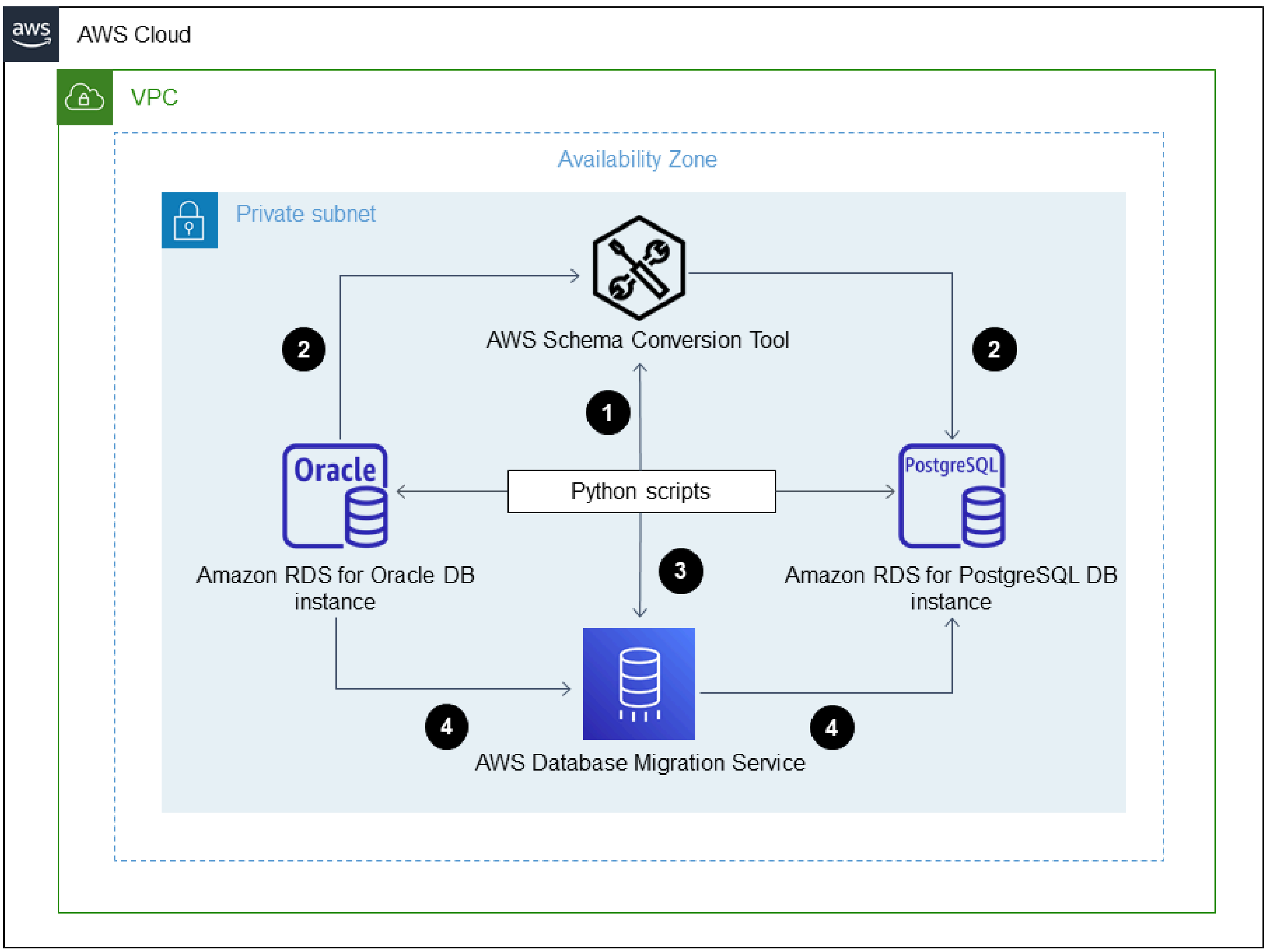
- Amazon RDS for Oracle

Target technology stack

- Amazon RDS for PostgreSQL

Source and target architecture

The following diagram shows the migration of an Amazon RDS for Oracle DB instance to an Amazon RDS for PostgreSQL DB instance using AWS DMS and Python scripts.



The diagram shows the following migration workflow:

1. The Python script uses AWS SCT to connect to the source and target DB instances.
2. The user starts AWS SCT with the Python script, converts the Oracle code to PostgreSQL code, and runs it on the target DB instance.
3. The Python script creates AWS DMS replication tasks for the source and target DB instances.
4. The user deploys Python scripts to start the AWS DMS tasks and then stops the tasks after the data migration is complete.

Automation and scale

You can automate this migration by adding parameters and security-related changes to your Python script, to provide additional functionality.

Tools

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command line shell.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions. This pattern converts the `.csv` input file to a `.json` input file by using a Python script. The `.json` file is used in AWS CLI commands to create an AWS CloudFormation stack that creates multiple AWS DMS replication tasks with Amazon Resource Names (ARNs), migration types, task settings, and table mappings.
- AWS Database Migration Service (AWS DMS) helps you migrate data stores to the AWS Cloud or between combinations of cloud and on-premises setups. This pattern uses AWS DMS to create, start, and stop tasks with a Python script that runs on the command line, and to create the AWS CloudFormation template.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. This patterns requires the `AWSSchemaConversionToolBatch.jar` file from the installed AWS SCT directory.

Code

The `cli-sct-dms-cft.zip` file (attached) contains the complete source code for this pattern.

Epics

Configure AWS SCT and create database objects in the AWS CLI

Task	Description	Skills required

Configure AWS SCT to run from the AWS CLI.	1. Configure the source and target environment configuration details in the `database_migration.txt` file by using the following format:
	 `#source_vendor,source_hostname,source_dbname,source_user,source_pwd,source_schema,source_port,source_sid,target_vendor,target_hostname,target_user,target_pwd,target_dbname,target_port`
	 `ORACLE,myoracledb.cokmvis0v46q.us-east-1.rds.amazonaws.com,ORCL,orcl,orcl1234,orcl,1521,ORCL,POSTGRESQL,mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com,pguser,pgpassword,pgdb,5432`
	 2. Modify the AWS SCT configuration parameters according to your requirements in the following files: `project_s`
		DBA

Task	Description	Skills required
	`ettings.xml` , `Oracle_PG_Test_Batch.xml` , and `ORACLE-orcl-to-POSTGRESQL.xml` .	
Run the `run_aws_sct.py` Python script.	Run the `run_aws_sct.py` Python script by using the following command: `$ python run_aws_sct.py database_migration.txt` The Python script converts the database objects from Oracle to PostgreSQL and creates SQL files in PostgreSQL format. The script also creates the PDF file `Database migration assessment report` , which provides you with detailed recommendations and conversion statistics for database objects.	DBA
Create objects in Amazon RDS for PostgreSQL.	1. Manually modify the SQL files generated by AWS SCT, if required. 2. Run the SQL files and create objects in your Amazon RDS for PostgreSQL DB instance.	DBA

Configure and create AWS DMS tasks by using the AWS CLI and AWS CloudFormation

Task	Description	Skills required
Create an AWS DMS replication instance.	Sign in to the AWS Management Console, open the AWS DMS console, and create a replication instance that is configured according to your requirements. For more information, see Creating a replication instance in the AWS DMS documentation and How do I create an AWS DMS replication instance in the AWS Support documentation.	DBA
Create the source endpoint.	On the AWS DMS console, choose **Endpoints** and then create a source endpoint for the Oracle database according to your requirements. **Note** The extra connection attribute must be `numberDataTypeScale` with a `-2` value. For more information, see Creating source and target	DBA

Task	Description	Skills required
	endpoints in the AWS DMS documentation.	
Create the target endpoint.	On the AWS DMS console, choose **Endpoints** and then create a target endpoint for the PostgreSQL database according to your requireme nts. For more information, see Creating source and target endpoints in the AWS DMS documentation.	DevOps engineer

Task	Description	Skills required
Configure the AWS DMS replication details to run from the AWS CLI.	Configure the AWS DMS source and target endpoints and replication details in the `dms-arn-list.txt` file with the source endpoint ARN, target endpoint ARN, and the replication instance ARN by using the following format: `#sourceARN,targetARN,repARN arn:aws:dms:us-east-1:123456789012:endpoint:EH7AINRUDZ5GOYIY6HVMXECMCQ arn:aws:dms:us-east-1:123456789012:endpoint:HHJVUV57N7O3CQF4PJZKGIOYY5 arn:aws:dms:us-east-1:123456789012:rep:LL57N77AQQAHHJF4PJFHNEDZ5G`	DBA

Task	Description	Skills required
Run the `dms-create-task.py` Python script to create the AWS DMS tasks.	1. Run the `dms-create-task.py` Python script by using the following command: `$ python dms-create-task.py database_migration.txt dms-arn-list.txt <cft-stack-name> <migration-type>` where: • `database_migration.txt` is the database migration text file. • `dms-arn-list.txt` is the ARN list for AWS DMS. • `<cft-stack-name>` is the user-defined AWS CloudFormation stack name. • `<migration-type>` is `full-load`, `cdc`, or `full-load-and-cdc`. 2. Depending on your migration type, you can use the following commands to create three types of AWS DMS tasks: • `$ python dms-create-task.py database_migration`	DBA

Task	Description	Skills required
	.txt dms-arn-l ist.txt dms-cli-c ft-stack full-load • `$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack cdc` • `$ python dms-create-task.py database_migration.txt dms-arn-list.txt dms-cli-cft-stack full-load-and-cdc`	
Verify that AWS DMS tasks are ready.	On the AWS DMS console, check that your AWS DMS tasks are in `Ready` status in the **Status** section.	DBA

Start and stop the AWS DMS tasks by using the AWS CLI

Task	Description	Skills required
Start the AWS DMS tasks.	Run the `dms-start-task.py` Python script by using the following command:	DBA

Task	Description	Skills required
	`$ python dms-start-task.py start '<cdc-start-datetime>'` **Note** The start date and time must be in the `'DD-MON-YYYY'` or `'YYYY-MM-DDTHH:MI:SS'` format (for example, `'01-Dec-2019'` or `'2018-03-08T12:12:12'`). You can review the AWS DMS task status in the **Table statistics** tab on the **Tasks** page of the AWS DMS console.	

Task	Description	Skills required
Validate the data.	1. After the full-load migration is complete, the task is continuously kept running for CDC. 2. When CDC is complete or no more changes need to be migrated, review and validate the migration task results and data in your Oracle and PostgreSQL databases. You can validate your data by checking status and count columns (`Validation state`, `Validation pending`, `Validation failed`, `Validation suspended`, and `Validation details`) in the **Table statistics** tab of your database migration task on the **Tasks** page of the AWS DMS console. For more information, see [AWS DMS data validation](#) in the AWS DMS documentation.	DBA

Task	Description	Skills required
Stop the AWS DMS tasks.	Run the Python script by using the following command: `$ python dms-start-task.py stop` **Note** AWS DMS tasks might stop with a `failed`status, depending on the validation status. For more information, see the next section.	DBA

Troubleshooting

Issue	Solution
AWS SCT source and target test connections fail.	Configure the JDBC driver versions and VPC security group inbound rules to accept the incoming traffic.
Source or target endpoint test run fails.	Check if the endpoint settings and replication instance are in `Available` status. Check if the endpoint connection status is `Successful`. For more information, see [How do I troubleshoot AWS DMS endpoint connectivity failures](https://repost.aws/knowledge-center/dms-endpoint-connectivity-failures) in the AWS Support documentation.

Issue	Solution
Full-load run fails.	Check if the source and target databases have matching data types and sizes. For more information, see Troubleshooting migration tasks in AWS DMS in the AWS DMS documentation.
You encounter validation run errors.	Check if the table has a primary key because non-primary key tables are not validated. If the table has a primary key and errors, check that the extra connection attribute in the source endpoint has `numberDataTypeScale=-2` . For more information, see Endpoint settings when using Oracle as a source for AWS DMS, OracleSettings, and Troubleshooting in the AWS DMS documentation.

Related resources

- Installing and configuring AWS SCT
- Introduction to AWS DMS (video)
- Examples of CloudFormation stack operation commands for the AWS CLI and PowerShell
- Navigating the user interface of the AWS SCT
- Using an Oracle database as a source for AWS DMS
- Connecting to Oracle databases with AWS SCT
- Using a PostgreSQL database as a target for AWS DMS
- Sources for data migration
- Targets for data migration
- cloudformation (AWS CLI documentation)
- create-stack (AWS CLI documentation)

- dms (AWS CLI documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL in SSL mode by using AWS DMS

Pinesh Singal, Amazon Web Services

Summary

This pattern provides guidance for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle database instance to an Amazon RDS for PostgreSQL database on the Amazon Web Services (AWS) Cloud. To encrypt connections between the databases, the pattern uses certificate authority (CA) and SSL mode in Amazon RDS and AWS Database Migration Service (AWS DMS).

The pattern describes an online migration strategy with little or no downtime for a multi-terabyte Oracle source database with a high number of transactions. For data security, the pattern uses SSL when transferring the data.

This pattern uses AWS Schema Conversion Tool (AWS SCT) to convert the Amazon RDS for Oracle database schema to an Amazon RDS for PostgreSQL schema. Then the pattern uses AWS DMS to migrate data from the Amazon RDS for Oracle database to the Amazon RDS for PostgreSQL database.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Amazon RDS database certificate authority (CA) configured with ***rds-ca-rsa2048-g1*** only
  - The ***rds-ca-2019*** certificate expired in August 2024.
  - The ***rds-ca-2015*** certificate expired on March 5, 2020.
- AWS SCT
- AWS DMS
- pgAdmin
- SQL tools (for example, SQL Developer or SQL*Plus)

Limitations

- Amazon RDS for Oracle database – The minimum requirement is for Oracle versions 19c for the Enterprise and Standard Two editions.
- Amazon RDS for PostgreSQL database – The minimum requirement is for PostgreSQL version 12 and later (for versions 9.x and later).

Product versions

- Amazon RDS for Oracle database version 12.1.0.2 instance
- Amazon RDS for PostgreSQL database version 11.5 instance

Architecture

Source technology stack

- An Amazon RDS for Oracle database instance with version 12.1.0.2.v18.

Target technology stack

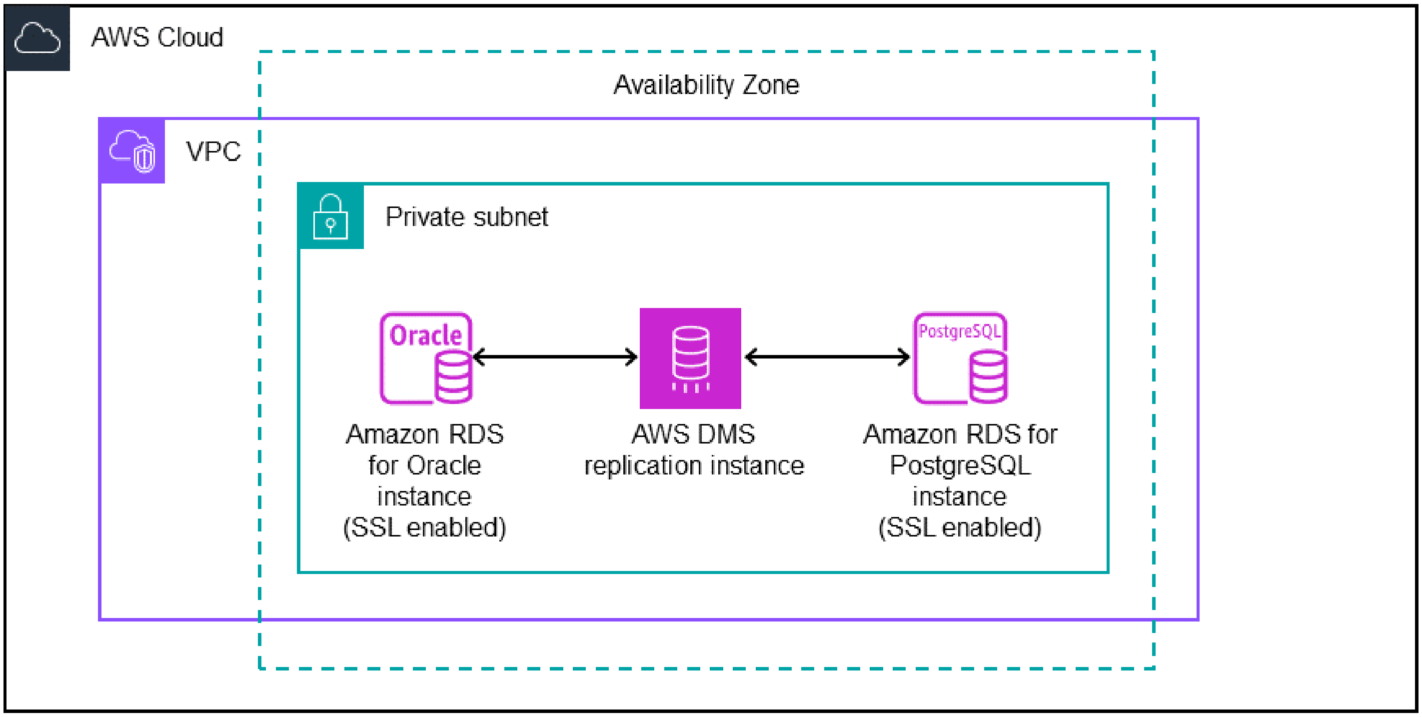
- AWS DMS
- An Amazon RDS for PostgreSQL database instance with version 11.5.

Target architecture

The following diagram shows the architecture for data migration architecture between Oracle (source) and PostgreSQL (target) databases. The architecture includes the following:

- A virtual private cloud (VPC)
- An Availability Zone
- A private subnet
- An Amazon RDS for Oracle database
- An AWS DMS replication instance
- An RDS for PostgreSQL database

To encrypt connections for source and target databases, CA and SSL mode must be enabled in Amazon RDS and AWS DMS.



Tools

AWS services

- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Amazon Relational Database Service (Amazon RDS) for Oracle helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.
- Amazon Relational Database Service (Amazon RDS) for PostgreSQL helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.

Other services

- pgAdmin is an open source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.

Best practices

Amazon RDS provides new CA certificates as an AWS security best practice. For information about the new certificates and the supported AWS Regions, see Using SSL/TLS to encrypt a connection to a DB instance or cluster.

If your RDS instance is currently on CA certificate `rds-ca-2019`, and you want to upgrade to `rds-ca-rsa2048-g1`, follow the instructions in Updating your CA certificate by modifying your DB instance or cluster or Updating your CA certificate by applying maintenance.

Epics

Configure the Amazon RDS for Oracle instance

Task	Description	Skills required
Create the Oracle database instance.	Sign in to your AWS account, open the AWS Managemen t Console, and navigate to the Amazon RDS console. On the console, choose **Create database**, and then choose **Oracle**.	General AWS, DBA
Configure security groups.	Configure inbound and outbound security groups.	General AWS
Create an option group.	Create an option group in the same VPC and security group as the Amazon RDS for Oracle database. For **Option**, choose **SSL**. For **Port**, choose **2484** (for SSL connections).	General AWS
Configure the option settings.	Use the following settings: • `SQLNET.CIPHER_SUITE :SSL_RSA_WITH_AES_256_CBC_SHA`	General AWS

Task	Description	Skills required
	• `SQLNET.SSL_VERSION : 1.2 or 1.0`	
Modify the RDS for Oracle DB instance.	Set the CA certificate as **rds-ca-rsa2048-g1**. Under **Option group**, attach the previously created option group.	DBA, General AWS

Task	Description	Skills required

Confirm that the RDS for Oracle DB instance is available.	Make sure that the Amazon RDS for Oracle database instance is up and running and that the database schema is accessible.
	
	To connect to the RDS for Oracle DB, use the `sqlplus` command from the command line.
	
	`$ sqlplus orcl/****@myoracledb.cokmvis0v46q.us-east-1.rds.amazonaws.com:1521/ORCL`
	`SQL*Plus: Release 12.1.0.2.0 Production on Tue Oct 15 18:11:07 2019`
	`Copyright (c) 1982, 2016, Oracle.  All rights reserved.`
	`Last Successful login time: Mon Dec 16 2019 23:17:31 +05:30`
	`Connected to:`
	`Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production`
	`With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options`
	`SQL>`
		DBA

Task	Description	Skills required
Create objects and data in the RDS for Oracle database.	Create objects and insert data in the schema.	DBA

Configure the Amazon RDS for PostgreSQL instance

Task	Description	Skills required
Create the RDS for PostgreSQL database.	On the Amazon RDS console **Create database** page, choose **PostgreSQL** to create an Amazon RDS for PostgreSQL database instance.	DBA, General AWS
Configure security groups.	Configure inbound and outbound security groups.	General AWS
Create a parameter group.	If you are using PostgreSQL version 11.x, create a parameter group to set SSL parameters. In PostgreSQL version 12, the SSL parameter group is enabled by default.	General AWS
Edit parameters.	Change the `rds.force_ssl` parameter to 1 (on). By default, the `ssl` parameter is 1 (on). By setting the `rds.force_ssl` parameter to 1, you force all connections to connect through SSL mode only.	General AWS

Task	Description	Skills required
Modify the RDS for PostgreSQL DB instance.	Set the CA certificate as **rds-ca-rsa2048-g1**. Attach the default parameter group or the previously created parameter group, depending on your PostgreSQL version.	DBA, General AWS

Task	Description	Skills required

Confirm that the RDS for PostgreSQL DB instance is available.	Make sure that the Amazon RDS for PostgreSQL database is up and running.
	
	The `psql` command establishes an SSL connection with `sslmode` set from the command line.
	
	One option is to set `sslmode=1` in the parameter group and use a `psql` connection without including the `sslmode` parameter in the command.
	
	The following output shows that the SSL connection is established.
	
	`$ psql -h mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com -p 5432 "dbname=pgdb user=pguser"`
	`Password for user pguser:`
	`psql (11.3, server 11.5)`
	`SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)`
	`Type "help" for help.`
	`pgdb=>`
		DBA

Task	Description	Skills required
	A second option is to set `sslmode=1` in the parameter group and to include the `sslmode` parameter in the `psql` command. The following output shows that the SSL connection is established. `$ psql -h mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com -p 5432 "dbname=pgdb user=pguser sslmode=require" Password for user pguser: psql (11.3, server 11.5) SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off) Type "help" for help. pgdb=>`	

Configure and run AWS SCT

Task	Description	Skills required
Install AWS SCT.	Install the latest version of the AWS SCT application.	General AWS

Task	Description	Skills required
Configure AWS SCT with JDBC drivers.	Download the Java Database Connectivity (JDBC) drivers for Oracle (ojdbc8.jar) and PostgreSQL (postgresql-42.2.5.jar). To configure the drivers in AWS SCT, choose **Settings**, **Global settings**, **Drivers**.	General AWS

Task	Description	Skills required
Create the AWS SCT project.	Create the AWS SCT project and report, using Oracle as the source DB engine and Amazon RDS for PostgreSQL as the target DB engine: 1. Test connections to the source Oracle database and target Amazon RDS for PostgreSQL database by providing connection details. For the source Oracle database, the following permissions or privileges are required: • CONNECT • SELECT_CATALOG_ROLE • SELECT ANY DICTIONARY • SELECT on SYS.USER$ TO <sct_user> For more information, see Using Oracle Database as a source for AWS SCT. Both source and target connections must be successful before AWS SCT can start the migration report.	General AWS

Task	Description	Skills required
	2. After the report, enter the schema to be converted, and choose **Finish**.	
Validate database objects.	1. Choose **Load schema**. AWS SCT displays the source and the converted target objects, including objects that that have errors. Update any incorrect objects on the target database. 2. Review the errors, and clear them by using manual intervention. 3. After all errors are cleared, choose **Load schema** again. 4. Choose **Apply to database**. 5. Connect to pgAdmin, or any tool that supports a PostgreSQL DB connection, and check the schema and objects.	DBA, General AWS

Configure and run AWS DMS

Task	Description	Skills required
Create a replication instance.	1. Sign in to your account, open the AWS Managemen t Console, and navigate to the AWS DMS console.	General AWS

Task	Description	Skills required
	2. Create a replication instance with valid settings for the VPC, security group, Availability Zone, and extra connection attributes.	
Import the certificate.	Download the [certificate bundle (PEM)]() for your AWS Region. The bundle contains both the `rds-ca-2019` intermediate and root certificates. The bundle also contains the `rds-ca-rsa2048-g1`, `rds-ca-rsa4096-g1`, and `rds-ca-ecc384-g1` root CA certificates. Your application trust store needs to register only the root CA certificate.	General AWS

Task	Description	Skills required
Create the source endpoint.	1. Create a source endpoint for Amazon RDS for Oracle by choosing **Select RDS DB instance** and then selecting the RDS for Oracle DB instance that you created. The endpoint configuration details will be automatically populated . 2. Choose **Provide access information manually**. For **Port**, make sure that you enter **2484**. 3. Under **Secure Socket Layer (SSL) mode**, choose `verify-ca` , and then choose the CA certificate that you created previousl y. 4. Under **Endpoint settings**, add the extra connectio n attribute `NumberDat aTypeScale=-2` to support the `NUMBER` data type with no size. For more information, see [Using an Oracle database as a source for AWS Database Migration Service]().	General AWS

Task	Description	Skills required
Create the target endpoint.	1. Create a target endpoint for Amazon RDS for PostgreSQL by choosing **Select RDS DB instance** and then selecting your RDS for PostgreSQL DB instance. The endpoint configuration details will be automatically populated . 2. Choose **Provide access information manually**. For **Port**, make sure that you enter **2484**. For more information, see [Using a PostgreSQL database as a target for AWS Database Migration Service]().	General AWS
Test the endpoints.	1. Test the source and target endpoints to confirm that both are successful and available. 2. If a test fails, make sure that the security group inbound rules are valid.	General AWS

Task	Description	Skills required
Create migration tasks.	To create a migration task for full load and change data capture (CDC) or for data validation, do the following: 1. To create a database migration task, choose the replication instance, source database endpoint, target database endpoint. Specify the migration type as one of the following: • **Migrate existing data** (full load) • **Replicate data changes only (CDC)** • **Migrate existing data and replicate ongoing changes** (full load and CDC) 2. Under **Table mappings**, you can configure selection rules and transformation rules in GUI or JSON formats: • Under **Selection rules**, select the schema, enter the table name, and select the action (Include or Exclude) to be configured; for example, Schema ORCL, Table name %, Action Include.	General AWS

Task	Description	Skills required

	- Under **Transformation rules**, do one of the following: - Select the schema and choose the action (case, prefix, suffix); for example, Target Schema ORCL, Action Make lowercase. - Select the schema, enter the table name, and choose the action (case, prefix, suffix); for example, Target Schema ORCL, Table %, Action Make lowercase. 3. Turn on Amazon CloudWatch Logs monitoring.
	4. For the mapping rules, add the following JSON code.
	<pre>{
	    "rules": [
	        {
	            "rule-type": "transformation",
	            "rule-id": "1",
	            "rule-name": "1",
	            "rule-target": "table",</pre>	

Task	Description	Skills required

	``
	            "object-l
	ocator": {
	
	 "schema-name": "%",
	
	 "table-name": "%"
	            },
	            "rule-
	action": "convert-
	lowercase",
	            "value":
	 null,
	            "old-valu
	e": null
	        },
	        {
	            "rule-
	type": "transfor
	mation",
	            "rule-id"
	: "2",
	            "rule-nam
	e": "2",
	            "rule-tar
	get": "schema",
	            "object-l
	ocator": {
	
	 "schema-name":
	 "ORCL",
	
	 "table-name": "%"
	            },
	            "rule-
	action": "convert-
	lowercase",
	            "value":
	 null,
	            "old-valu
	e": null
	        },
	        {
	``	

Task	Description	Skills required

	``
	            "rule-typ
	e": "selection",
	            "rule-id"
	: "3",
	            "rule-nam
	e": "3",
	            "object-l
	ocator": {
	
	 "schema-name":
	 "ORCL",
	
	 "table-name": "DEPT"
	            },
	            "rule-act
	ion": "include",
	            "filters"
	: []
	        }
	    ]
	}
	``	

Plan the production run.	Confirm downtime with stakeholders such as applicati on owners to run AWS DMS in production systems.	Migration lead

Task	Description	Skills required
Run the migration task.	1. Start the AWS DMS task that has a status of **Ready**, and monitor the migration task logs in Amazon CloudWatch for any errors. If you chose **Migrate existing data and replicate ongoing changes** as the migration type, and the status is **Load complete ongoing replication**, the full load with CDC data migration is completed and validation is ongoing. 2. After you start the migration, you can obtain additional SSL-connection information in CloudWatch. For Oracle, CloudWatch shows the following connection string. `2019-12-17T09:15:11 [SOURCE_UNLOAD ]I: Connecting to Oracle: Beginning session (oracle_endpoint_conn.c:834)` The PostgreSQL connection string will be similar to the following example.	General AWS

Task	Description	Skills required
	`2019-12-17T09:15:11 [TARGET_LOAD ]I: Going to connect to ODBC connection string: PROTOCOL=7.4-0;DRIVER={PostgreSQL};SERVER=mypgdbinstance.cokmvis0v46q.us-east-1.rds.amazonaws.com;DATABASE=pgdb;PORT=5432;sslmode=require;UID=pguser;(odbc_endpoint_imp.c:2218)`	
Validate the data.	Review migration task results and data in the source Oracle and target PostgreSQL databases: 1. Connect to pgAdmin and check the data in your PostgreSQL database with schema `ORCL`. 2. For CDC, check the ongoing changes by inserting or updating data in the source Oracle database.	DBA
Stop the migration task.	After you successfully complete the data validation, stop the migration task.	General AWS

Clean up the resources

Task	Description	Skills required
Delete the AWS DMS tasks.	1. On the AWS DMS console, navigate to **Database migration tasks**, and stop any ongoing or running AWS DMS task. 2. Select the task or tasks, choose **Actions**, and choose **Delete**.	General AWS
Delete the AWS DMS endpoints.	Select the source and target endpoints that you created, choose **Actions**, and choose **Delete**.	General AWS
Delete the AWS DMS replicati on instance.	Choose the replication instance, choose **Actions**, and then choose **Delete**.	General AWS
Delete the PostgreSQL database.	1. On the Amazon RDS console, choose **Databases**. 2. Select the PostgreSQL database instance that you created, choose **Actions**, and then choose **Delete**.	General AWS
Delete the Oracle database.	On the Amazon RDS console, select the Oracle database instance, choose **Actions**, and then choose **Delete**.	General AWS

Troubleshooting

Issue	Solution
AWS SCT source and target test connections are failing.	Configure JDBC driver versions and VPC security group inbound rules to accept the incoming traffic.
The Oracle source endpoint test run fails.	Check the endpoint settings and whether the replication instance is available.
The AWS DMS task full-load run fails.	Check whether the source and target databases have matching data types and sizes.
The AWS DMS validation migration task returns errors.	1. Check whether the table has a primary key. Tables without a primary key are not validated. 2. If the table has a primary key but returns errors, check the extra connection attribute in the source endpoint. The extra connection attribute must have `numberDataTypeScale=-2` to support the `NUMBER` data type with no size dynamically based on data available in table.

Related resources

Databases

- [Amazon RDS for Oracle]
- [Amazon RDS for PostgreSQL]

SSL DB connection

- [Using SSL/TLS to encrypt a connection to a DB instance]
  - [Using SSL with an RDS for Oracle DB instance]
  - [Securing connections to RDS for PostgreSQL with SSL/TLS]

  - Download certificate bundles for specific AWS Regions
    - Download CA-2019 root certificate (expired in August 2024)
- Working with option groups
  - Adding options to Oracle DB instances
  - Oracle Secure Sockets Layer
- Working with parameter groups
- PostgreSQL sslmode connection parameter
- Using SSL from JDBC
- Rotating your SSL/TLS certificate
  - Updating your CA certificate by modifying your DB instance or cluster
  - Updating your CA certificate by applying maintenance

AWS SCT

- AWS Schema Conversion Tool
- AWS Schema Conversion Tool User Guide
- Using the AWS SCT user interface
- Using Oracle Database as a source for AWS SCT

AWS DMS

- AWS Database Migration Service
- AWS Database Migration Service User Guide
  - Using an Oracle database as a source for AWS DMS
  - Using a PostgreSQL database as a target for AWS DMS
- Using SSL with AWS Database Migration Service
- Migrating Applications Running Relational Databases to AWS

Additional information

Amazon RDS Certificate Authority certificates `rds-ca-2019` expired in August 2024. If you use or plan to use SSL or TLS with certificate verification to connect to your RDS DB instances or Multi-

AZ DB clusters, consider using one of the new CA certificates: `rds-ca-rsa2048-g1`, `rds-ca-rsa4096-g1`, or `rds-ca-ecc384-g1`.

Migrate Oracle SERIALLY_REUSABLE pragma packages into PostgreSQL

Vinay Paladi, Amazon Web Services

Summary

This pattern provides a step-by-step approach for migrating Oracle packages that are defined as SERIALLY_REUSABLE pragma to PostgreSQL on Amazon Web Services (AWS). This approach maintains the functionality of the SERIALLY_REUSABLE pragma.

PostgreSQL doesn't support the concept of packages and the SERIALLY_REUSABLE pragma. To get similar functionality in PostgreSQL, you can create schemas for packages and deploy all the related objects (such as functions, procedures, and types) inside the schemas. To achieve the functionality of the SERIALLY_REUSABLE pragma, the example wrapper function script that's provided in this pattern uses an AWS Schema Conversion Tool (AWS SCT) extension pack.

For more information, see SERIALLY_REUSABLE Pragma in the Oracle documentation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The latest version of AWS SCT and the required drivers
- An Amazon Aurora PostgreSQL-Compatible Edition database or an Amazon Relational Database Service (Amazon RDS) for PostgreSQL database

Product versions

- Oracle Database version 10g and later

Architecture

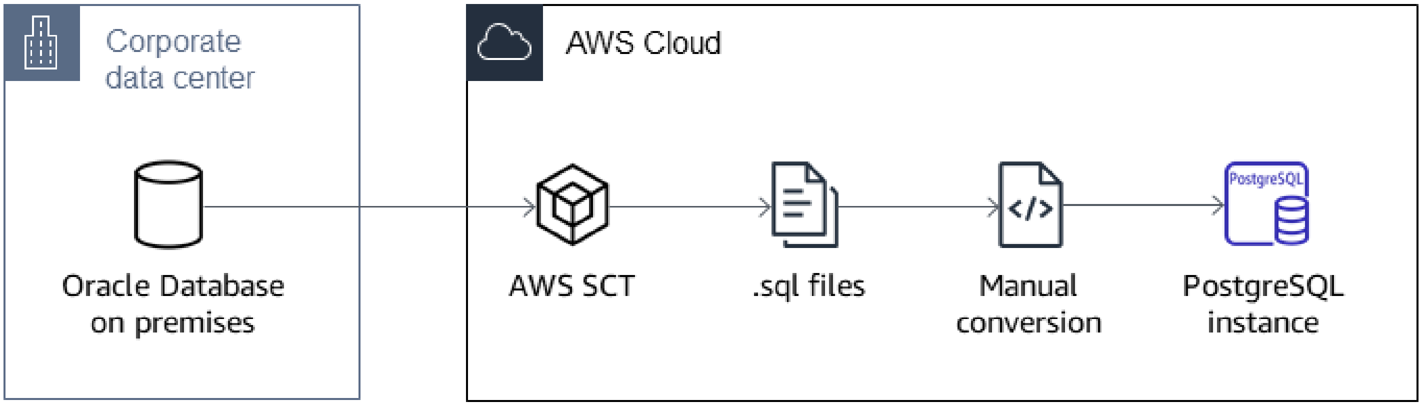
Source technology stack

- Oracle Database on premises

Target technology stack

- Aurora PostgreSQL-Compatible or Amazon RDS for PostgreSQL
- AWS SCT

Migration architecture



Tools

AWS services

- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.
- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- Amazon Relational Database Service (Amazon RDS) for PostgreSQL helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.

Other tools

- pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.

Epics

Migrate the Oracle package by using AWS SCT

Task	Description	Skills required
Set up AWS SCT.	Configure AWS SCT connectiv ity to the source database. For more information, see [Using Oracle Database as a source for AWS SCT]().	DBA, Developer
Convert the script.	Use AWS SCT to convert the Oracle package by selecting the target database as Aurora PostgreSQL-Compatible.	DBA, Developer
Save the .sql files.	Before you save the .sql file, modify the **Project Settings** option in AWS SCT to **Single file per stage**. AWS SCT will separate the .sql file into multiple .sql files based on object type.	DBA, Developer
Change the code.	Open the `init` function generated by AWS SCT, and change it as shown in the example in the *Additional information* section. It will add a variable to achieve the functionality `pg_serialize = 0` .	DBA, Developer
Test the conversion.	Deploy the `init` function to the Aurora PostgreSQL-Compatible database, and test the results.	DBA, Developer

Related resources

- [AWS Schema Conversion Tool]
- [Amazon RDS]
- [Amazon Aurora features]
- [SERIALLY_REUSABLE Pragma]

Additional information

```
Source Oracle Code:

CREATE OR REPLACE PACKAGE test_pkg_var
IS
PRAGMA SERIALLY_REUSABLE;
PROCEDURE function_1
 (test_id number);
PROCEDURE function_2
 (test_id number
 );
END;

CREATE OR REPLACE PACKAGE BODY test_pkg_var
IS
PRAGMA SERIALLY_REUSABLE;
v_char VARCHAR2(20) := 'shared.airline';
v_num number := 123;

PROCEDURE function_1(test_id number)
IS
begin
dbms_output.put_line( 'v_char-'|| v_char);
dbms_output.put_line( 'v_num-'||v_num);
v_char:='test1';
function_2(0);
END;

PROCEDURE function_2(test_id number)
is
begin
dbms_output.put_line( 'v_char-'|| v_char);
dbms_output.put_line( 'v_num-'||v_num);
```

```
END;
END test_pkg_var;

Calling the above functions

set serveroutput on


EXEC test_pkg_var.function_1(1);


EXEC test_pkg_var.function_2(1);


Target Postgresql Code:


CREATE SCHEMA test_pkg_var;

CREATE OR REPLACE FUNCTION test_pkg_var.init(pg_serialize IN INTEGER DEFAULT 0)

RETURNS void
AS
$BODY$

DECLARE

BEGIN

if aws_oracle_ext.is_package_initialized( 'test_pkg_var' ) AND pg_serialize = 0

then

return;

end if;

PERFORM aws_oracle_ext.set_package_initialized( 'test_pkg_var' );

PERFORM aws_oracle_ext.set_package_variable( 'test_pkg_var', 'v_char',
 'shared.airline.basecurrency'::CHARACTER

VARYING(100));
```

```
PERFORM aws_oracle_ext.set_package_variable('test_pkg_var', 'v_num', 123::integer);

END;

$BODY$

LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION test_pkg_var.function_1(pg_serialize int default 1)

RETURNS void
AS

$BODY$
DECLARE

BEGIN

PERFORM test_pkg_var.init(pg_serialize);

raise notice 'v_char%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_char');

raise notice 'v_num%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_num');

PERFORM aws_oracle_ext.set_package_variable( 'test_pkg_var', 'v_char',
 'test1'::varchar);

PERFORM test_pkg_var.function_2(0);
END;

$BODY$
LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION test_pkg_var.function_2(IN pg_serialize integer default 1)

RETURNS void

AS

$BODY$

DECLARE
```

```
BEGIN

PERFORM test_pkg_var.init(pg_serialize);

raise notice 'v_char%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_char');

raise notice 'v_num%',aws_oracle_ext.get_package_variable( 'test_pkg_var', 'v_num');

END;
$BODY$
LANGUAGE plpgsql;


Calling the above functions

select test_pkg_var.function_1()

 select test_pkg_var.function_2()
```

Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible

anuradha chintha and Rakesh Raghav, Amazon Web Services

Summary

External tables give Oracle the ability to query data that is stored outside the database in flat files. You can use the ORACLE_LOADER driver to access any data stored in any format that can be loaded by the SQL*Loader utility. You can't use Data Manipulation Language (DML) on external tables, but you can use external tables for query, join, and sort operations.

Amazon Aurora PostgreSQL-Compatible Edition doesn't provide functionality similar to external tables in Oracle. Instead, you must use modernization to develop a scalable solution that meets functional requirements and is frugal.

This pattern provides steps for migrating different types of Oracle external tables to Aurora PostgreSQL-Compatible Edition on the Amazon Web Services (AWS) Cloud by using the `aws_s3` extension.

We recommend thoroughly testing this solution before implementing it in a production environment.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI)
- An available Aurora PostgreSQL-Compatible database instance.
- An on-premises Oracle database with an external table
- pg.Client API
- Data files

Limitations

- This pattern doesn't provide the functionality to act as a replacement for Oracle external tables. However, the steps and sample code can be enhanced further to achieve your database modernization goals.

- Files should not contain the character that is passing as a delimiter in `aws_s3` export and import functions.

Product versions

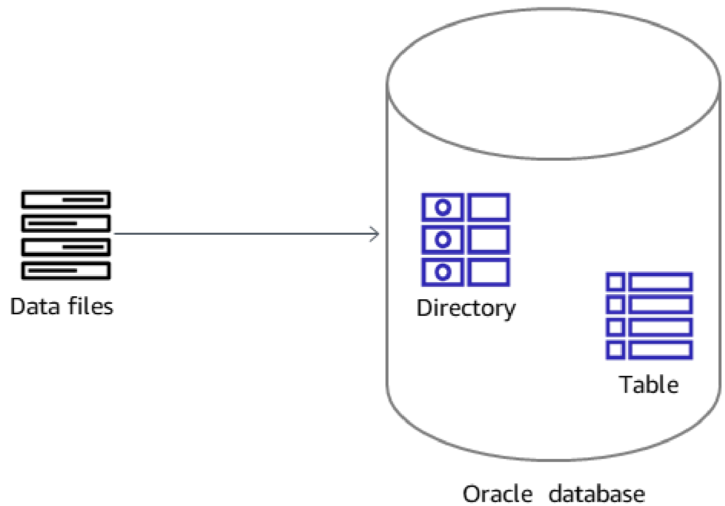
- To import from Amazon S3 into RDS for PostgreSQL the database must be running PostgreSQL version 10.7 or later.

Architecture

Source technology stack

- Oracle

Source architecture

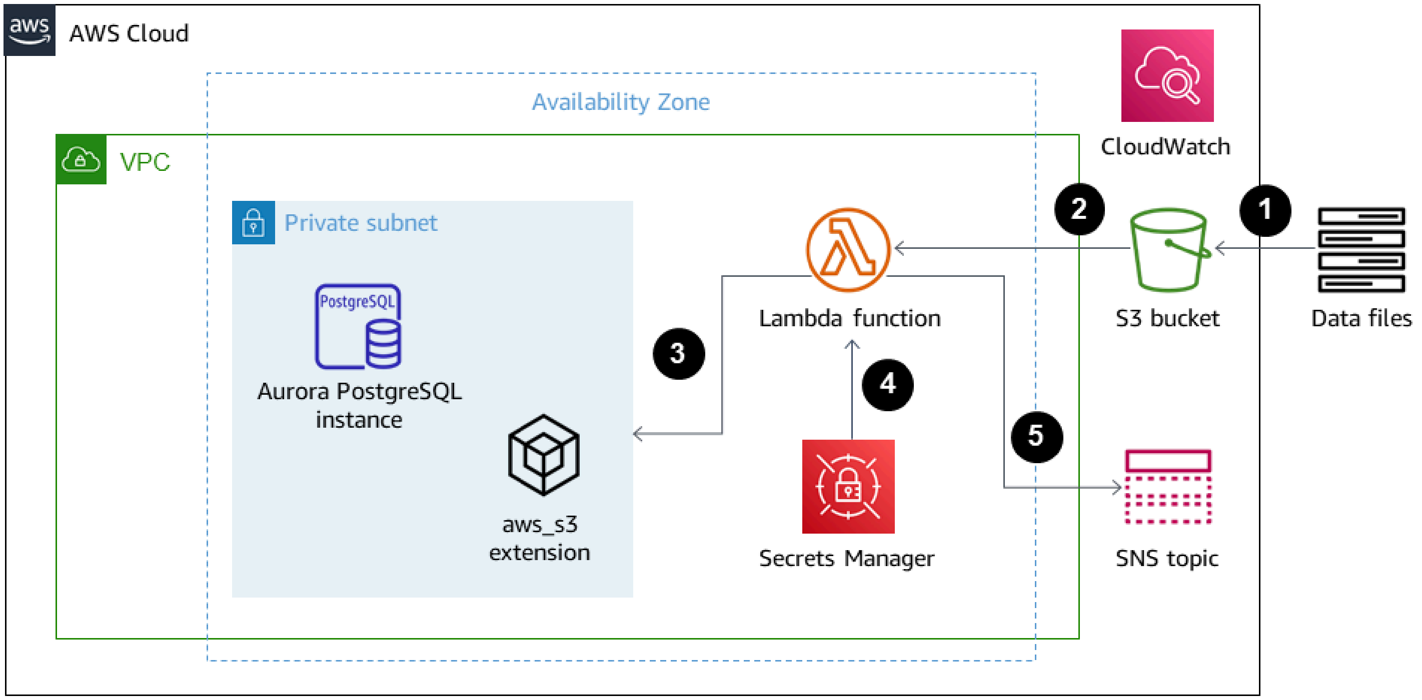


Target technology stack

- Amazon Aurora PostgreSQL-Compatible
- Amazon CloudWatch
- AWS Lambda
- AWS Secrets Manager
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Storage Service (Amazon S3)

Target architecture

The following diagram shows a high-level representation of the solution.



1. Files are uploaded to the S3 bucket.
2. The Lambda function is initiated.
3. The Lambda function initiates the DB function call.
4. Secrets Manager provides the credentials for database access.
5. Depending on the DB function, an SNS alarm is created.

Automation and scale

Any additions or changes to the external tables can be handled with metadata maintenance.

Tools

- Amazon Aurora PostgreSQL-Compatible – Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine that combines the speed and reliability of high-end commercial databases with the cost-effectiveness of open-source databases.

- AWS CLI – AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With only one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.
- Amazon CloudWatch – Amazon CloudWatch monitors Amazon S3 resources and utilization.
- AWS Lambda – AWS Lambda is a serverless compute service that supports running code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes. In this pattern, Lambda runs the database function whenever a file is uploaded to Amazon S3.
- AWS Secrets Manager – AWS Secrets Manager is a service for credential storage and retrieval. Using Secrets Manager, you can replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) provides a storage layer to receive and store files for consumption and transmission to and from the Aurora PostgreSQL-Compatible cluster.
- aws_s3 – The `aws_s3` extension integrates Amazon S3 and Aurora PostgreSQL-Compatible.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients. In this pattern, Amazon SNS is used to send notifications.

Code

Whenever a file is placed in the S3 bucket, a DB function must be created and called from the processing application or the Lambda function. For details, see the code (attached).

Epics

Create an external file

Task	Description	Skills required
Add an external file to the source database.	Create an external file, and move it to the `oracle` directory.	DBA

Configure the target (Aurora PostgreSQL-Compatible)

Task	Description	Skills required
Create an Aurora PostgreSQL database.	Create a DB instance in your Amazon Aurora PostgreSQL-Compatible cluster.	DBA
Create a schema, aws_s3 extension, and tables.	Use the code under `ext_tbl_scripts` in the *Additional information* section. The tables include actual tables, staging tables, error and log tables, and a metatable.	DBA, Developer
Create the DB function.	To create the DB function, use the code under `load_external_table_latest` function in the *Additional information* section.	DBA, Developer

Create and configure the Lambda function

Task	Description	Skills required
Create a role.	Create a role with permissions to access Amazon S3 and Amazon Relational Database Service (Amazon RDS). This role will be assigned to Lambda for running the pattern.	DBA
Create the Lambda function.	Create a Lambda function that reads the file name from Amazon S3 (for example,	DBA

Task	Description	Skills required
	`file_key = info.get('object', {}).get('key')`) and calls the DB function (for example, `curs.callproc("load_external_tables", [file_key])`) with the file name as the input parameter. Depending on the function call result, an SNS notification will be initiated (for example, `client.publish(TopicArn='arn:',Message='fileloadsuccess',Subject='fileloadsuccess')`). Based on your business needs, you can create a Lambda function with extra code if required. For more information, see the [Lambda documentation]().	
Configure an S3 bucket event trigger.	Configure a mechanism to call the Lambda function for all object creation events in the S3 bucket.	DBA
Create a secret.	Create a secret name for the database credentials by using Secrets Manager. Pass the secret in the Lambda function.	DBA

Task	Description	Skills required
Upload the Lambda supporting files.	Upload a .zip file that contains the Lambda support packages and the attached Python script for connecting to Aurora PostgreSQL-Compatible. The Python code calls the function that you created in the database.	DBA
Create an SNS topic.	Create an SNS topic to send mail for the success or failure of the data load.	DBA

Add integration with Amazon S3

Task	Description	Skills required
Create an S3 bucket.	On the Amazon S3 console, create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts.	DBA
Create IAM policies.	To create the AWS Identity and Access Management (IAM) policies, use the code under `s3bucketpolicy_for_import` in the *Additional information* section.	DBA
Create roles.	Create two roles for Aurora PostgreSQL-Compatible,	DBA

Task	Description	Skills required
	one role for Import and one role for Export. Assign the corresponding policies to the roles.	
Attach the roles to the Aurora PostgreSQL-Compatible cluster.	Under **Manage roles**, attach the Import and Export roles to the Aurora PostgreSQL cluster.	DBA
Create supporting objects for Aurora PostgreSQL-Compatible.	For the table scripts, use the code under `ext_tbl_scripts` in the *Additional information* section. For the custom function, use the code under `load_external_Table_latest` in the *Additional information* section.	DBA

Process a test file

Task	Description	Skills required
Upload a file into the S3 bucket.	To upload a test file into the S3 bucket, use the console or the following command in AWS CLI. `aws s3 cp /Users/Desktop/ukpost/exttbl/"testing files"/aps s3://s3importtest/inputext/aps`	DBA

Task	Description	Skills required
	As soon as the file is uploaded, a bucket event initiates the Lambda function, which runs the Aurora PostgreSQL-Compatible function.	
Check the data and the log and error files.	The Aurora PostgreSQL-Compatible function loads the files into the main table, and it creates `.log` and `.bad` files in the S3 bucket.	DBA
Monitor the solution.	In the Amazon CloudWatch console, monitor the Lambda function.	DBA

Related resources

- Amazon S3 integration
- Amazon S3
- Working with Amazon Aurora PostgreSQL-Compatible Edition
- AWS Lambda
- Amazon CloudWatch
- AWS Secrets Manager
- Setting up Amazon SNS notifications

Additional information

ext_table_scripts

```
CREATE EXTENSION aws_s3 CASCADE;
CREATE TABLE IF NOT EXISTS meta_EXTERNAL_TABLE
(
    table_name_stg character varying(100) ,
```

```
    table_name character varying(100)  ,
    col_list character varying(1000)  ,
    data_type character varying(100)  ,
    col_order numeric,
    start_pos numeric,
    end_pos numeric,
    no_position character varying(100)  ,
    date_mask character varying(100)  ,
    delimeter character(1)  ,
    directory character varying(100)  ,
    file_name character varying(100)  ,
    header_exist character varying(5)
);
CREATE TABLE IF NOT EXISTS ext_tbl_stg
(
    col1 text
);
CREATE TABLE IF NOT EXISTS error_table
(
    error_details text,
    file_name character varying(100),
    processed_time timestamp without time zone
);
CREATE TABLE IF NOT EXISTS log_table
(
    file_name character varying(50) COLLATE pg_catalog."default",
    processed_date timestamp without time zone,
    tot_rec_count numeric,
    proc_rec_count numeric,
    error_rec_count numeric
);
sample insert scripts of meta data:
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'source_filename', 'character varying', 2, 8, 27, NULL, NULL, NULL, 'databasedev',
 'externalinterface/loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'record_type_identifier', 'character varying', 3, 28, 30, NULL, NULL, NULL,
 'databasedev', 'externalinterface/loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
```

```
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'fad_code', 'numeric', 4, 31, 36, NULL, NULL, NULL, 'databasedev', 'externalinterface/
loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'session_sequence_number', 'numeric', 5, 37, 42, NULL, NULL, NULL, 'databasedev',
 'externalinterface/loaddir/APS', 'NO');
INSERT INTO meta_EXTERNAL_TABLE (table_name_stg, table_name, col_list, data_type,
 col_order, start_pos, end_pos, no_position, date_mask, delimeter, directory,
 file_name, header_exist) VALUES ('F_EX_APS_TRANSACTIONS_STG', 'F_EX_APS_TRANSACTIONS',
 'transaction_sequence_number', 'numeric', 6, 43, 48, NULL, NULL, NULL, 'databasedev',
 'externalinterface/loaddir/APS', 'NO');
```

s3bucketpolicy_for import

```
---Import role policy
--Create an IAM policy to allow, Get,  and list actions on S3 bucket
 {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "s3import",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::s3importtest",
                "arn:aws:s3:::s3importtest/*"
            ]
        }
    ]
}
--Export Role policy
--Create an IAM policy to allow, put,  and list actions on S3 bucket
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "s3export",
            "Action": [
```

```
                "S3:PutObject",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::s3importtest/*"
            ]
        }
    ]
}
```

Sample DB function load_external_tables_latest

```
CREATE OR REPLACE FUNCTION public.load_external_tables(pi_filename text)
 RETURNS character varying
 LANGUAGE plpgsql
AS $function$
/* Loading data from S3 bucket into a APG table */
DECLARE
 v_final_sql TEXT;
 pi_ext_table TEXT;
 r refCURSOR;
 v_sqlerrm text;
 v_chunk numeric;
 i integer;
 v_col_list TEXT;
 v_postion_list CHARACTER VARYING(1000);
 v_len  integer;
 v_delim varchar;
 v_file_name CHARACTER VARYING(1000);
 v_directory CHARACTER VARYING(1000);
 v_table_name_stg CHARACTER VARYING(1000);
 v_sql_col TEXT;
 v_sql TEXT;
 v_sql1 TEXT;
 v_sql2 TEXT;
 v_sql3 TEXT;
 v_cnt integer;
 v_sql_dynamic TEXT;
 v_sql_ins TEXT;
 proc_rec_COUNT integer;
 error_rec_COUNT integer;
 tot_rec_COUNT integer;
```

```
 v_rec_val integer;
 rec record;
 v_col_cnt integer;
 kv record;
 v_val text;
 v_header text;
 j integer;
 ERCODE VARCHAR(5);
 v_region text;
 cr CURSOR FOR
 SELECT distinct DELIMETER,
   FILE_NAME,
   DIRECTORY
 FROM  meta_EXTERNAL_TABLE
 WHERE table_name = pi_ext_table
   AND DELIMETER IS NOT NULL;


 cr1 CURSOR FOR
   SELECT   col_list,
   data_type,
   start_pos,
   END_pos,
   concat_ws('',' ',TABLE_NAME_STG) as TABLE_NAME_STG,
   no_position,date_mask
 FROM  meta_EXTERNAL_TABLE
 WHERE table_name = pi_ext_table
 order by col_order asc;
cr2 cursor FOR
SELECT  distinct table_name,table_name_stg
   FROM  meta_EXTERNAL_TABLE
   WHERE upper(file_name) = upper(pi_filename);


BEGIN
 -- PERFORM utl_file_utility.init();
   v_region := 'us-east-1';
   /* find tab details from file name */


   --DELETE FROM  ERROR_TABLE WHERE file_name= pi_filename;
  -- DELETE FROM  log_table WHERE file_name= pi_filename;
```

```
  BEGIN


    SELECT distinct table_name,table_name_stg INTO strict pi_ext_table,v_table_name_stg
    FROM  meta_EXTERNAL_TABLE
    WHERE upper(file_name) = upper(pi_filename);
  EXCEPTION
    WHEN NO_DATA_FOUND THEN
     raise notice 'error 1,%',sqlerrm;
     pi_ext_table := null;
     v_table_name_stg := null;
       RAISE USING errcode = 'NTFIP' ;
     when others then
         raise notice 'error others,%',sqlerrm;
  END;
  j :=1 ;

 for rec in  cr2
  LOOP




   pi_ext_table     := rec.table_name;
   v_table_name_stg := rec.table_name_stg;
   v_col_list := null;


  IF pi_ext_table IS NOT NULL
   THEN
     --EXECUTE concat_ws('','truncate table  ' ,pi_ext_table) ;
    EXECUTE concat_ws('','truncate table  ' ,v_table_name_stg) ;




        SELECT distinct DELIMETER INTO STRICT v_delim
        FROM  meta_EXTERNAL_TABLE
        WHERE table_name = pi_ext_table;


        IF v_delim IS NOT NULL THEN
      SELECT distinct DELIMETER,
        FILE_NAME,
```

```
      DIRECTORY ,
      concat_ws('',' ',table_name_stg),
      case  header_exist when 'YES' then 'CSV HEADER' else 'CSV' end as header_exist
    INTO STRICT v_delim,v_file_name,v_directory,v_table_name_stg,v_header
    FROM  meta_EXTERNAL_TABLE
    WHERE table_name = pi_ext_table
      AND DELIMETER IS NOT NULL;


    IF    upper(v_delim) = 'CSV'
    THEN
      v_sql := concat_ws('','SELECT aws_s3.table_import_FROM_s3 ( ''',
      v_table_name_stg,''','''',
      ''DELIMITER '''',''''  CSV HEADER QUOTE ''''"'''''', aws_commons.create_s3_uri
( ''',
      v_directory,''',''',v_file_name,''', ''',v_region,'''))');
      ELSE
      v_sql := concat_ws('','SELECT aws_s3.table_import_FROM_s3(''',
          v_table_name_stg, ''','''', ''DELIMITER AS ''''^''''',''',',' 
         aws_commons.create_s3_uri
          ( ''',v_directory, ''',''',
          v_file_name, ''',',
           '''',v_region,''')
         )');
         raise notice 'v_sql , %',v_sql;
      begin
       EXECUTE  v_sql;
      EXCEPTION
        WHEN OTHERS THEN
          raise notice 'error 1';
        RAISE USING errcode = 'S3IMP' ;
      END;


      select count(col_list) INTO v_col_cnt
      from  meta_EXTERNAL_TABLE where table_name = pi_ext_table;







       -- raise notice 'v_sql 2, %',concat_ws('','update ',v_table_name_stg, ' set
col1 = col1||''',v_delim,'''');
```

```
      execute concat_ws('','update ',v_table_name_stg, ' set col1 =
col1||''',v_delim,'''');




      i :=1;
      FOR rec in cr1
      loop
      v_sql1 := concat_ws('',v_sql1,'split_part(col1,''',v_delim,''',', i,')',' as
',rec.col_list,',');
      v_sql2 := concat_ws('',v_sql2,rec.col_list,',');
  --    v_sql3 := concat_ws('',v_sql3,'rec.',rec.col_list,'::',rec.data_type,',');


      case
        WHEN upper(rec.data_type) = 'NUMERIC'
        THEN v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
               THEN null
                ELSE
                coalesce((trim(split_part(col1,''',v_delim,''',',
i,')))::NUMERIC,0)::',rec.data_type,' END as ',rec.col_list,',') ;
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
'YYYYMMDD'
        THEN v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(split_part(col1,''',v_delim,''',',
i,'))),''99990101''),''YYYYMMDD'')::',rec.data_type,' END as ',rec.col_list,',');
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
 'MM/DD/YYYY hh24:mi:ss'
        THEN v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(split_part(col1,''',v_delim,''',',
i,'))),''01/01/9999 0024:00:00''),''MM/DD/YYYY hh24:mi:ss'')::',rec.data_type,' END as
',rec.col_list,',');
         ELSE
```

```
        v_sql3 := concat_ws('',v_sql3,' case WHEN
length(trim(split_part(col1,''',v_delim,''',', i,'))) =0
                THEN null
                 ELSE
                  coalesce((trim(split_part(col1,''',v_delim,''',',
i,'))),''''):: ',rec.data_type,' END as ',rec.col_list,',') ;
      END case;


      i :=i+1;
      end loop;


        -- raise notice 'v_sql 3, %',v_sql3;


      SELECT trim(trailing ' ' FROM v_sql1) INTO v_sql1;
      SELECT trim(trailing ',' FROM v_sql1) INTO v_sql1;


      SELECT trim(trailing ' ' FROM v_sql2) INTO v_sql2;
      SELECT trim(trailing ',' FROM v_sql2) INTO v_sql2;


      SELECT trim(trailing ' ' FROM v_sql3) INTO v_sql3;
      SELECT trim(trailing ',' FROM v_sql3) INTO v_sql3;


      END IF;
     raise notice 'v_delim , %',v_delim;


    EXECUTE concat_ws('','SELECT COUNT(*) FROM ',v_table_name_stg)  INTO v_cnt;


   raise notice 'stg cnt , %',v_cnt;


   /* if upper(v_delim) = 'CSV' then
      v_sql_ins := concat_ws('', ' SELECT * from ' ,v_table_name_stg );
    else
     -- v_sql_ins := concat_ws('',' SELECT ',v_sql1,'  from (select col1 from
' ,v_table_name_stg , ')sub ');
```

```
       v_sql_ins := concat_ws('',' SELECT ',v_sql3,'  from (select col1 from
 ' ,v_table_name_stg , ')sub ');
       END IF;*/


v_chunk := v_cnt/100;




for i in 1..101
loop
     BEGIN
    -- raise notice 'v_sql , %',v_sql;
       -- raise notice 'Chunk number , %',i;
       v_sql_ins := concat_ws('',' SELECT ',v_sql3,'  from (select col1 from
 ' ,v_table_name_stg , ' offset ',v_chunk*(i-1), ' limit ',v_chunk,') sub ');


     v_sql := concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins);
     -- raise notice 'select statement , %',v_sql_ins;
          -- v_sql := null;
     -- EXECUTE concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins, 'offset
 ',v_chunk*(i-1), ' limit ',v_chunk );
     --v_sql := concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins );


     -- raise notice 'insert statement , %',v_sql;


    raise NOTICE 'CHUNK START %',v_chunk*(i-1);
   raise NOTICE 'CHUNK END %',v_chunk;


     EXECUTE v_sql;


  EXCEPTION
       WHEN OTHERS THEN
       -- v_sql_ins := concat_ws('',' SELECT ',v_sql1, '  from (select col1 from
 ' ,v_table_name_stg , ' )sub ');
         -- raise notice 'Chunk number for cursor , %',i;
```

```
    raise NOTICE 'Cursor - CHUNK START %',v_chunk*(i-1);
   raise NOTICE 'Cursor -  CHUNK END %',v_chunk;
         v_sql_ins := concat_ws('',' SELECT ',v_sql3, '  from (select col1 from
 ' ,v_table_name_stg , ' )sub ');


         v_final_sql := REPLACE (v_sql_ins, ''''::text, ''''''::text);
        -- raise notice 'v_final_sql %',v_final_sql;
         v_sql :=concat_ws('','do $a$ declare  r refcursor;v_sql text; i
 numeric;v_conname text;  v_typ  ',pi_ext_table,'[]; v_rec  ','record',';
           begin







           open r for execute ''select col1 from ',v_table_name_stg ,'  offset
 ',v_chunk*(i-1), ' limit ',v_chunk,''';
           loop
           begin
           fetch r into v_rec;
           EXIT WHEN NOT FOUND;





           v_sql := concat_ws('''',''insert into  ',pi_ext_table,' SELECT ',REPLACE
 (v_sql3, ''''::text, ''''''::text) , '  from ( select '''''',v_rec.col1,'''''' as
 col1) v'');
            execute v_sql;


           exception
            when others then
          v_sql := ''INSERT INTO  ERROR_TABLE VALUES (concat_ws('''''''',''''Error
 Name: '''',$$''||SQLERRM||''$$,''''Error State: '''',''''''||
SQLSTATE||'''''',''''record : '''',$$''||v_rec.col1||''$$),''''''||
pi_filename||''''',now())'';


               execute v_sql;
             continue;
           end ;
```

```
            end loop;
            close r;
            exception
            when others then
          raise;
            end ; $a$');
       -- raise notice ' inside excp v_sql %',v_sql;
           execute v_sql;
       --  raise notice 'v_sql %',v_sql;
        END;
   END LOOP;
      ELSE


      SELECT distinct DELIMETER,FILE_NAME,DIRECTORY ,concat_ws('',' ',table_name_stg),
        case  header_exist when 'YES' then 'CSV HEADER' else 'CSV' end as header_exist
        INTO STRICT v_delim,v_file_name,v_directory,v_table_name_stg,v_header
      FROM  meta_EXTERNAL_TABLE
      WHERE table_name = pi_ext_table                  ;
      v_sql := concat_ws('','SELECT aws_s3.table_import_FROM_s3(''',
        v_table_name_stg, ''','''', ''DELIMITER AS ''''#'''' ',v_header,' '',','
       aws_commons.create_s3_uri
        ( ''',v_directory, ''',''',
        v_file_name, ''',',
         '''',v_region,''')
       )');
          EXECUTE  v_sql;


      FOR rec in cr1
      LOOP


       IF rec.start_pos IS NULL AND rec.END_pos IS NULL AND rec.no_position = 'recnum'
       THEN
         v_rec_val := 1;
       ELSE


        case
          WHEN upper(rec.data_type) = 'NUMERIC'
          THEN v_sql1 := concat_ws('',' case WHEN length(trim(substring(COL1,
  ',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
                 THEN null
```

```
                ELSE
                coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1)))::NUMERIC,0)::',rec.data_type,' END as
',rec.col_list,',') ;
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
'YYYYMMDD'
        THEN v_sql1 := concat_ws('','case WHEN length(trim(substring(COL1,
',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1))),''99990101''),''YYYYMMDD'')::',rec.data_type,'
END as ',rec.col_list,',');
        WHEN UPPER(rec.data_type) = 'TIMESTAMP WITHOUT TIME ZONE' AND rec.date_mask =
'YYYYMMDDHH24MISS'
        THEN v_sql1 := concat_ws('','case WHEN length(trim(substring(COL1,
',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
               THEN null
                ELSE
                to_date(coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1))),''9999010100240000''),''YYYYMMDDHH24MISS'')::',rec.data_
END as ',rec.col_list,',');
         ELSE
       v_sql1 := concat_ws('',' case WHEN length(trim(substring(COL1,
',rec.start_pos ,',', rec.END_pos,'-',rec.start_pos ,'+1))) =0
               THEN null
                ELSE
                 coalesce((trim(substring(COL1, ',rec.start_pos ,',',
rec.END_pos,'-',rec.start_pos ,'+1))),'''')::',rec.data_type,' END as
',rec.col_list,',') ;
      END case;


     END IF;
     v_col_list := concat_ws('',v_col_list ,v_sql1);
    END LOOP;




         SELECT trim(trailing ' ' FROM v_col_list) INTO v_col_list;
         SELECT trim(trailing ',' FROM v_col_list) INTO v_col_list;
```

```
          v_sql_col   :=  concat_ws('',trim(trailing ',' FROM v_col_list) , ' FROM
 ',v_table_name_stg,' WHERE col1 IS NOT NULL AND length(col1)>0 ');




          v_sql_dynamic := v_sql_col;


          EXECUTE  concat_ws('','SELECT COUNT(*) FROM ',v_table_name_stg) INTO v_cnt;




        IF v_rec_val = 1 THEN
            v_sql_ins := concat_ws('',' select row_number() over(order by ctid) as
line_number ,' ,v_sql_dynamic) ;


        ELSE
              v_sql_ins := concat_ws('',' SELECT' ,v_sql_dynamic) ;
          END IF;


    BEGIN
      EXECUTE concat_ws('','insert into  ', pi_ext_table ,' ', v_sql_ins);
          EXCEPTION
             WHEN OTHERS THEN
         IF v_rec_val = 1 THEN
                 v_final_sql := ' select row_number() over(order by ctid) as
line_number ,col1 from ';
               ELSE
                v_final_sql := ' SELECT col1 from';
              END IF;
      v_sql :=concat_ws('','do $a$ declare  r refcursor;v_rec_val numeric :=
',coalesce(v_rec_val,0),';line_number numeric; col1 text; v_typ  ',pi_ext_table,'[];
v_rec  ',pi_ext_table,';
            begin
            open r for execute ''',v_final_sql, ' ',v_table_name_stg,' WHERE col1 IS
NOT NULL AND length(col1)>0 '' ;
            loop
            begin
            if   v_rec_val = 1 then
            fetch r into line_number,col1;
```

```
            else
            fetch r into col1;
            end if;


            EXIT WHEN NOT FOUND;
             if v_rec_val = 1 then
             select line_number,',trim(trailing ',' FROM v_col_list) ,' into v_rec;
             else
               select ',trim(trailing ',' FROM v_col_list) ,' into v_rec;
             end if;


            insert into  ',pi_ext_table,' select v_rec.*;
             exception
             when others then
              INSERT INTO  ERROR_TABLE VALUES (concat_ws('''',''Error Name:
'',SQLERRM,''Error State: '',SQLSTATE,''record : '',v_rec),''',pi_filename,''',now());
              continue;
             end ;
              end loop;
            close r;
             exception
             when others then
             raise;
             end ; $a$');
        execute v_sql;


    END;


        END IF;


  EXECUTE concat_ws('','SELECT COUNT(*) FROM  ' ,pi_ext_table)   INTO proc_rec_COUNT;


  EXECUTE concat_ws('','SELECT COUNT(*) FROM  error_table WHERE file_name
=''',pi_filename,''' and processed_time::date = clock_timestamp()::date')  INTO
error_rec_COUNT;


  EXECUTE concat_ws('','SELECT COUNT(*) FROM ',v_table_name_stg)   INTO tot_rec_COUNT;
```

```
   INSERT INTO  log_table values(pi_filename,now(),tot_rec_COUNT,proc_rec_COUNT,
 error_rec_COUNT);


   raise notice 'v_directory, %',v_directory;


   raise notice 'pi_filename, %',pi_filename;


   raise notice 'v_region, %',v_region;


  perform aws_s3.query_export_to_s3('SELECT
 replace(trim(substring(error_details,position(''('' in
 error_details)+1),'')''),'','','';''),file_name,processed_time FROM  error_table WHERE
 file_name = '''||pi_filename||'''',
   aws_commons.create_s3_uri(v_directory, pi_filename||'.bad', v_region),
   options :='FORmat csv, header, delimiter $$,$$'
   );


raise notice 'v_directory, %',v_directory;


   raise notice 'pi_filename, %',pi_filename;


   raise notice 'v_region, %',v_region;


  perform aws_s3.query_export_to_s3('SELECT * FROM  log_table WHERE file_name = '''||
pi_filename||'''',
   aws_commons.create_s3_uri(v_directory, pi_filename||'.log', v_region),
   options :='FORmat csv, header, delimiter $$,$$'
   );




   END IF;
 j := j+1;
```

```
 END LOOP;


       RETURN 'OK';
EXCEPTION
    WHEN  OTHERS THEN
  raise notice 'error %',sqlerrm;
   ERCODE=SQLSTATE;
   IF ERCODE = 'NTFIP' THEN
     v_sqlerrm := concat_Ws('',sqlerrm,'No data for the filename');
   ELSIF ERCODE = 'S3IMP' THEN
    v_sqlerrm := concat_Ws('',sqlerrm,'Error While exporting the file from S3');
   ELSE
      v_sqlerrm := sqlerrm;
   END IF;


 select distinct directory into v_directory from  meta_EXTERNAL_TABLE;




 raise notice 'exc v_directory, %',v_directory;


   raise notice 'exc pi_filename, %',pi_filename;


   raise notice 'exc v_region, %',v_region;


  perform aws_s3.query_export_to_s3('SELECT * FROM  error_table WHERE file_name = '''||
pi_filename||'''',
   aws_commons.create_s3_uri(v_directory, pi_filename||'.bad', v_region),
   options :='FORmat csv, header, delimiter $$,$$'
   );
    RETURN null;
END;
$function$
```

Migrate function-based indexes from Oracle to PostgreSQL

Veeranjaneyulu Grandhi and Navakanth Talluri, Amazon Web Services

Summary

Indexes are a common way to enhance database performance. An index allows the database server to find and retrieve specific rows much faster than it could without an index. But indexes also add overhead to the database system as a whole, so they should be used sensibly. Function-based indexes, which are based on a function or expression, can involve multiple columns and mathematical expressions. A function-based index improves the performance of queries that use the index expression.

Natively, PostgreSQL doesn't support creating function-based indexes using functions that have volatility defined as stable. However, you can create similar functions with volatility as `IMMUTABLE` and use them in index creation.

An `IMMUTABLE` function cannot modify the database, and it's guaranteed to return the same results given the same arguments forever. This category allows the optimizer to pre-evaluate the function when a query calls it with constant arguments.

This pattern helps in migrating the Oracle function-based indexes when used with functions such as `to_char`, `to_date`, and `to_number` to the PostgreSQL equivalent.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account
- A source Oracle database instance with the listener service set up and running
- Familiarity with PostgreSQL databases

Limitations

- Database size limit is 64 TB.
- Functions used in index creation must be IMMUTABLE.

Product versions

- All Oracle database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c
- PostgreSQL versions 9.6 and later

Architecture

Source technology stack

- An Oracle database on premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance, or an Amazon RDS for Oracle DB instance

Target technology stack

- Any PostgreSQL engine

Tools

- **pgAdmin 4** is an open source management tool for Postgres. The pgAdmin 4 tool provides a graphical interface for creating, maintaining, and using database objects.
- **Oracle SQL Developer** is an integrated development environment (IDE) for developing and managing Oracle Database in both traditional and cloud deployments.

Epics

Create a function-based index using a default function

Task	Description	Skills required

Create a function-based index on a column using the to_char function.	Use the following code to create the function-based index.
	`postgres=# create table funcindex( col1 timestamp without time zone);`
	`CREATE TABLE`
		DBA, App developer

Task	Description	Skills required

	<pre>postgres=# insert
	 into funcindex values
	 (now());
	INSERT 0 1
	postgres=# select *
	 from funcindex;
	            col1
	---------------------- 	
	------ 	
	 2022-08-09 16:00:57.
	77414
	(1 rows)
	
	postgres=# create
	 index funcindex_idx
	 on funcindex(to_char(
	col1,'DD-MM-YYYY
	 HH24:MI:SS'));
	ERROR: functions in
	 index expression must
	 be marked IMMUTABLE</pre> **Note** PostgreSQL doesn’t allow creating a function-based index without the `IMMUTABLE` clause.	

Check the volatility of the function.	To check the function volatility, use the code in the *Additional information* section.	DBA

Create function-based indexes using a wrapper function

Task	Description	Skills required
Create a wrapper function.	To create a wrapper function, use the code in the *Additional information section*.	PostgreSQL developer
Create an index by using the wrapper function.	Use the code in the *Additional information* section to create a user-defined function with the keyword `IMMUTABLE` in the same schema as the application, and refer to it in the index-creation script. If a user-defined function is created in a common schema (from the previous example), update the `search_path` as shown. `ALTER ROLE <ROLENAME> set search_path=$user, COMMON;`	DBA, PostgreSQL developer

Validate index creation

Task	Description	Skills required
Validate index creation.	Validate that the index needs to be created, based on query access patterns.	DBA
Validate that the index can be used.	To check whether the function-based index is picked up by the PostgreSQ	DBA

Task	Description	Skills required
	L Optimizer, run an SQL statement using explain or explain analyze. Use the code in the *Additional information* section. If possible, gather the table statistics as well. **Note** If you notice the explain plan, PostgreSQL optimizer has chosen a function-based index because of the predicate condition.	

Related resources

- Function-based indexes (Oracle documentation)
- Indexes on Expressions (PostgreSQL documentation)
- PostgreSQL volatility (PostgreSQL documentation)
- PostgreSQL search_path (PostgreSQL documentation)
- Oracle Database 19c to Amazon Aurora PostgreSQL Migration Playbook

Additional information

Create a wrapper function

```
CREATE OR REPLACE FUNCTION myschema.to_char(var1 timestamp without time zone, var2
 varchar) RETURNS varchar AS $BODY$ select to_char(var1, 'YYYYMMDD'); $BODY$ LANGUAGE
 sql IMMUTABLE;
```

Create an index by using the wrapper function

```
postgres=# create function common.to_char(var1 timestamp without time zone, var2
 varchar) RETURNS varchar AS $BODY$ select to_char(var1, 'YYYYMMDD'); $BODY$ LANGUAGE
 sql IMMUTABLE;
CREATE FUNCTION
postgres=# create index funcindex_idx on funcindex(common.to_char(col1,'DD-MM-YYYY
 HH24:MI:SS'));
CREATE INDEX
```

Check the volatility of the function

```
SELECT DISTINCT p.proname as "Name",p.provolatile as "volatility" FROM
 pg_catalog.pg_proc p
 LEFT JOIN pg_catalog.pg_namespace n ON n.oid = p.pronamespace
 LEFT JOIN pg_catalog.pg_language l ON l.oid = p.prolang
 WHERE n.nspname OPERATOR(pg_catalog.~) '^(pg_catalog)$' COLLATE pg_catalog.default AND
 p.proname='to_char'GROUP BY p.proname,p.provolatile
ORDER BY 1;
```

Validate that the index can be used

```
explain analyze <SQL>



postgres=# explain select col1 from funcindex where common.to_char(col1,'DD-MM-YYYY
 HH24:MI:SS') = '09-08-2022 16:00:57';
                                                       QUERY PLAN
-----------------------------------------------------------------------------------------------------
 Index Scan using funcindex_idx on funcindex  (cost=0.42..8.44 rows=1 width=8)
   Index Cond: ((common.to_char(col1, 'DD-MM-YYYY HH24:MI:SS'::character
 varying))::text = '09-08-2022 16:00:57'::text)
(2 rows)
```

Migrate Oracle native functions to PostgreSQL using extensions

Pinesh Singal, Amazon Web Services

Summary

This migration pattern provides step-by-step guidance for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle database instance to an Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition database by modifying the `aws_oracle_ext` and `orafce` extensions to PostgreSQL (`psql`) native built-in code. This will save processing time.

The pattern describes an offline manual migration strategy with no downtime for a multi-terabyte Oracle source database with a high number of transactions.

The migration process uses AWS Schema Conversion Tool (AWS SCT) with the `aws_oracle_ext` and `orafce` extensions to convert an Amazon RDS for Oracle database schema to an Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database schema. Then the code is manually changed to PostgreSQL supported native `psql` built-in code. This is because the extension calls impact code processing on the PostgreSQL database server, and not all the extension code is fully complaint or compatible with PostgreSQL code.

This pattern primarily focuses on manually migrating SQL codes using AWS SCT and the extensions `aws_oracle_ext` and `orafce`. You convert the already used extensions into native PostgreSQL (`psql`) built-ins. Then you remove all references to the extensions and convert the codes accordingly.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Operating system (Windows or Mac) or Amazon EC2 instance (up and running)
- Orafce

Limitations

Not all Oracle functions using `aws_oracle_ext` or `orafce` extensions can be converted to native PostgreSQL functions. It might need manual rework so as to compile it with PostgreSQL libraries.

One drawback of using AWS SCT extensions is its slow performance in running and fetching the results. Its cost can be understood from simple [PostgreSQL EXPLAIN plan] (execution plan of a statement) on the Oracle `SYSDATE` function migration to the PostgreSQL `NOW()` function between all three codes (`aws_oracle_ext`, `orafce`, and `psql` default), as explained in the *Performance comparison check* section in the attached document.

Product versions

- **Source:** Amazon RDS for Oracle database 10.2 and later (for 10.x), 11g (11.2.0.3.v1 and later) and up to 12.2, 18c, and 19c (and later) for Enterprise Edition, Standard Edition, Standard Edition 1, and Standard Edition 2
- **Target**: Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database 9.4 and later (for 9.x), 10.x, 11.x, 12.x, 13.x, and 14.x (and later)
- **AWS SCT**: Latest version (this pattern was tested with 1.0.632)
- **Orafce**: Latest version (this pattern was tested with 3.9.0)

Architecture

Source technology stack

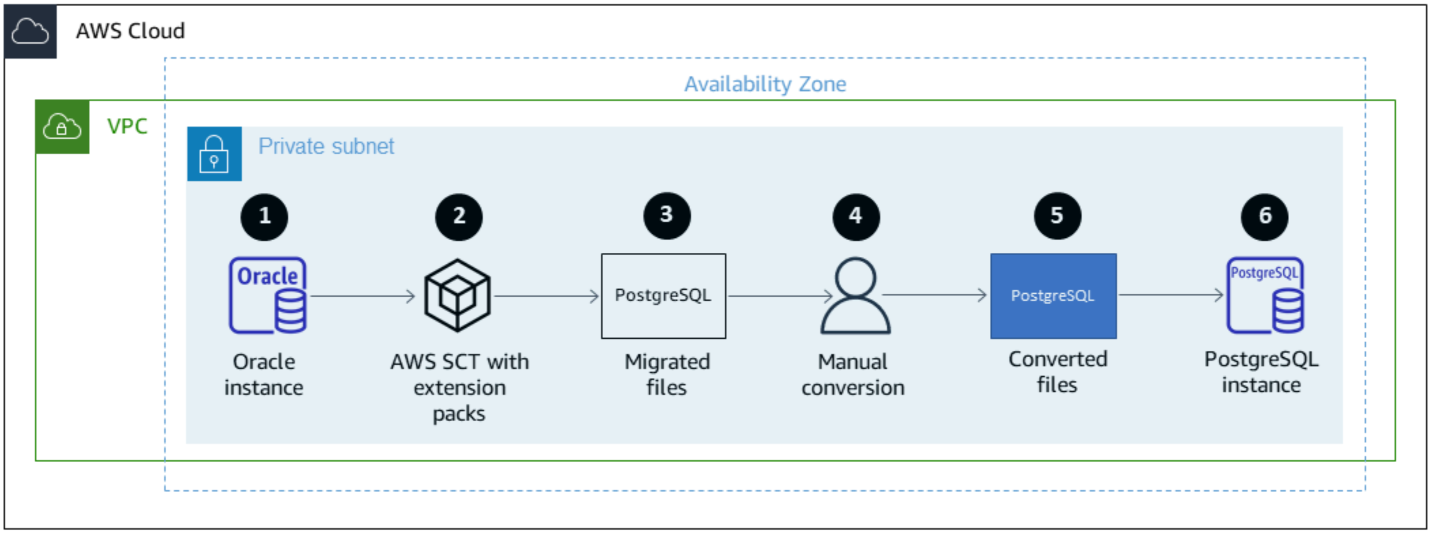
- An Amazon RDS for Oracle database instance with version 12.1.0.2.v18

Target technology stack

- An Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database instance with version 11.5

Database migration architecture

The following diagram represents the database migration architecture between the source Oracle and target PostgreSQL databases. The architecture involves AWS Cloud, a virtual private cloud (VPC), Availability Zones, a private subnet, an Amazon RDS for Oracle database, AWS SCT, an Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database, extensions for Oracle (`aws_oracle_ext` and `orafce`), and structured query language (SQL) files.



1. Launch Amazon RDS for Oracle DB instance (source DB).
2. Use AWS SCT with the `aws_oracle_ext` and `orafce` extension packs to convert the source code from Oracle to PostreSQL.
3. The conversion produces PostgreSQL-supported migrated .sql files.
4. Manually convert the non-converted Oracle extension codes to PostgreSQL (`psql`) codes.
5. The manual conversion produces PostgreSQL-supported converted .sql files.
6. Run these .sql files on your Amazon RDS for PostgreSQL DB instance (target DB).

Tools

Tools

AWS services

- AWS SCT - AWS Schema Conversion Tool (AWS SCT) converts your existing database schema from one database engine to another. You can convert relational Online Transactional Processing (OLTP) schema, or data warehouse schema. Your converted schema is suitable for an Amazon RDS for MySQL DB instance, an Amazon Aurora DB cluster, an Amazon RDS for PostgreSQL DB instance, or an Amazon Redshift cluster. The converted schema can also be used with a database on an Amazon EC2 instance or stored as data in an Amazon S3 bucket.

  AWS SCT provides a project-based user interface to automatically convert the database schema of your source database into a format compatible with your target Amazon RDS instance.

You can use AWS SCT to do migration from an Oracle source database to any of the targets listed preceding. Using AWS SCT, you can export the source database object definitions such as schema, views, stored procedures, and functions.

You can use AWS SCT to convert data from Oracle to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition.

In this pattern, you use AWS SCT to convert and migrate Oracle code into PostgreSQL using the extensions `aws_oracle_ext` and `orafce`, and manually migrating the extension codes into `psql` default or native built-in code.

- The AWS SCT extension pack is an add-on module that emulates functions present in the source database that are required when converting objects to the target database. Before you can install the AWS SCT extension pack, you need to convert your database schema.

  When you convert your database or data warehouse schema, AWS SCT adds an additional schema to your target database. This schema implements SQL system functions of the source database that are required when writing your converted schema to your target database. This additional schema is called the extension pack schema.

  The extension pack schema for OLTP databases is named according to the source database. For Oracle databases, the extension pack schema is `AWS_ORACLE_EXT`.

Other tools

- Orafce – Orafce is a module that implements Oracle compatible functions, data types, and packages. It's an open-source tool with a Berkeley Source Distribution (BSD) license so that anyone can use it. The `orafce` module is useful for migrating from Oracle to PostgreSQL because it has many Oracle functions implemented in PostgreSQL.

Code

For a list of all commonly used and migrated code from Oracle to PostgreSQL to avoid AWS SCT extension code usage, see the attached document.

Epics

Configure the Amazon RDS for Oracle source database

Task	Description	Skills required
Create the Oracle database instance.	Create an Amazon RDS for Oracle or Aurora PostgreSQL-Compatible database instance from the Amazon RDS console.	General AWS, DBA
Configure the security groups.	Configure inbound and outbound security groups.	General AWS
Create the database.	Create the Oracle database with needed users and schemas.	General AWS, DBA
Create the objects.	Create objects and insert data in schema.	DBA

Configure the Amazon RDS for PostgreSQL target database

Task	Description	Skills required
Create the PostgreSQL database instance.	Create an Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL database instance from the Amazon RDS console.	General AWS, DBA
Configure the security groups.	Configure inbound and outbound security groups.	General AWS
Create the database.	Create the PostgreSQL database with needed users and schemas.	General AWS, DBA

Task	Description	Skills required
Validate the extensions.	Make sure that `aws_oracle_ext` and `orafce` are installed and configured correctly in the PostgreSQL database.	DBA
Verify that the PostgreSQL database is available.	Make sure that the PostgreSQL database is up and running.	DBA

Migrate the Oracle schema into PostgreSQL using AWS SCT and the extensions

Task	Description	Skills required
Install AWS SCT.	Install the latest version of AWS SCT.	DBA
Configure AWS SCT.	Configure AWS SCT with Java Database Connectivity (JDBC) drivers for Oracle (`ojdbc8.jar`) and PostgreSQL (`postgresql-42.2.5.jar`).	DBA
Enable the AWS SCT extension pack or template.	Under AWS SCT **Project Settings**, enable built-in function implementation with the `aws_oracle_ext` and `orafce` extensions for the Oracle database schema.	DBA
Convert the schema.	In AWS SCT, choose **Convert Schema** to convert the schema from Oracle to PostgreSQL and generate the .sql files.	DBA

Convert AWS SCT extension code to psql code

Task	Description	Skills required
Manually convert the code.	Manually convert each line of extension-supported code into `psql` default built-in code, as detailed in the attached document. For example, change `AWS_ORACLE_EXT.SYSDATE()` or `ORACLE.SYSDATE()` to `NOW()`.	DBA
Validate the code	(Optional) Validate each line of code by temporary running it in the PostgreSQL database.	DBA
Create objects in the PostgreSQL database.	To create objects in the PostgreSQL database, run the .sql files that were generated by AWS SCT and modified in the previous two steps.	DBA

Related resources

- Database
  - [Oracle on Amazon RDS]
  - [PostgreSQL on Amazon RDS]
  - [Working with Amazon Aurora PostgreSQL]
  - [PostgreSQL EXPLAIN plan]
- AWS SCT
  - [AWS Schema Conversion Tool Overview]
  - [AWS SCT User Guide]

  - Using the AWS SCT user interface
  - Using Oracle Database as a source for AWS SCT
- Extensions for AWS SCT
  - Using the AWS SCT extension pack
  - Oracle functionality (en)
  - PGXN orafce
  - GitHub orafce

Additional information

For more information, follow the detailed commands, with syntax and examples, for manually converting code in the attached document.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Migrate a Db2 database from Amazon EC2 to Aurora MySQL-Compatible by using AWS DMS

Pinesh Singal, Amazon Web Services

Summary

After you migrate your IBM Db2 for LUW database to Amazon Elastic Compute Cloud (Amazon EC2), consider re-architecting the database by moving to an Amazon Web Services (AWS) cloud-native database. This pattern covers migrating an IBM Db2 for LUW database running on an Amazon EC2 instance to an Amazon Aurora MySQL-Compatible Edition database on AWS.

The pattern describes an online migration strategy with minimal downtime for a multi-terabyte Db2 source database with a high number of transactions.

This pattern uses AWS Schema Conversion Tool (AWS SCT) to convert the Db2 database schema to an Aurora MySQL-Compatible schema. Then the pattern uses AWS Database Migration Service (AWS DMS) to migrate data from the Db2 database to the Aurora MySQL-Compatible database. Manual conversions will be required for the code that isn't converted by AWS SCT.

Prerequisites and limitations

Prerequisites

- An active AWS account with a virtual private cloud (VPC)
- AWS SCT
- AWS DMS

Product versions

- AWS SCT latest version
- Db2 for Linux version 11.1.4.4 and later

Architecture

Source technology stack

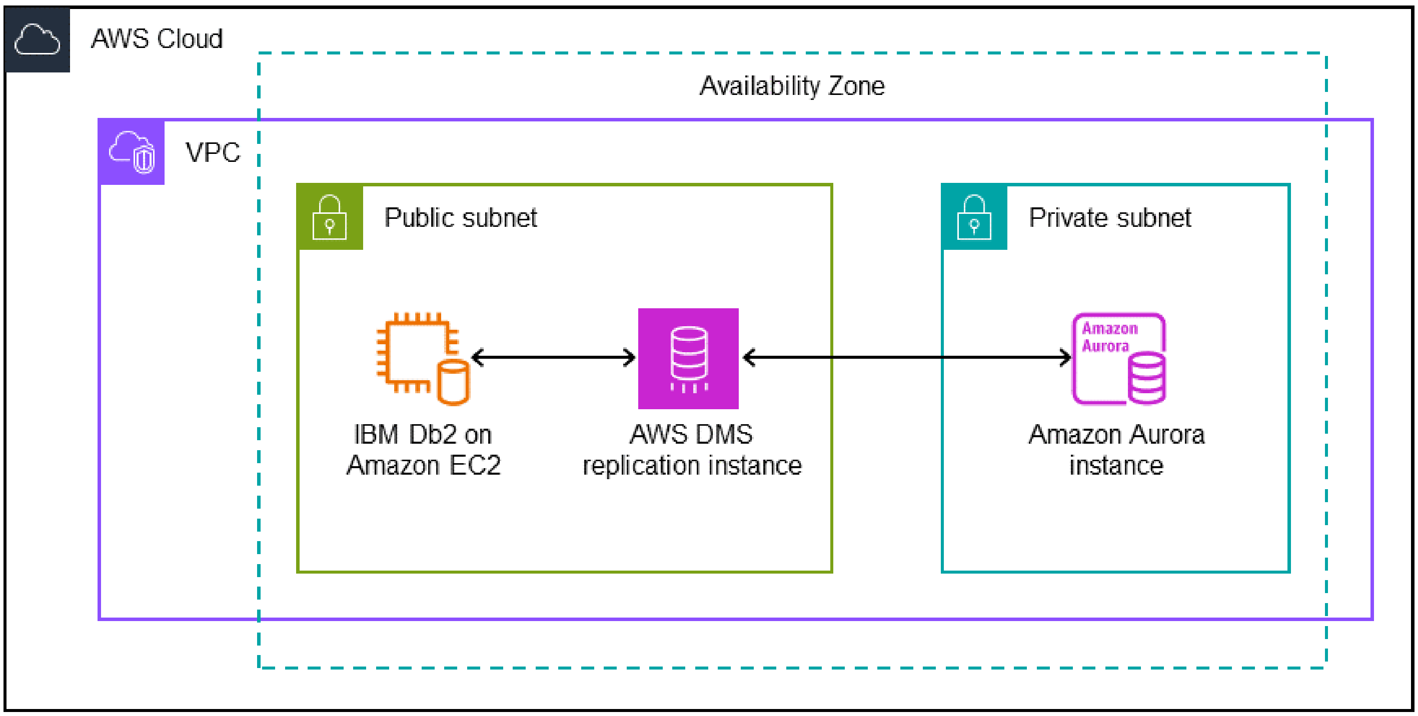
- DB2/Linux x86-64 bit mounted on an EC2 instance

Target technology stack

- An Amazon Aurora MySQL-Compatible Edition database instance

Source and target architecture

The following diagram shows the data migration architecture between the source Db2 and target Aurora MySQL-Compatible databases. The architecture on the AWS Cloud includes a virtual private cloud (VPC) (Virtual Private Cloud), an Availability Zone, a public subnet for the Db2 instance and the AWS DMS replication instance, and a private subnet for the Aurora MySQL-Compatible database.



Tools

AWS services

- [Amazon Aurora](#) is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL.
- [AWS Database Migration Service (AWS DMS)](#) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- [Amazon Elastic Compute Cloud (Amazon EC2)](#) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.

- [AWS Schema Conversion Tool (AWS SCT)](#) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. AWS SCT supports as a source IBM Db2 for LUW versions 9.1, 9.5, 9.7, 10.1, 10.5, 11.1, and 11.5.

Best practices

For best practices, see Best practices for AWS Database Migration Service.

Epics

Configure the source IBM Db2 database

Task	Description	Skills required
Create the IBM Db2 database on Amazon EC2.	You can create an IBM Db2 database on an EC2 instance by using an Amazon Machine Image (AMI) from AWS Marketplace or by installin g Db2 software on an EC2 instance. Launch an EC2 instance by selecting an AMI for IBM Db2 (for example, IBM Db2 v11.5.7 RHEL 7.9), which is similar to an on-premises database.	DBA, General AWS
Configure security groups.	Configure the VPC security group inbound rules for SSH (Secure Shell) and TCP with port 22 and 50000, respectiv ely.	General AWS
Create the database instance.	Create a new instance (user) and database (schema), or use the default `db2inst1`	DBA

Task	Description	Skills required
	instance and sample database. 1. Connect to the EC2 instance by using the terminal to connect to the Db2 database. Alternatively, you can install any DB client software that will connect to the Db2 database. 2. To set the password of the db2inst1 user, run the command `sudo passwd db2inst1`. 3. To connect to the db2inst1 instance, run the command `sudo su - db2inst1`. 4. To connect to the Db2 database, run the command `db2`. 5. To connect to the sample database, use the command `connect to sample`. Alternatively, connect to the database that you created. 6. After you connect to the database instance, create objects and insert data into these objects by using Db2 SQL statements.	

Task	Description	Skills required
Confirm that the Db2 DB instance is available.	To confirm that the Db2 database instance is up and running, use the `Db2pd -` command.	DBA

Configure the target Aurora MySQL-Compatible database

Task	Description	Skills required
Create the Aurora MySQL-Compatible database.	Create an Amazon Aurora with MySQL compatibility Database from AWS RDS service • Create a database on Amazon Aurora with MySQL compatibility and version of your choice, e.g. Aurora (MySQL)–5.6.10a • Install MySQL Workbench application or your preferred DB client software which allows you to connect to MySQL database	DBA, General AWS
Configure security groups.	Configure the VPC security group inbound rules for SSH and TCP connections.	General AWS
Confirm that the Aurora database is available.	To make sure that the Aurora MySQL-Compatible database is up and running, do the following:	DBA

Task	Description	Skills required
	1. Connect to the EC2 instance through SSH. 2. Configure and connect to the Aurora MySQL-Compatible instance from MySQL Workbench. Use the endpoint as the hostname, as shown in the following example. `mysql-cluster-instance-1.cokmvis0v46q.us-east-1.rds.amazonaws.com` 3. Create and connect to the new schema (for example, `mysql-sample-db2`). 4. Run the MySQL statements to check the schemas and objects in the database.	

Configure and run AWS SCT

Task	Description	Skills required
Install AWS SCT.	Download and install the latest version of [AWS SCT](https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Installing.html) (the current latest version 1.0.628).	General AWS
Configure AWS SCT.	1. Download the Java Database Connectivity (JDBC) drivers for IBM Db2	General AWS

Task	Description	Skills required
	(4.22.X version) and MySQL (8.x). 2. To configure the drivers in AWS SCT, choose **Settings**, **Global settings**, **Drivers**.	
Create an AWS SCT project.	Create an AWS SCT project and report that uses Db2 for LUW as the source DB engine and Aurora MySQL-Compatible for the target DB engine. To identify the privileges needed to connect to a Db2 for LUW database, see [Using Db2 LUW as a source for AWS SCT]().	General AWS

Task	Description	Skills required
Validate the objects.	Choose **Load schema**, validate the objects. Update any incorrect objects on the target database: 1. Connect to the Amazon Aurora MySQL-Compatible server by providing the connection details, and choose **Test connection**. Both source and target connections must be successful before AWS SCT can start the migration report. 2. After the report is completed, enter the schema to be converted, and choose **Finish**. AWS SCT lists any source and target objects that are converted and have errors. 3. Review the errors, and clear them manually. 4. After all errors are cleared, open the context (right-cl ick) menu for the schema, and choose **Load schema**. 5. Choose **Apply to database**. 6. In MySQL Workbench , connect to the Aurora MySQL-Compatible	DBA, General AWS

Task	Description	Skills required
	database, and check the schema and objects.	

Configure and run AWS DMS

Task	Description	Skills required
Create a replication instance.	Sign in to the AWS Management Console, navigate to the AWS DMS service, and create a replicati on instance with valid settings for the VPC security group that you configure d for the source and target databases.	General AWS
Create endpoints.	Create the source endpoint for the Db2 database, and create the target endpoint for the Aurora MySQL-Com patible database: 1. Create an endpoint for IBM Db2 as the source by choosing **Select RDS DB instance** and then choosing the Db2 instance that you created. The endpoint configuration details will be automatic ally populated. 2. In the endpoint-specific settings, add the following	General AWS

Task	Description	Skills required
	extra connection attributes. `CurrentLSN=<scan>; MaxKBytesPerRead=64; SetDataCaptureChanges=true` If you don’t mention these attributes, the source endpoint test connection will not be successful. For more information, see Using IBM Db2 LUW as a source for AWS DMS. 3. Create an endpoint for Aurora MySQL-Compatible as the target by choosing **Select RDS DB instance** and then choosing the Aurora MySQL-Compatible instance that you created. The endpoint configuration details will be automatically populated. For more information, see Using a MySQL-compatible database as a target for AWS Database Migration Service. 4. Test the source and target endpoints. Confirm that both are successful and available	

Task	Description	Skills required
	5. If the test fails, make sure that the security group inbound rules are valid.	

Task	Description	Skills required
Create migration tasks.	Create a single migration task or multiple migration tasks for full load and CDC or Data validation: 1. To create a database migration task, choose the replication instance, source database endpoint, target database endpoint. Specify the migration type as **Migrate existing data** (full load), **Replicate data changes only (CDC)**, or **Migrate existing data and replicate ongoing changes** (full load and CDC). 2. Under **Table mappings**, you can configure selection rules and transformation rules in GUI or JSON formats. 3. Under **Selection rules**, select the schema, enter table name, and select Action (Include/Exclude) to be configured (for example, Schema: SAMPLE; Table name: %, Action: Include). 4. Under **Transformation rules**, select the target (Schema, Table, or Column). Select the schema name, and choose	General AWS

Task	Description	Skills required
	the action (case, prefix, suffix); for example, Target: Schema; `mysql-sample-db`; Action: Make lowercase. 5. Turn on Amazon CloudWatch Logs monitoring.	
Plan the production run.	Confirm downtime with stakeholders such as applicati on owners to run AWS DMS in production systems.	Migration lead
Run the migration tasks.	1. Start the AWS DMS task that has a status of **Ready**. 2. Monitor the migration task logs in Amazon CloudWatc h Logs for any errors.	General AWS

Task	Description	Skills required
Validate the data.	Review migration task results and data in the source Db2 and target MySQL databases: 1. If the status is **Load complete ongoing replication**, the full load with CDC data migration is completed, and validation is ongoing. 2. Connect to the Aurora MySQL-Compatible database, and check the data. 3. Check the ongoing changes by inserting or updating data in the Db2 database.	DBA
Stop migration tasks.	After data validation is successfully completed, stop the validation migration tasks.	General AWS

Troubleshooting

Issue	Solution
AWS SCT source and target test connections are failing.	Configure JDBC driver versions and VPC security group inbound rules to accept the incoming traffic.
The Db2 source endpoint test run fails.	Configure the extra connection setting `CurrentLSN=<scan>;` .

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td>The AWSDMS task fails to connect to the Db2 source, and the following error is returned.

<code>database is recoverable if either or both of the database configura tion parameters LOGARCHMETH1 and LOGARCHMETH2 are set to ON</code></td><td>To avoid the error, run the following commands:

1. <code>$ db2 update db cfg for sample using LOGARCHMETH1 DISK:/home/db2inst1/logs</code>
2. <code>$ db2stop</code>
3. <code>$ db2start</code>
4. <code>$ db2 connect to sample</code>
<pre>SQL1116N A connection to or
 activation of database "SAMPLE"
 cannot be made because of BACKUP
 PENDING. SQLSTATE=57019</pre>5. <code>$ db2 backup database sample to ../logs</code>
<pre>SQL2036N The path for the file or
 device "../logs" is not valid</pre>6. <code>$ cd</code>
7. <code>$ pwd</code>
<pre> /home/db2inst1</pre>8. <code>$ mkdir /tmp/backup</code>
9. <code>$ db2 backup database sample to /tmp/backup</code>
<pre>Backup successful. The timestamp for
 this backup image is : 201905300
84921</pre>10 <code>$ db2 connect to sample</code>
<pre>Database Connection Information</pre></td></tr>
</table>

Issue	Solution
	```Database server = DB2/LINUX 9.7.1 SQL authorization ID = DB2INST1 Local database alias = SAMPLE```

Related resources

Amazon EC2

- Amazon EC2
- Amazon EC2 User Guides

Databases

- IBM Db2 Database
- Amazon Aurora
- Working with Amazon Aurora MySQL

AWS SCT

- AWS DMS Schema Conversion
- AWS Schema Conversion Tool User Guide
- Using the AWS SCT user interface
- Using IBM Db2 LUW as a source for AWS SCT

AWS DMS

- AWS Database Migration Service
- AWS Database Migration Service User Guide
- Sources for data migration
- Targets for data migration
- AWS Database Migration Service and AWS Schema Conversion Tool now support IBM Db2 LUW as a source (blog post)

- [Migrating Applications Running Relational Databases to AWS]

Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS

Umamaheswara Nooka, Amazon Web Services

Summary

This pattern describes how to use AWS Database Migration Service (AWS DMS) to migrate a Microsoft SQL Server database hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance to an Amazon DocumentDB (with MongoDB compatibility) database.

The AWS DMS replication task reads the table structure of the SQL Server database, creates the corresponding collection in Amazon DocumentDB, and performs a full-load migration.

You can also use this pattern to migrate an on-premises SQL Server or an Amazon Relational Database Service (Amazon RDS) for SQL Server DB instance to Amazon DocumentDB. For more information, see the guide Migrating Microsoft SQL Server databases to the AWS Cloud on the AWS Prescriptive Guidance website.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing SQL Server database on an EC2 instance.
- Fixed database (**db_owner**) role assigned to AWS DMS in the SQL Server database. For more information, see Database-level roles in the SQL Server documentation.
- Familiarity with using the `mongodump`, `mongorestore`, `mongoexport`, and `mongoimport` utilities to move data in and out of an Amazon DocumentDB cluster.
- Microsoft SQL Server Management Studio, installed and configured.

Limitations

- The cluster size limit in Amazon DocumentDB is 64 TB. For more information, see Cluster limits in the Amazon DocumentDB documentation.
- AWS DMS doesn't support the merging of multiple source tables into a single Amazon DocumentDB collection.

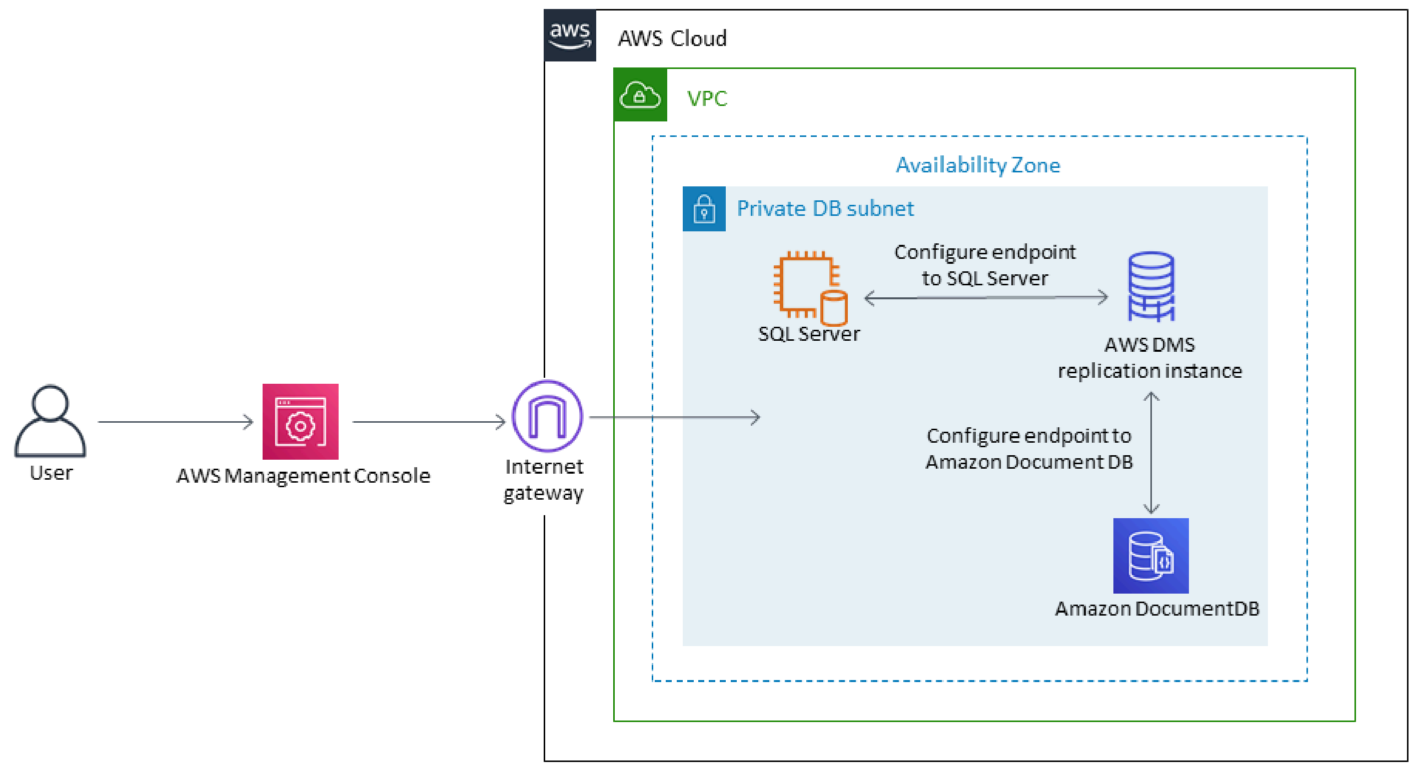
- If AWS DMS processes any changes from a source table without a primary key, it will ignore large object (LOB) columns in the source table.

Architecture

Source technology stack

- Amazon EC2

Target architecture



Target technology stack

- Amazon DocumentDB

Tools

- [AWS DMS](#) – AWS Database Migration Service (AWS DMS) helps you migrate databases easily and securely.

- [Amazon DocumentDB](#) – Amazon DocumentDB (with MongoDB compatibility) is a fast, reliable, and fully managed database service.
- [Amazon EC2](#) – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud.
- [Microsoft SQL Server](#) – SQL Server is a relational database management system.
- [SQL Server Management Studio (SSMS)](#) – SSMS is a tool for managing SQL Server, including accessing, configuring, and administering SQL Server components.

Epics

Create and configure a VPC

Task	Description	Skills required
Create a VPC.	Sign in to the AWS Management Console and open the Amazon VPC console. Create a virtual private cloud (VPC) with an IPv4 CIDR block range.	System administrator
Create security groups and network ACLs.	On the Amazon VPC console, create security groups and network access control lists (network ACLs) for your VPC, according to your requireme nts. You can also use the default settings for these configurations. For more information about this and other stories, see the "Related resources" section.	System administrator

Create and configure the Amazon DocumentDB cluster

Task	Description	Skills required
Create an Amazon DocumentDB cluster.	Open the Amazon DocumentDB console and choose "Clusters." Choose "Create," and create an Amazon DocumentDB cluster with one instance. Important : Make sure you configure this cluster with your VPC's security groups.	System administrator
Install the mongo shell.	The mongo shell is a command-line utility that you use to connect to and query your Amazon DocumentD B cluster. To install it, run the "/etc/yum.repos.d/ mongodb-org-3.6.repo" command to create the repository file. Run the "sudo yum install -y mongodb-o rg-shell" command to install the mongo shell. To encrypt data in transit, download the public key for Amazon DocumentDB, and then connect to your Amazon DocumentDB instance. For more information about these steps, see the "Related resources" section.	System administrator
Create a database in the Amazon DocumentDB cluster.	Run the "use" command with the name of your database	System administrator

Task	Description	Skills required
	to create a database in your Amazon DocumentDB cluster.	

Create and configure the AWS DMS replication instance

Task	Description	Skills required
Create the AWS DMS replicati on instance.	Open the AWS DMS console and choose "Create replicati on instance." Enter a name and description for your replication task. Choose the instance class, engine version, storage, VPC, Multi-AZ, and make it publicly accessibl e. Choose the "Advanced" tab to set the network and encryption settings. Specify the maintenance settings, and then choose "Create replicati on instance."	System administrator
Configure the SQL Server database.	Log in to Microsoft SQL Server and add an inbound rule for communication between the source endpoint and the AWS DMS replication instance. Use the replication instance's private IP address as the source. Important : The replication instance and target endpoint should be on the same VPC. Use an alternative source in the security group if the VPCs are	System administrator

Task	Description	Skills required
	different for the source and replication instances.	

Create and test the source and target endpoints in AWS DMS

Task	Description	Skills required
Create the source and target database endpoints.	Open the AWS DMS console and choose "Connect source and target database endpoints." Specify the connection information for the source and target databases. If required, choose the "Advanced" tab to set values for "Extra connection attributes." Download and use the certificate bundle in your endpoint configuration.	System administrator
Test the endpoint connection.	Choose "Run test" to test the connection. Troubleshoot any error messages by verifying the security group settings and the connections to the AWS DMS replication instance from both the source and target database instances.	System administrator

Migrate data

Task	Description	Skills required
Create the AWS DMS migration task.	On the AWS DMS console, choose "Tasks," "Create task." Specify the task options, including the source and destination endpoint names, and replication instance names. Under "Migratio n type" choose "Migrate existing data," and "Replicat e data changes only." Choose "Start task."	System administrator
Run the AWS DMS migration task.	Under "Task settings," specify the settings for the table preparation mode, such as "Do nothing," "Drop tables on target," "Truncate," and "Include LOB columns in replication." Set a maximum LOB size that AWS DMS will accept and choose "Enable logging." Leave the "Advanced settings" at their default values and choose "Create task."	System administrator
Monitor the migration.	On the AWS DMS console, choose "Tasks" and choose your migration task. Choose "Task monitoring" to monitor your task. The task stops when the full-load migration	System administrator

Task	Description	Skills required
	is complete and cached changes are applied.	

Test and verify the migration

Task	Description	Skills required
Connect to the Amazon DocumentDB cluster by using the mongo shell.	Open the Amazon DocumentDB console, choose your cluster under "Clusters." In the "Connecti vity and Security" tab, choose "Connect to this cluster with the mongo shell."	System administrator
Verify the results of your migration.	Run the "use" command with the name of your database and then run the "show collections" command. Run the "db. .count();" command with the name of your database. If the results match your source database, then your migration was successfu l.	System administrator

Related resources

Create and configure a VPC

- [Create a security group for your VPC]
- [Create a network ACL]

Create and configure the Amazon DocumentDB cluster

- Create an Amazon DocumentDB cluster
- Install the mongo shell for Amazon DocumentDB
- Connect to your Amazon DocumentDB cluster

Create and configure the AWS DMS replication instance

- Use public and private replication instances

Create and test the source and target endpoints in AWS DMS

- Use Amazon DocumentDB as a target for AWS DMS
- Use a SQL Server database as a source for AWS DMS
- Use AWS DMS endpoints

Migrate data

- Migrate to Amazon DocumentDB

Other resources

- Limitations on using SQL Server as a source for AWS DMS
- How to use Amazon DocumentDB to build and manage applications at scale

Migrate an on-premises ThoughtSpot Falcon database to Amazon Redshift

Battulga Purevragchaa and Antony Prasad Thevaraj, Amazon Web Services

Summary

On-premises data warehouses require significant administration time and resources, particularly for large datasets. The financial cost of building, maintaining, and growing these warehouses is also very high. To help manage costs, keep extract, transform, and load (ETL) complexity low, and deliver performance as your data grows, you must constantly choose which data to load and which data to archive.

By migrating your on-premises [ThoughtSpot Falcon databases](#) to the Amazon Web Services (AWS) Cloud, you can access cloud-based data lakes and data warehouses that increase your business agility, security, and application reliability, in addition to reducing your overall infrastructure costs. Amazon Redshift helps to significantly lower the cost and operational overhead of a data warehouse. You can also use Amazon Redshift Spectrum to analyze large amounts of data in its native format without data loading.

This pattern describes the steps and process for migrating a ThoughtSpot Falcon database from an on-premises data center to an Amazon Redshift database on the AWS Cloud.

Prerequisites and limitations

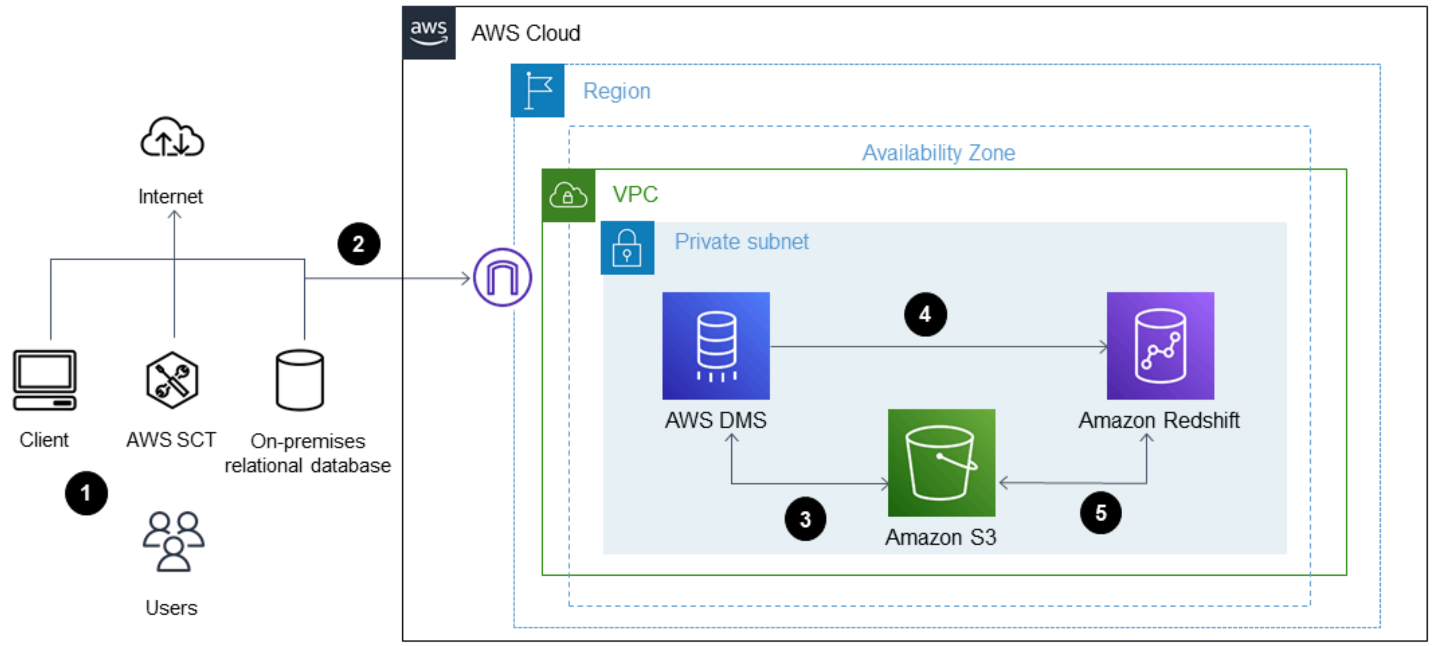
Prerequisites

- An active AWS account
- A ThoughtSpot Falcon database hosted in an on-premises data center

Product versions

- ThoughtSpot version 7.0.1

Architecture



The diagram shows the following workflow:

1. Data is hosted in an on-premises relational database.
2. AWS Schema Conversion Tool (AWS SCT) converts the data definition language (DDL) that is compatible with Amazon Redshift.
3. After the tables are created, you can migrate the data by using AWS Database Migration Service (AWS DMS).
4. The data is loaded into Amazon Redshift.
5. The data is stored in Amazon Simple Storage Service (Amazon S3) if you use Redshift Spectrum or already host the data in Amazon S3.

Tools

- AWS DMS – AWS Data Migration Service (AWS DMS) helps you quickly and securely migrate databases to AWS.
- Amazon Redshift – Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse service that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools.

- AWS SCT – AWS Schema Conversion Tool (AWS SCT) converts your existing database schema from one database engine to another.

Epics

Prepare for the migration

Task	Description	Skills required
Identify the appropriate Amazon Redshift configuration.	Identify the appropriate Amazon Redshift cluster configuration based on your requirements and data volume. For more information, see Amazon Redshift clusters in the Amazon Redshift documentation.	DBA
Research Amazon Redshift to evaluate if it meets your requirements.	Use the Amazon Redshift FAQs to understand and evaluate whether Amazon Redshift meets your requirements.	DBA

Prepare the target Amazon Redshift cluster

Task	Description	Skills required
Create an Amazon Redshift cluster.	Sign in to the AWS Management Console, open the Amazon Redshift console, and then create an Amazon Redshift cluster in a virtual private cloud (VPC).	DBA

Task	Description	Skills required
	For more information, see Creating a cluster in a VPC in the Amazon Redshift documentation.	
Conduct a PoC for your Amazon Redshift database design.	Follow Amazon Redshift best practices by conducting a proof of concept (PoC) for your database design. For more information, see Conducting a proof of concept for Amazon Redshift in the Amazon Redshift documentation.	DBA
Create database users.	Create the users in your Amazon Redshift database and grant the appropriate roles for access to the schema and tables. For more information, see Grant access privileges for a user or user group in the Amazon Redshift documentation.	DBA

Task	Description	Skills required
Apply configuration settings to the target database.	Apply configuration settings to the Amazon Redshift database according to your requirements. For more information about enabling database, session, and server-level parameter s, see the Configuration reference in the Amazon Redshift documentation.	DBA

Create objects in the Amazon Redshift cluster

Task	Description	Skills required
Manually create tables with DDL in Amazon Redshift.	(Optional) If you use AWS SCT, the tables are automatic ally created. However, if there are failures when replicating DDLs, you have to manually create the tables	DBA
Create external tables for Redshift Spectrum.	Create an external table with an external schema for Amazon Redshift Spectrum. To create external tables, you must be the owner of the external schema or a database superuser. For more information, see Creating external tables for Amazon Redshift Spectrum	DBA

Task	Description	Skills required
	in the Amazon Redshift documentation.	

Migrate data using AWS DMS

Task	Description	Skills required
Use AWS DMS to migrate the data.	After you create the DDL of the tables in the Amazon Redshift database, migrate your data to Amazon Redshift by using AWS DMS. For detailed steps and instructions, see [Using an Amazon Redshift database as a target for AWS DMS] in the AWS DMS documentation.	DBA
Use the COPY command to load the data.	Use the Amazon Redshift COPY command to load the data from Amazon S3 to Amazon Redshift. For more information, see [Using the COPY command to load from Amazon S3] in the Amazon Redshift documenta tion.	DBA

Validate the Amazon Redshift cluster

Task	Description	Skills required
Validate the source and target records.	Validate the table count for the source and target records that were loaded from your source system.	DBA
Implement Amazon Redshift best practices for performance tuning.	Implement Amazon Redshift best practices for table and database design. For more information, see the blog post [Top 10 performance tuning techniques for Amazon Redshift]().	DBA
Optimize query performance.	Amazon Redshift uses SQL-based queries to interact with data and objects in the system. Data manipulation language (DML) is the subset of SQL that you can use to view, add, change, and delete data. DDL is the subset of SQL that you use to add, change, and delete database objects such as tables and views. For more information, see [Tuning query performance]() in the Amazon Redshift documentation.	DBA
Implement WLM.	You can use workload management (WLM) to define multiple query queues and	DBA

Task	Description	Skills required
	route queries to appropriate queues at runtime. For more information, see Implementing workload management in the Amazon Redshift documentation.	
Work with concurrency scaling.	By using the Concurrency Scaling feature, you can support virtually unlimited concurrent users and concurrent queries, with consistently fast query performance. For more information, see Working with concurrency scaling in the Amazon Redshift documentation.	DBA
Use Amazon Redshift best practices for table design.	When you plan your database, certain important table design decisions can strongly influence overall query performance. For more information about choosing the most appropriate table design option, see Amazon Redshift best practices for designing tables in the Amazon Redshift documentation.	DBA

Task	Description	Skills required
Create materialized views in Amazon Redshift.	A materialized view contains a precomputed results set based on an SQL query over one or more base tables. You can issue `SELECT` statement s to query a materialized view in the same way that you query other tables or views in the database. For more information, see [Creating materialized views in Amazon Redshift]() in the Amazon Redshift documenta tion.	DBA
Define joins between the tables.	To search more than one table at the same time in ThoughtSpot, you must define joins between the tables by specifying columns that contain matching data across two tables. These columns represent the `primary key` and `foreign key` of the join. You can define them by using the `ALTER TABLE` command in Amazon Redshift or ThoughtSpot. For more information, see [ALTER TABLE]() in the Amazon Redshift documentation.	DBA

Set up ThoughtSpot connection to Amazon Redshift

Task	Description	Skills required
Add an Amazon Redshift connection.	Add an Amazon Redshift connection to your on-premises ThoughtSpot Falcon database. For more information, see [Add an Amazon Redshift connection] in the ThoughtSpot documentation.	DBA
Edit the Amazon Redshift connection.	You can edit the Amazon Redshift connection to add tables and columns. For more information, see [Edit an Amazon Redshift connection] in the ThoughtSpot documentation.	DBA
Remap the Amazon Redshift connection.	Modify the connection parameters by editing the source mapping .yaml file that was created when you added the Amazon Redshift connection. For example, you can remap the existing table or column to a different table or column in an existing database connection. ThoughtSpot recommends that you check the dependencies before and after you remap a table or	DBA

Task	Description	Skills required
	column in a connection to ensure that they display as required. For more information, see [Remap an Amazon Redshift connection] in the ThoughtSpot documentation.	
Delete a table from the Amazon Redshift connection.	(Optional) If you attempt to remove a table in an Amazon Redshift connection, ThoughtSpot checks for dependencies and shows a list of dependent objects. You can choose the listed objects to delete them or remove the dependency. You can then remove the table. For more information, see [Delete a table from an Amazon Redshift connection] in the ThoughtSpot documentation.	DBA

Task	Description	Skills required
Delete a table with dependent objects from an Amazon Redshift connection.	(Optional) If you try to delete a table with dependent objects, the operation is blocked. A `Cannot delete` window appears, with a list of links to dependent objects. When all the dependencies are removed, you can then delete the table For more information, see [Delete a table with dependent objects from an Amazon Redshift connection](#) in the ThoughtSpot documentation.	DBA
Delete an Amazon Redshift connection.	(Optional) Because a connection can be used in multiple data sources or visualizations, you must delete all of the sources and tasks that use that connection before you can delete the Amazon Redshift connection. For more information, see [Delete an Amazon Redshift connection](#) in the ThoughtSpot documentation.	DBA

Task	Description	Skills required
Check connection reference for Amazon Redshift.	Make sure that you provide the required information for your Amazon Redshift connection by using the Connection reference in the ThoughtSpot documentation.	DBA

Additional information

- AI-driven analytics at any scale with ThoughtSpot and Amazon Redshift
- Amazon Redshift pricing
- Getting started with AWS SCT
- Getting started with Amazon Redshift
- Using data extraction agents
- Chick-fil-A improves speed to insight with ThoughtSpot and AWS

Migrate an Oracle database to Amazon DynamoDB using AWS DMS

Rambabu Karnena, Amazon Web Services

Summary

This pattern walks you through the steps for migrating an Oracle database to Amazon DynamoDB using AWS Database Migration Service (AWS DMS). It covers three types of source databases:

- On-premises Oracle databases
- Oracle databases on Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Relational Database Service (Amazon RDS) for Oracle DB instances

In this proof of concept, this pattern focuses on migrating from an Amazon RDS for Oracle DB instance.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An application connecting to an Amazon RDS for Oracle database
- A table created in the source Amazon RDS for Oracle database with a primary key and sample data

Limitations

- Oracle database objects, such as procedures, functions, packages, and triggers, are not considered for migration because Amazon DynamoDB does not support these database objects.

Product versions

- This pattern applies to all editions and versions of Oracle databases that are supported by AWS DMS. For more information, see using an Oracle database as a source for AWS DMS and using an Amazon DynamoDB database as a target for AWS DMS. We recommend that you use the latest versions of AWS DMS for the most comprehensive version and feature support.

Architecture

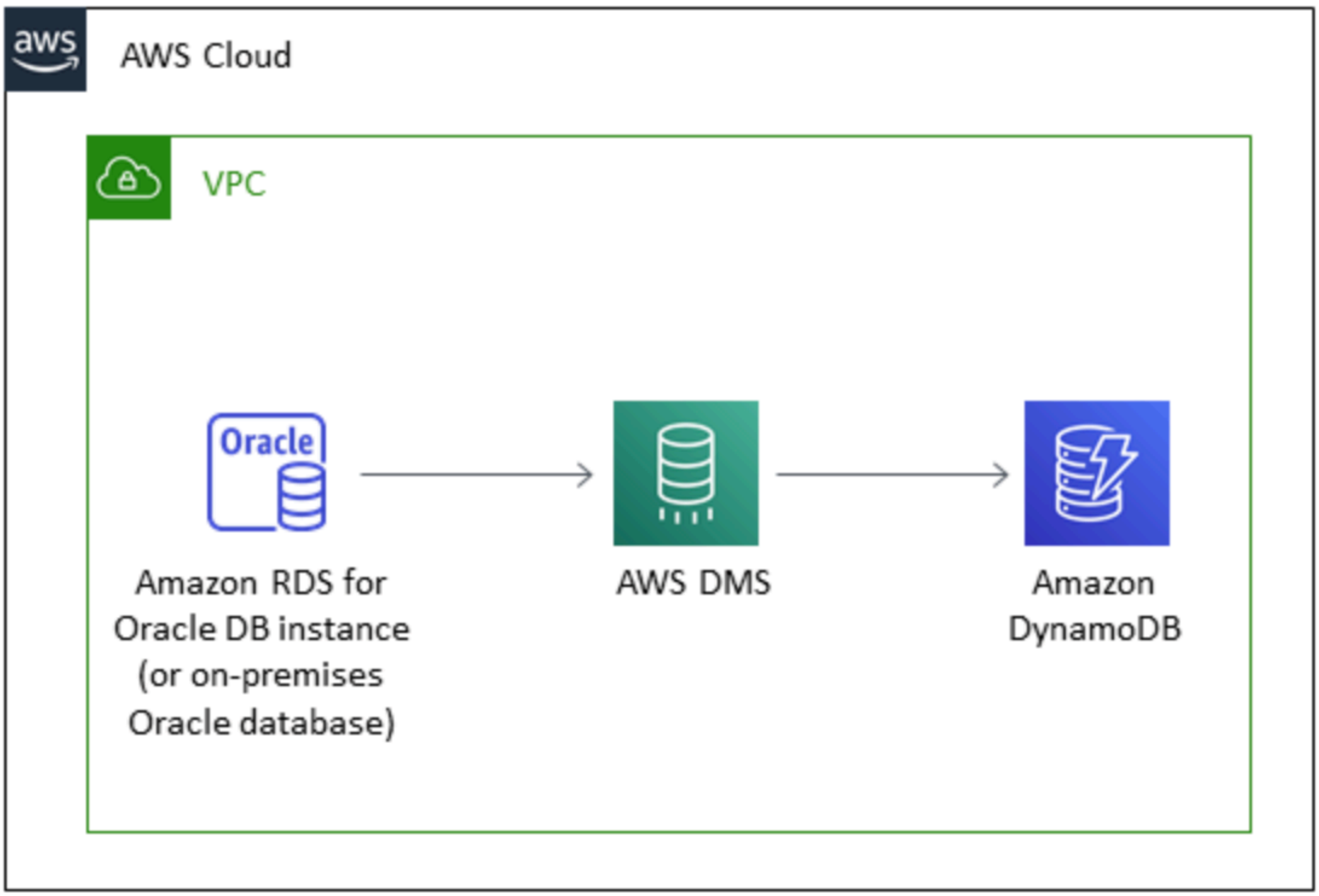
Source technology stack

- Amazon RDS for Oracle DB instances, Oracle on Amazon EC2, or on-premises Oracle databases

Target technology stack

- Amazon DynamoDB

AWS data migration architecture



Tools

- [AWS Database Migration Service (AWS DMS)](#) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.

- [Amazon DynamoDB](#) is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- [Amazon Relational Database Service (Amazon RDS)](#) helps you set up, operate, and scale a relational database in the AWS Cloud. This pattern uses Amazon RDS for Oracle.

Epics

Plan the migration

Task	Description	Skills required
Create a VPC.	In your AWS account, create a virtual private cloud (VPC) and a private subnet.	Systems administrator
Create security groups and network access control lists.	For more information, see the AWS documentation.	Systems administrator
Configure and start the Amazon RDS for Oracle DB instance.	For more information, see the AWS documentation.	DBA, Systems administrator

Migrate data

Task	Description	Skills required
Create an IAM role to access DynamoDB.	In the AWS Identity and Access Management (IAM) console, create the role, attach the policy `AmazonDynamoDBFullAccess to it`, and select AWS DMS as the service.	Systems administrator
Create an AWS DMS replication instance for migration.	The replication instance should be in the same	Systems administrator

Task	Description	Skills required
	Availability Zone and VPC as the source database.	
Create source and target endpoints in AWS DMS.	To create the source database endpoint, you have two options: • On the Amazon RDS console, choose **Databases**, **DB identifier**, **Connectivity & Security**, and choose the endpoint. • On the AWS DMS console, choose **Select RDS DB instance**. To create the target database endpoint, choose the role Amazon Resource Name (ARN) from the previous task to access DynamoDB.	Systems administrator
Create an AWS DMS task to load the source Oracle database tables to DynamoDB.	Choose the source and destination endpoint names and the replication instance from the previous steps. The type can be full load. Choose the Oracle schema and specify **%** to select all tables.	Systems administrator
Validate the tables in DynamoDB.	To view the migration results, choose **Tables** from the left navigation pane in the DynamoDB console.	DBA

Migrate the application

Task	Description	Skills required
Modify the application code.	To connect to and retrieve data from DynamoDB, update the application code.	App owner, DBA, Systems administrator

Cut over

Task	Description	Skills required
Switch the application clients to use DynamoDB.		DBA, App owner, Systems administrator

Close the project

Task	Description	Skills required
Shut down AWS resources.	For example, the shut down the Amazon RDS for Oracle instance, DynamoDB, and the AWS DMS replication instance.	DBA, Systems administrator
Gather metrics.	Metrics include the time to migrate, the percentages of manual work and work performed by the tool, and cost savings.	DBA, App owner, Systems administrator

Related resources

- AWS Database Migration Service and Amazon DynamoDB: What You Need to Know (blog post)
- Using an Oracle Database as a source for AWS DMS

- Using an Amazon DynamoDB database as a target for AWS Database Migration Service
- Best Practices for Migrating from RDBMS to Amazon DynamoDB (whitepaper)

Migrate an Oracle partitioned table to PostgreSQL by using AWS DMS

Saurav Mishra and Eduardo Valentim, Amazon Web Services

Summary

This pattern describes how to speed up loading a partitioned table from Oracle to PostgreSQL by using AWS Database Migration Service (AWS DMS), which doesn't support native partitioning. The target PostgreSQL database can be installed on Amazon Elastic Compute Cloud (Amazon EC2), or it can be an Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition DB instance.

Uploading a partitioned table includes the following steps:

1. Create a parent table similar to the Oracle partition table, but don't include any partition.
2. Create child tables that will inherit from the parent table that you created in step 1.
3. Create a procedure function and trigger to handle the inserts in the parent table.

However, because the trigger is fired for every insert, the initial load using AWS DMS can be very slow.

To speed up initial loads from Oracle to PostgreSQL 9.0, this pattern creates a separate AWS DMS task for each partition and loads the corresponding child tables. You then create a trigger during cutover.

PostgreSQL version 10 supports native partitioning. However, you might decide to use inherited partitioning in some cases. For more information, see the [Additional information](#) section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle database with a partitioned table
- A PostgreSQL database on AWS

Product versions

- PostgreSQL 9.0

Architecture

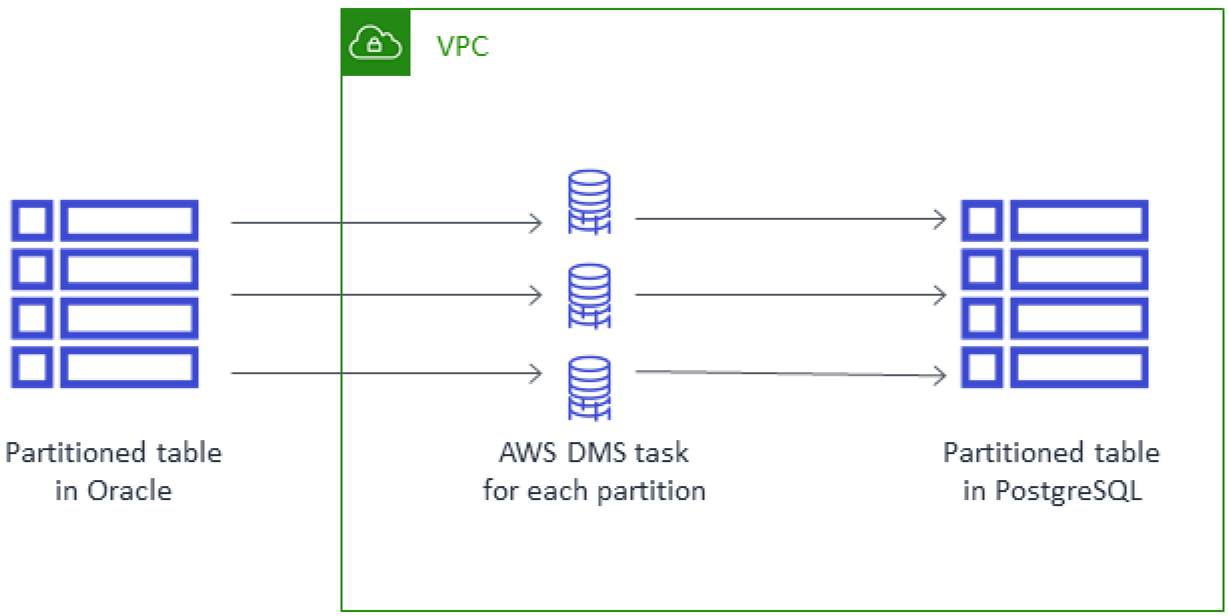
Source technology stack

- A partitioned table in Oracle

Target technology stack

- A partitioned table in PostgreSQL (on Amazon EC2, Amazon RDS for PostgreSQL, or Aurora PostgreSQL)

Target architecture



Tools

- [AWS Database Migration Service (AWS DMS)] helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.

Epics

Set up AWS DMS

Task	Description	Skills required
Create the tables in PostgreSQL.	Create the parent and corresponding child tables in PostgreSQL with the required check conditions for partition s.	DBA
Create the AWS DMS task for each partition.	Include the filter condition of the partition in the AWS DMS task. Map the partitions to the corresponding PostgreSQ L child tables.	DBA
Run the AWS DMS tasks using full load and change data capture (CDC).	Make sure that the `StopTaskCachedChan gesApplied` parameter is set to `true` and the `StopTaskCachedChan gesNotApplied` parameter is set to `false`.	DBA

Cut over

Task	Description	Skills required
Stop the replication tasks.	Before you stop the tasks, confirm that the source and destination are in sync.	DBA
Create a trigger on the parent table.	Because the parent table will receive all insert and update commands, create a	DBA

Task	Description	Skills required
	trigger that will route these commands to the respectiv e child tables based on the partitioning condition.	

Related resources

- AWS DMS
- Table Partitioning (PostgreSQL documentation)

Additional information

Although PostgreSQL version 10 supports native partitioning, you might decide to use inherited partitioning for the following use cases:

- Partitioning enforces a rule that all partitions must have the same set of columns as the parent, but table inheritance supports children having extra columns.
- Table inheritance supports multiple inheritances.
- Declarative partitioning supports only list and range partitioning. With table inheritance, you can divide the data as you want. However, if the constraint exclusion can't prune partitions effectively, query performance will suffer.
- Some operations need a stronger lock when using declarative partitioning than when using table inheritance. For example, adding or removing a partition to or from a partitioned table requires an `ACCESS EXCLUSIVE` lock on the parent table, whereas a `SHARE UPDATE EXCLUSIVE` lock is enough for regular inheritance.

When you use separate job partitions, you can also reload partitions if there are any AWS DMS validation issues. For better performance and replication control, run tasks on separate replication instances.

Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL

Jitender Kumar, Srini Ramaswamy, and Neha Sharma, Amazon Web Services

Summary

This pattern provides guidance for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance to an Amazon RDS for MySQL DB instance on Amazon Web Services (AWS). The pattern uses AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT).

The pattern provides best practices for handling the migration of stored procedures. It also covers and code changes to support the application layer.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An Amazon RDS for Oracle source database.
- An Amazon RDS for MySQL target database. Source and target databases should be in the same virtual private cloud (VPC). If you're using multiple VPCs, or you must have the required access permissions.
- Security groups that allow connectivity between the source and target databases, AWS SCT, the application server, and AWS DMS.
- A user account with the required privilege to run AWS SCT on the source database.
- Supplemental logging enabled for running AWS DMS on the source database.

Limitations

- The source and target Amazon RDS database size limit is 64 TB. For Amazon RDS size information, see the AWS documentation.
- Oracle is case-insensitive for database objects, but MySQL is not. AWS SCT can handle this issue while creating an object. However, some manual work is required to support full case insensitivity.
- This migration doesn't use MySQL extensions to enable Oracle-native functions. AWS SCT handles most of the conversion, but some work is required to change code manually.

- Java Database Connectivity (JDBC) driver changes are required in the application.

Product versions

- Amazon RDS for Oracle 12.2.0.1 and later. For currently supported RDS for Oracle versions, see the AWS documentation.
- Amazon RDS for MySQL 8.0.15 and later. For currently supported RDS for MySQL versions, see the AWS documentation.
- AWS DMS version 3.3.0 and later. See the AWS documentation for more information about AWS DMS supported source endpoints and target endpoints.
- AWS SCT version 1.0.628 and later. See the AWS SCT source and target endpoint support matrix in the AWS documentation.

Architecture

Source technology stack

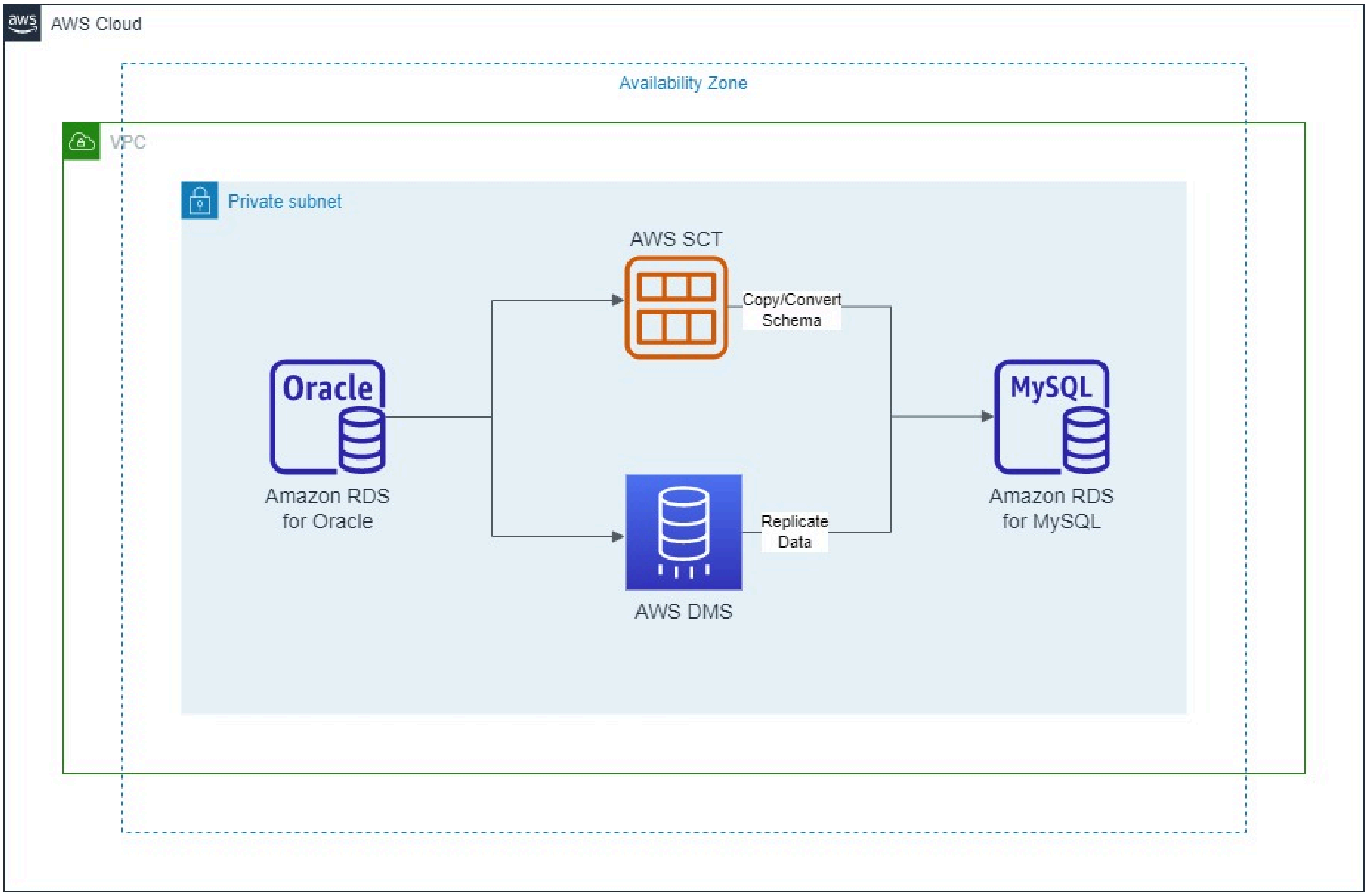
- Amazon RDS for Oracle. For more information, see Using an Oracle database as a source for AWS DMS.

Target technology stack

- Amazon RDS for MySQL. For more information, see Using a MySQL-Compatible database as a target for AWS DMS.

Migration architecture

In the following diagram, AWS SCT copies and converts schema objects from the Amazon RDS for Oracle source database and sends the objects to the Amazon RDS for MySQL target database. AWS DMS replicates data from the source database and sends it to the Amazon RDS for MySQL instance.



Tools

- AWS Data Migration Service helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud. This pattern uses Amazon RDS for Oracle and Amazon RDS for MySQL.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.

Epics

Prepare for migration

Task	Description	Skills required
Validate the source and target database versions and engines.		DBA
Identify hardware requirements for the target server instance.		DBA, SysAdmin
Identify storage requirements (storage type and capacity).		DBA, SysAdmin
Choose the proper instance type (capacity, storage features, network features).		DBA, SysAdmin
Identify network-access security requirements for the source and target databases.		DBA, SysAdmin
Choose an application migration strategy.	Consider whether you want full downtime or partial downtime for cutover activities.	DBA, SysAdmin, App owner

Configure infrastructure

Task	Description	Skills required
Create a VPC and subnets.		SysAdmin

Task	Description	Skills required
Create security groups and network access control lists (ACLs).		SysAdmin
Configure and start the Amazon RDS for Oracle instance.		DBA, SysAdmin
Configure and start the Amazon RDS for MySQL instance.		DBA, SysAdmin
Prepare a test case for validation of code conversion.	This will help in unit-testing for the converted code.	DBA, Developer
Configure the AWS DMS instance.		
Configure source and target endpoints in AWS DMS.		

Migrate data

Task	Description	Skills required
Generate the target database script using AWS SCT.	Check the accuracy of the code that was converted by AWS SCT. Some manual work will be required.	DBA, Developer
In AWS SCT, choose the "Case Insensitive" setting.	In AWS SCT, choose Project Settings, Target Case Sensitivi ty, Case Insensitive.	DBA, Developer

Task	Description	Skills required
In AWS SCT, choose not to use the Oracle native function.	In Project Settings, check the functions TO_CHAR/TO_NUMBER/TO_DATE.	DBA, Developer
Make changes for "sql%notfound" code.	You might have to convert the code manually.	
Query on tables and objects in stored procedures (use lowercase queries).		DBA, Developer
Create the primary script after all changes are made, and then deploy the primary script on the target database.		DBA, Developer
Unit-test stored procedures and application calls using sample data.		
Clean up data that was created during unit testing.		DBA, Developer
Drop foreign key constraints on the target database.	This step is required to load initial data. If you don't want to drop the foreign key constraints, you must create a migration task for data specific to the primary and secondary tables.	DBA, Developer
Drop primary keys and unique keys on the target database.	This step results in better performance for the initial load.	DBA, Developer
Enable supplemental logging on the source database.		DBA

Task	Description	Skills required
Create a migration task for the initial load in AWS DMS, and then run it.	Choose the option to migrate existing data.	DBA
Add the primary keys and foreign keys to the target database.	Constraints need to be added after the initial load.	DBA, Developer
Create a migration task for ongoing replication.	Ongoing replication keeps the target database synchronized with the source database.	DBA

Migrate applications

Task	Description	Skills required
Replace Oracle native functions with MySQL native functions.		App owner
Make sure that only lowercase names are used for database objects in SQL queries.		DBA, SysAdmin, App owner

Cut over to the target database

Task	Description	Skills required
Shut down the application server.		App owner
Validate that the source and target databases are in sync.		DBA, App owner

Task	Description	Skills required
Stop the Amazon RDS for Oracle DB instance.		DBA
Stop the migration task.	This will stop automatic ally after you complete the previous step.	DBA
Change the JDBC connection from Oracle to MySQL.		App owner, DBA
Start the application.		DBA, SysAdmin, App owner

Close the project

Task	Description	Skills required
Review and validate the project documents.		DBA, SysAdmin
Gather metrics about time to migrate, percentage of manual versus tool tasks, cost savings, etc.		DBA, SysAdmin
Stop and delete AWS DMS instances.		DBA
Remove the source and target endpoints.		DBA
Remove migration tasks.		DBA
Take a snapshot of the Amazon RDS for Oracle DB instance.		DBA

Task	Description	Skills required
Delete the Amazon RDS for Oracle DB instance.		DBA
Shut down and delete any other temporary AWS resources you used.		DBA, SysAdmin
Close the project and provide any feedback.		DBA

Related resources

- AWS DMS
- AWS SCT
- Amazon RDS Pricing
- Getting Started with AWS DMS
- Getting Started with Amazon RDS

Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL-Compatible using AWS DMS and AWS SCT

Sirsendu Halder and Abhimanyu Chhabra, Amazon Web Services

Summary

This pattern provides guidance for migrating an IBM Db2 database on an Amazon Elastic Compute Cloud (Amazon EC2) instance to an Amazon Aurora PostgreSQL-Compatible Edition DB instance. This pattern uses AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT) for data migration and schema conversion.

The pattern targets an online migration strategy with little or no downtime for a multi-terabyte IBM Db2 database that has a high number of transactions. We recommend that you convert the columns in primary keys (PKs) and foreign keys (FKs) with the data type `NUMERIC` to `INT` or `BIGINT` in PostgreSQL for better performance.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source IBM Db2 database on an EC2 instance

Product versions

- DB2/LINUXX8664 version 11.1.4.4 and later

Architecture

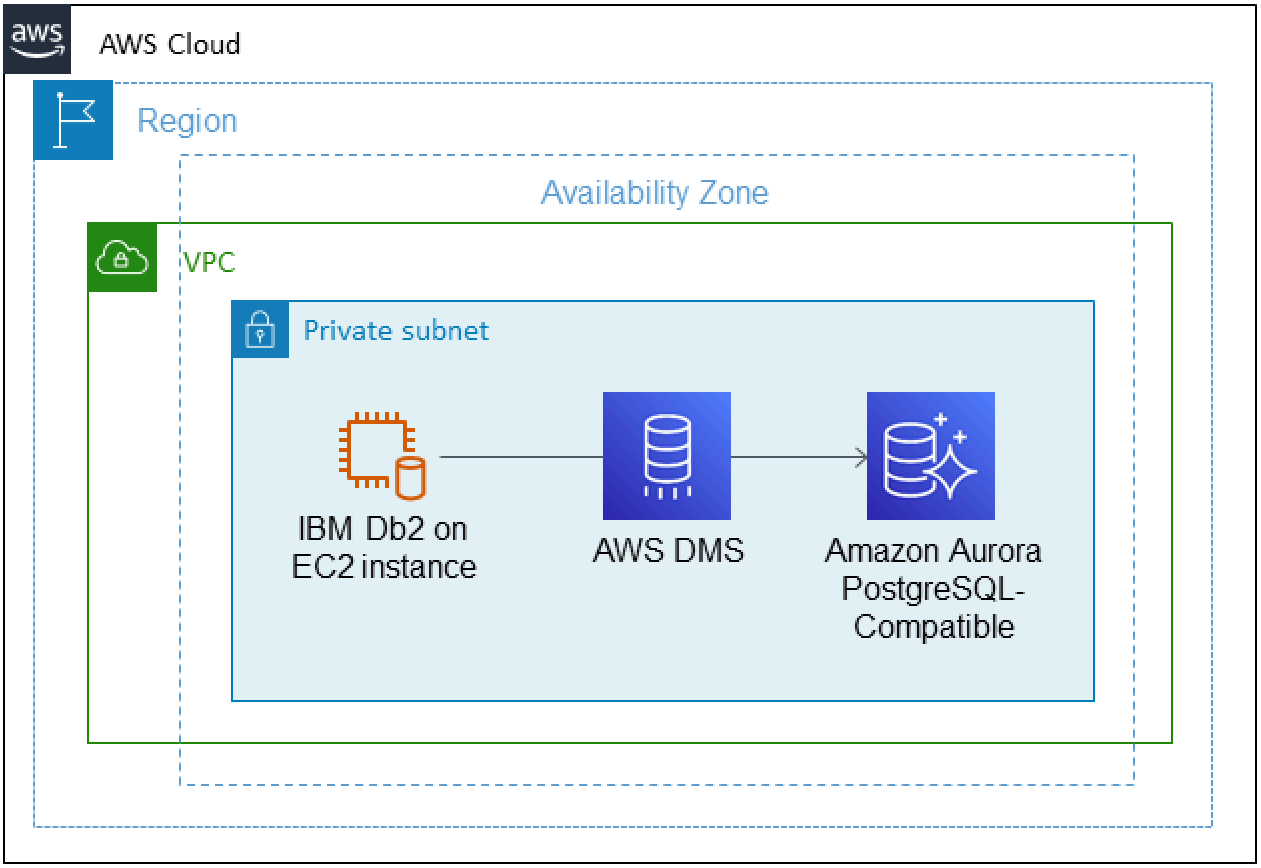
Source technology stack

- A Db2 database on an EC2 instance

Target technology stack

- An Aurora PostgreSQL-Compatible version 10.18 or later DB instance

Database migration architecture



Tools

- [AWS Database Migration Service (AWS DMS)](https://docs.aws.amazon.com/dms/latest/userguide/Welcome.html) helps you migrate databases into the AWS Cloud or between combinations of cloud and on-premises setups. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. You can use AWS DMS to migrate your data to and from the most widely used commercial and open-source databases. AWS DMS supports heterogeneous migrations between different database platforms, such as IBM Db2 to Aurora PostgreSQL-Compatible version 10.18 or higher. For details, see Sources for Data Migration and Targets for Data Migration in the AWS DMS documentation.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format that's compatible with the target database. Any objects that are not automatically converted are clearly marked so that they can be manually converted to complete the migration. AWS SCT can also scan the application source code for embedded SQL statements and convert them.

Epics

Set up the environment

Task	Description	Skills required
Create an Aurora PostgreSQL-Compatible DB instance.	To create the DB instance, follow the instructions in the [AWS documentation](). For engine type, choose **Amazon Aurora**. For edition, choose **Amazon Aurora PostgreSQL-Compatible Edition**. The Aurora PostgreSQL-Compatible version 10.18 or later DB instance should be in the same virtual private cloud (VPC) as the source IBM Db2 database.	Amazon RDS

Convert your database schema

Task	Description	Skills required
Install and verify AWS SCT.	1. Install AWS SCT by following the steps in the [AWS SCT documentation](). 2. Verify the installation by following the procedures in the [AWS SCT documentation]().	AWS administrator, DBA, Migration engineer
Start AWS SCT and create a project.	To start the AWS SCT tool and create a new project to run a database migration assessment report, follow the	Migration engineer

Task	Description	Skills required
	instructions in the [AWS SCT documentation].	
Add database servers and create a mapping rule.	1. Add source and target database servers by following the instructions in the [AWS SCT documenta tion]. 2. Create a mapping rule to define the target database platform for your source database. For instructi ons, see the [AWS SCT documentation].	Migration engineer
Create a database migration assessment report.	Create the database migration assessment report by following the steps in the [AWS SCT documentation].	Migration engineer
View the assessment report.	Use the **Summary** tab of the database migration assessment report to view the report and analyze the data. This analysis will help you determine the complexit y of the migration. For more information, see the [AWS SCT documentation].	Migration engineer

Task	Description	Skills required
Convert the schema.	To convert your source database schemas: 1. On the AWS SCT console, choose **View**, and then **Main view**. 2. Select the object or parent node from your source schema, open the context (right-click) menu, and then choose **Convert schema**. For more information, see the [AWS SCT documentation]().	Migration engineer
Apply the converted database schema to the target DB instance.	1. Choose the schema element in the right panel of your project that displays the planned schema for your target DB instance. 2. Open the context (right-cl ick) menu for the schema element, and then choose **Apply to database**. For more information, see the [AWS SCT documentation]().	Migration engineer

Migrate your data

Task	Description	Skills required
Set up a VPC and DB parameter groups.	Set up a VPC and DB parameter groups, and configure the inbound rules and parameters required for migration. For instructions, see the [AWS DMS documentation](). For the VPC security group, select both the EC2 instance for Db2 and the Aurora PostgreSQL-Compatible DB instance. This replication instance must be in the same VPC as the source and target DB instances.	Migration engineer
Prepare source and target DB instances.	Prepare the source and target DB instances for migration . In a production environme nt, the source database will already exist. For the source database, the server name must be the public Domain Name System (DNS) of the EC2 instance where Db2 is running. For user name, you can use `db2inst1` followed by the port, which will be 5000 for IBM Db2.	Migration engineer

Task	Description	Skills required
Create an Amazon EC2 client and endpoints.	1. Create an Amazon EC2 client. You use this client to populate your source database with data to replicate. You also use this client to verify replication by running queries on the target database. 2. Create endpoints for the source database and target DB instance to use for the next steps. For instructi ons, see the [AWS DMS documentation](). You must create separate endpoints for the source and target databases. For Aurora PostgreSQL-Compatible version 10.18 or later, the port will be 5432, and you can get the server name from the DB instance's endpoint.	Migration engineer
Create a replication instance.	Create a replication instance by using the AWS DMS console and specify the source and target endpoints . The replication instance performs the data migration between the endpoints. For more information, see the [AWS DMS documentation]().	Migration engineer

Task	Description	Skills required
Create an AWS DMS task to migrate the data.	Create a task to load the source IBM Db2 tables to the target PostgreSQL DB instance by following the steps in the [AWS DMS documentation](). • For source and target, use the source and destination endpoint names. • The migration type can be full load. • For the schema rule, you can use the `inst1` schema from the Db2 database. • For the table name, specify `%` to migrate all tables. When the load is complete, you'll see the Db2 tables of the `inst1` schema appearing in the Aurora PostgreSQL-Compatible database.	Migration engineer

Related resources

References

- Amazon Aurora documentation
- PostgreSQL foreign data wrapper (FDW) documentation
- PostgreSQL IMPORT FOREIGN SCHEMA documentation
- AWS DMS documentation
- AWS SCT documentation

Tutorials and videos

- Getting Started with AWS DMS (walkthrough)
- Introduction to Amazon EC2 - Elastic Cloud Server & Hosting with AWS (video)

Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS

Kumar Babu P G, Amazon Web Services

Summary

This pattern describes how to migrate an on-premises Oracle 8i or 9i database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora PostgreSQL. AWS Database Migration Service (AWS DMS) doesn't support Oracle 8i or 9i as a source, so Quest SharePlex replicates data from an on-premises 8i or 9i database to an intermediate Oracle database (Oracle 10g or 11g), which is compatible with AWS DMS.

From the intermediate Oracle instance, the schema and data are migrated to the PostgreSQL database on AWS by using AWS Schema Conversion Tool (AWS SCT) and AWS DMS. This method helps achieve continuous streaming of data from the source Oracle database to the target PostgreSQL DB instance with minimum replication lag. In this implementation, the downtime is limited to the length of time it takes to create or validate all the foreign keys, triggers, and sequences on the target PostgreSQL database.

The migration uses an Amazon Elastic Compute Cloud (Amazon EC2) instance with Oracle 10g or 11g installed to host the changes from the source Oracle database. AWS DMS uses this intermediate Oracle instance as the source to stream the data to Amazon RDS for PostgreSQL or Aurora PostgreSQL. Data replication can be paused and resumed from the on-premises Oracle database to the intermediate Oracle instance. It can also be paused and resumed from the intermediate Oracle instance to the target PostgreSQL database so you can validate the data by using either AWS DMS data validation or a custom data validation tool.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle 8i or 9i database in an on-premises data center
- AWS Direct Connect configured between the on-premises data center and AWS
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors installed either on a local machine or on the EC2 instance where AWS SCT is installed
- Familiarity with using an Oracle database as an AWS DMS source

- Familiarity with using a PostgreSQL database as an AWS DMS target
- Familiarity with Quest SharePlex data replication

Limitations

- The database size limit is 64 TB
- The on-premises Oracle database must be Enterprise Edition

Product versions

- Oracle 8i or 9i for the source database
- Oracle 10g or 11g for the intermediate database
- PostgreSQL 9.6 or later

Architecture

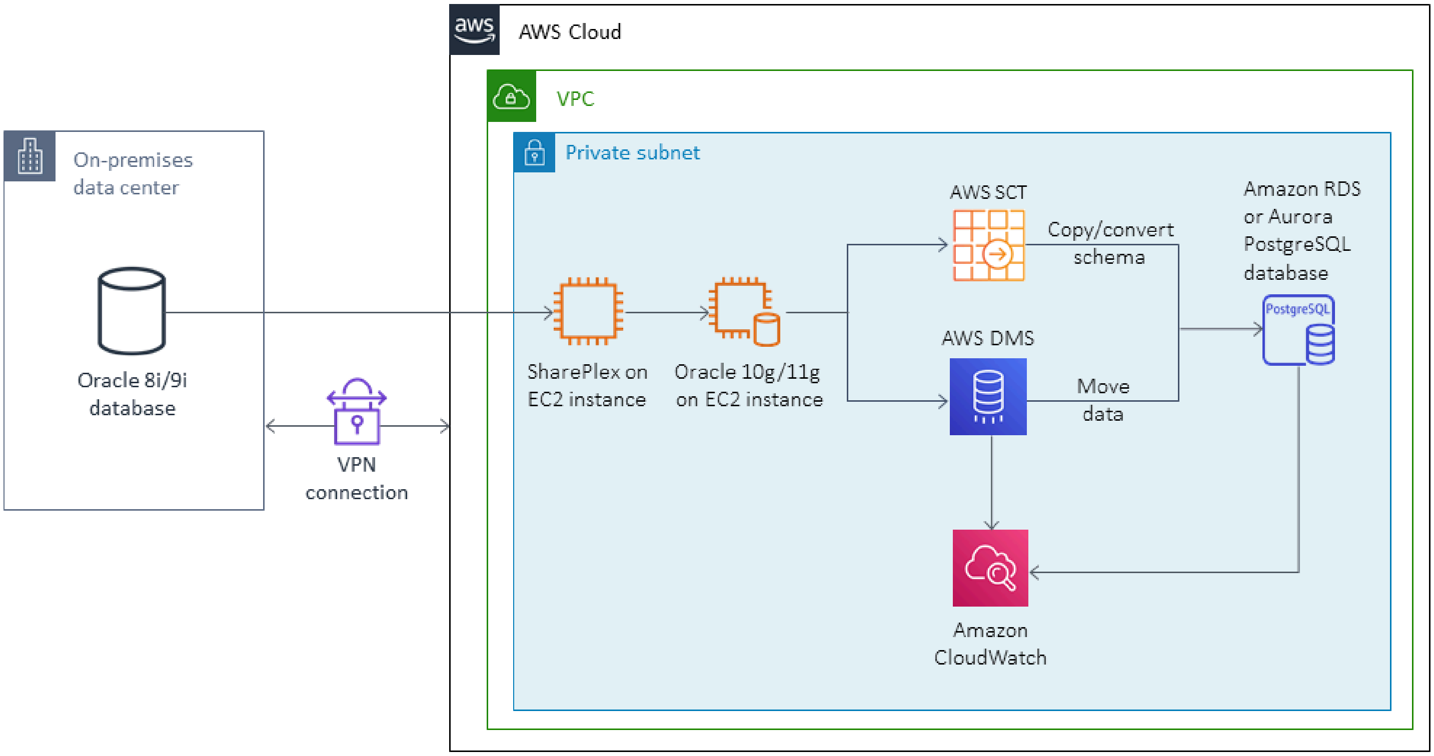
Source technology stack

- Oracle 8i or 9i database
- Quest SharePlex

Target technology stack

- Amazon RDS for PostgreSQL or Aurora PostgreSQL

Source and target architecture



Tools

- **AWS DMS** – AWS Database Migration Service (AWS DMS) helps you migrate databases quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.
- **AWS SCT** – AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database. Objects that cannot be automatically converted are clearly marked so that they can be manually converted to complete the migration. AWS SCT can also scan your application source code for embedded SQL statements and convert them as part of a database schema conversion project. During this process, AWS SCT performs cloud-native code optimization by converting legacy Oracle and SQL Server functions to their AWS equivalents, to help you modernize your applications while migrating your databases. When schema conversion is complete, AWS SCT can help migrate data from a range of data warehouses to Amazon Redshift by using built-in data migration agents.
- **Quest SharePlex** – Quest SharePlex is an Oracle-to-Oracle data replication tool for moving data with minimal downtime and no data loss.

Epics

Create the EC2 instance and install Oracle

Task	Description	Skills required
Set up the network for Amazon EC2.	Create the virtual private cloud (VPC), subnets, internet gateway, route tables, and security groups.	AWS SysAdmin
Create the new EC2 instance.	Select the Amazon Machine Image (AMI) for the EC2 instance. Choose the instance size and configure instance details: the number of instances (1), the VPC and subnet from the previous step, auto-assign public IP, and other options. Add storage, configure security groups, and launch the instance. When prompted, create and save a key pair for the next step.	AWS SysAdmin
Install Oracle on the EC2 instance.	Acquire the licenses and the required Oracle binaries, and install Oracle 10g or 11g on the EC2 instance.	DBA

Set up SharePlex on an EC2 instance and configure data replication

Task	Description	Skills required
Set up SharePlex.	Create an Amazon EC2 instance and install the	AWS SysAdmin, DBA

Task	Description	Skills required
	SharePlex binaries that are compatible with Oracle 8i or 9i.	
Configure data replication.	Follow SharePlex best practices to configure data replication from an on-premises Oracle 8i/9i database to an Oracle 10g/11g instance.	DBA

Convert the Oracle database schema to PostgreSQL

Task	Description	Skills required
Set up AWS SCT.	Create a new report, and then connect to Oracle as the source and PostgreSQL as the target. In project settings, open the SQL Scripting tab and change the target SQL script to Multiple Files.	DBA
Convert the Oracle database schema.	In the Action tab, choose Generate Report, Convert Schema, and then Save as SQL.	DBA
Modify the SQL scripts generated by AWS SCT.		DBA

Create and configure the Amazon RDS DB instance

Task	Description	Skills required
Create the Amazon RDS DB instance.	In the Amazon RDS console, create a new PostgreSQL DB instance.	AWS SysAdmin, DBA
Configure the DB instance.	Specify the DB engine version, DB instance class, Multi-AZ deployment, storage type, and allocated storage. Enter the DB instance identifier, a master user name, and a master password.	AWS SysAdmin, DBA
Configure network and security.	Specify the VPC, subnet group, public accessibility, Availability Zone preference, and security groups.	AWS SysAdmin, DBA
Configure database options.	Specify the database name, port, parameter group, encryption, and master key.	AWS SysAdmin, DBA
Configure backups.	Specify the backup retention period, backup window, start time, duration, and whether to copy tags to snapshots.	AWS SysAdmin, DBA
Configure monitoring options.	Enable or disable enhanced monitoring and performance insights.	AWS SysAdmin, DBA
Configure maintenance options.	Specify auto minor version upgrade, maintenance	AWS SysAdmin, DBA

Task	Description	Skills required
	window, and the start day, time, and duration.	
Run the pre-migration scripts from AWS SCT.	On the Amazon RDS instance, run these scripts: create_da tabase.sql, create_se quence.sql, create_table.sql, create_view.sql, and create_fu nction.sql.	AWS SysAdmin, DBA

Migrate data by using AWS DMS

Task	Description	Skills required
Create a replication instance in AWS DMS.	Complete the fields for the name, instance class, VPC (same as for the EC2 instance) , Multi-AZ, and public accessibility. In the advanced configuration section, specify allocated storage, subnet group, Availability Zone, VPC security groups, and AWS Key Management Service (AWS KMS) root key.	AWS SysAdmin, DBA
Create the source database endpoint.	Specify the endpoint name, type, source engine (Oracle), server name (Amazon EC2 private DNS name), port, SSL mode, user name, password, SID, VPC (specify the VPC that has the replication instance) , and replication instance. To test the connection, choose	AWS SysAdmin, DBA

Task	Description	Skills required
	Run Test, and then create the endpoint. You can also configure the following advanced settings: maxFileSize and numberDataTypeScale.	
Create the AWS DMS replication task.	Specify the task name, replication instance, source and target endpoints, and replication instance. For migration type, choose "Migrate existing data and replicate ongoing changes." Clear the "Start task on create" check box.	AWS SysAdmin, DBA
Configure the AWS DMS replication task settings.	For target table preparation mode, choose "Do nothing." Stop the task after the full load completes to create primary keys. Specify limited or full LOB mode, and enable control tables. Optionally, you can configure the CommitRate advanced setting.	DBA

Task	Description	Skills required
Configure the table mappings.	In the table mappings section, create an Include rule for all tables in all schemas included in the migration, and then create an Exclude rule. Add three transformation rules to convert the schema, table, and column names to lowercase, and add any other rules needed for this specific migration.	DBA
Start the task.	Start the replication task. Make sure that the full load is running. Run ALTER SYSTEM SWITCH LOGFILE on the primary Oracle database to kick-start the task.	DBA
Run the mid-migration scripts from AWS SCT.	In Amazon RDS for PostgreSQL, run these scripts: create_index.sql and create_constraint.sql.	DBA
Restart the task to continue change data capture (CDC).	In the Amazon RDS for PostgreSQL DB instance, run VACUUM, and restart the AWS DMS task to apply the cached CDC changes.	DBA

Cut over to the PostgreSQL database

Task	Description	Skills required
Check the AWS DMS logs and metadata tables.	Validate any errors and fix if required.	DBA
Stop all Oracle dependencies.	Shut down listeners on the Oracle database and run ALTER SYSTEM SWITCH LOGFILE. Stop the AWS DMS task when it shows no activity.	DBA
Run the post-migration scripts from AWS SCT.	In Amazon RDS for PostgreSQL, run these scripts: create_foreign_key_constraint.sql and create_triggers.sql.	DBA
Complete any additional Amazon RDS for PostgreSQL steps.	Increment sequences to match Oracle if needed, run VACUUM and ANALYZE, and take a snapshot for compliance.	DBA
Open the connections to Amazon RDS for PostgreSQL.	Remove the AWS DMS security groups from Amazon RDS for PostgreSQL, add production security groups, and point your applications to the new database.	DBA
Clean up AWS DMS resources.	Remove the endpoints, replication tasks, replication instances, and the EC2 instance.	SysAdmin, DBA

Related resources

- AWS DMS documentation
- AWS SCT documentation
- Amazon RDS for PostgreSQL pricing
- Using an Oracle database as a source for AWS DMS
- Using a PostgreSQL database as a target for AWS DMS
- Quest SharePlex documentation

Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS

Kumar Babu P G and Pragnesh Patel, Amazon Web Services

Summary

This pattern describes how to migrate an on-premises legacy Oracle 8i or 9i database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition.

AWS Database Migration Service (AWS DMS) doesn't support Oracle 8i or 9i as a source, so this pattern uses an intermediate Oracle database instance that's compatible with AWS DMS, such as Oracle 10g or 11g. It also uses the materialized views feature to migrate data from the source Oracle 8i/9i instance to the intermediate Oracle 10g/11g instance.

AWS Schema Conversion Tool (AWS SCT) converts the database schema, and AWS DMS migrates the data to the target PostgreSQL database.

This pattern helps users who want to migrate from legacy Oracle databases with minimum database downtime. In this implementation, the downtime would be limited to the length of time it takes to create or validate all the foreign keys, triggers, and sequences on the target database.

The pattern uses Amazon Elastic Compute Cloud (Amazon EC2) instances with an Oracle 10g/11g database installed to help AWS DMS stream the data. You can temporarily pause streaming replication from the on-premises Oracle database to the intermediate Oracle instance to enable AWS DMS to catch up on data validation or to use another data validation tool. The PostgreSQL DB instance and intermediate Oracle database will have the same data when AWS DMS has finished migrating current changes.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle 8i or 9i database in an on-premises data center
- AWS Direct Connect configured between the on-premises data center and AWS
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors installed either on a local machine or on the EC2 instance where AWS SCT is installed

- Familiarity with [using an Oracle database as an AWS DMS source]
- Familiarity with [using a PostgreSQL database as an AWS DMS target]

Limitations

- The database size limit is 64 TB

Product versions

- Oracle 8i or 9i for the source database
- Oracle 10g or 11g for the intermediate database
- PostgreSQL 10.17 or later

Architecture

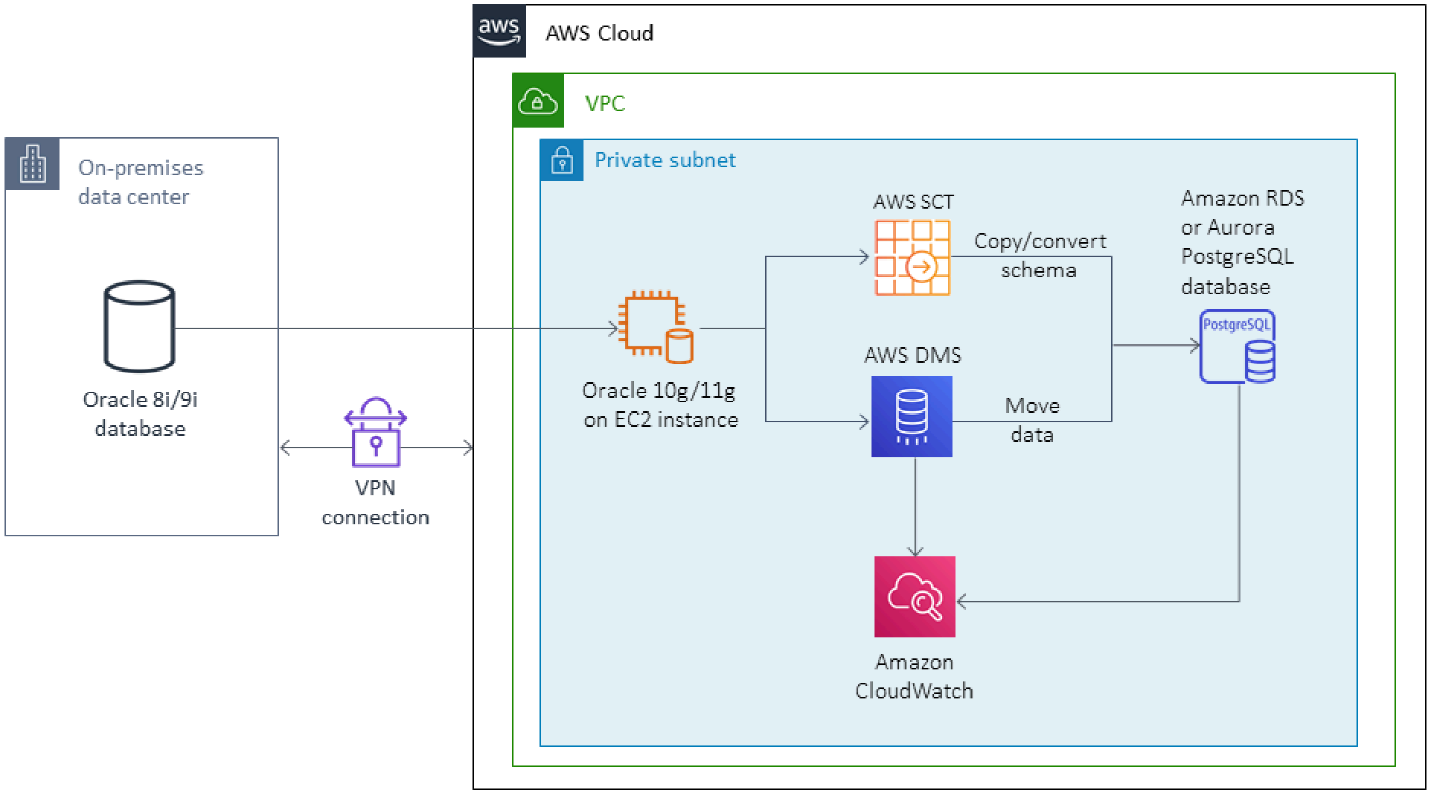
Source technology stack

- Oracle 8i or 9i database

Target technology stack

- Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible

Target architecture



Tools

- AWS DMS helps migrate databases quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases.
- AWS SCT automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format compatible with the target database. Objects that cannot be automatically converted are clearly marked so that they can be manually converted to complete the migration. AWS SCT can also scan your application source code for embedded SQL statements and convert them as part of a database schema conversion project. During this process, AWS SCT performs cloud-native code optimization by converting legacy Oracle and SQL Server functions to their AWS equivalents, to help you modernize your applications while migrating your databases. When schema conversion is complete, AWS SCT can help migrate data from a range of data warehouses to Amazon Redshift by using built-in data migration agents.

Best practices

For best practices for refreshing materialized views, see the following Oracle documentation:

- Refreshing materialized views
- Fast refresh for materialized views

Epics

Install Oracle on an EC2 instance and create materialized views

Task	Description	Skills required
Set up the network for the EC2 instance.	Create the virtual private cloud (VPC), subnets, internet gateway, route tables, and security groups.	AWS SysAdmin
Create the EC2 instance.	Select the Amazon Machine Image (AMI) for the EC2 instance. Choose the instance size and configure instance details: the number of instances (1), the VPC and subnet from the previous step, auto-assign public IP, and other options. Add storage, configure security groups, and launch the instance. When prompted, create and save a key pair for the next step.	AWS SysAdmin
Install Oracle on the EC2 instance.	Acquire the licenses and the required Oracle binaries, and install Oracle 10g or 11g on the EC2 instance.	DBA

Task	Description	Skills required
Configure Oracle networking.	Modify or add entries in `listener.ora` to connect to the on-premises source Oracle 8i/9i database, and then create the database links.	DBA
Create materialized views.	Identify the database objects to replicate in the source Oracle 8i/9i database, and then create materialized views for all the objects by using the database link.	DBA
Deploy scripts to refresh materialized views at required intervals.	Develop and deploy scripts to refresh materialized views at required intervals on the Amazon EC2 Oracle 10g/11g instance. Use the increment al refresh option to refresh materialized views.	DBA

Convert the Oracle database schema to PostgreSQL

Task	Description	Skills required
Set up AWS SCT.	Create a new report, and then connect to Oracle as the source and PostgreSQL as the target. In project settings, open the **SQL Scripting** tab. Change the target SQL script to **Multiple Files**. (AWS SCT doesn't support Oracle 8i/9i databases, so you have to	DBA

Task	Description	Skills required
	restore the schema-only dump on the intermediate Oracle 10g/11g instance and use it as a source for AWS SCT.)	
Convert the Oracle database schema.	On the **Action** tab, choose **Generate Report**, **Convert Schema**, and then **Save as SQL**.	DBA
Modify the SQL scripts.	Make modifications based on best practices. For example, switch to suitable data types and develop PostgreSQL equivalents for Oracle-sp ecific functions.	DBA, DevDBA

Create and configure the Amazon RDS DB instance to host the converted database

Task	Description	Skills required
Create the Amazon RDS DB instance.	In the Amazon RDS console, create a new PostgreSQL DB instance.	AWS SysAdmin, DBA
Configure the DB instance.	Specify the DB engine version, DB instance class, Multi-AZ deployment, storage type, and allocated storage. Enter the DB instance identifier, a master user name, and a master password.	AWS SysAdmin, DBA

Task	Description	Skills required
Configure network and security.	Specify the VPC, subnet group, public accessibility, Availability Zone preference, and security groups.	DBA, SysAdmin
Configure database options.	Specify the database name, port, parameter group, encryption, and master key.	DBA, AWS SysAdmin
Configure backups.	Specify the backup retention period, backup window, start time, duration, and whether to copy tags to snapshots.	AWS SysAdmin, DBA
Configure monitoring options.	Enable or disable enhanced monitoring and performance insights.	AWS SysAdmin, DBA
Configure maintenance options.	Specify auto minor version upgrade, maintenance window, and the start day, time, and duration.	AWS SysAdmin, DBA
Run the pre-migration scripts from AWS SCT.	On the target Amazon RDS for PostgreSQL instance, create the database schema by using the SQL scripts from AWS SCT with other modifications. These might include running multiple scripts and including user creation, database creation, schema creation, tables, views, functions, and other code objects.	AWS SysAdmin, DBA

Migrate data by using AWS DMS

Task	Description	Skills required
Create a replication instance in AWS DMS.	Complete the fields for the name, instance class, VPC (same as for the EC2 instance) , Multi-AZ, and public accessibility. In the advanced configuration section, specify allocated storage, subnet group, Availability Zone, VPC security groups, and AWS Key Management Service (AWS KMS) key.	AWS SysAdmin, DBA
Create the source database endpoint.	Specify the endpoint name, type, source engine (Oracle), server name (the EC2 instance's private DNS name), port, SSL mode, user name, password, SID, VPC (specify the VPC that has the replicati on instance), and replicati on instance. To test the connection, choose **Run Test**, and then create the endpoint. You can also configure the following advanced settings: **maxFileSize** and **numberDat aTypeScale**.	AWS SysAdmin, DBA
Connect AWS DMS to Amazon RDS for PostgreSQL.	Create a migration security group for connections across VPCs, if your PostgreSQL database is in another VPC.	AWS SysAdmin, DBA

Task	Description	Skills required
Create the target database endpoint.	Specify the endpoint name, type, source engine (PostgreSQL), server name (Amazon RDS endpoint), port, SSL mode, user name, password, database name, VPC (specify the VPC that has the replication instance), and replication instance. To test the connection, choose **Run Test**, and then create the endpoint. You can also configure the following advanced settings: **maxFileSize** and **numberDataTypeScale**.	AWS SysAdmin, DBA
Create the AWS DMS replication task.	Specify the task name, replication instance, source and target endpoints, and replication instance. For migration type, choose **Migrate existing data and replicate ongoing changes**. Clear the **Start task on create** check box.	AWS SysAdmin, DBA
Configure the AWS DMS replication task settings.	For target table preparation mode, choose **Do nothing**. Stop the task after full load completes (to create primary keys). Specify limited or full LOB mode, and enable control tables. Optionally, you can configure the **CommitRate** advanced setting.	DBA

Task	Description	Skills required
Configure the table mappings.	In the **Table mappings** section, create an Include rule for all tables in all schemas included in the migration, and then create an Exclude rule. Add three transformation rules to convert the schema, table, and column names to lowercase, and add any other rules you need for this specific migration.	DBA
Start the task.	Start the replication task. Make sure that the full load is running. Run `ALTER SYSTEM SWITCH LOGFILE` on the primary Oracle database to start the task.	DBA
Run the mid-migration scripts from AWS SCT.	In Amazon RDS for PostgreSQL, run the following scripts: `create_index.sql` and `create_constraint.sql` (if the complete schema wasn't initially created).	DBA
Resume the task to continue change data capture (CDC).	Run `VACUUM` on the Amazon RDS for PostgreSQL DB instance, and restart the AWS DMS task to apply cached CDC changes.	DBA

Cut over to the PostgreSQL database

Task	Description	Skills required
Check the AWS DMS logs and validation tables.	Check and fix any replication or validation errors.	DBA
Stop using the on-premises Oracle database and its dependencies.	Stop all Oracle dependencies, shut down listeners on the Oracle database, and run `ALTER SYSTEM SWITCH LOGFILE`. Stop the AWS DMS task when it shows no activity.	DBA
Run the post-migration scripts from AWS SCT.	In Amazon RDS for PostgreSQL, run these scripts: `create_foreign_key_constraint.sql` and `create_triggers.sql` . Make sure that the sequences are up to date.	DBA
Complete additional Amazon RDS for PostgreSQL steps.	Increment sequences to match Oracle if needed, run `VACUUM` and `ANALYZE`, and take a snapshot for compliance.	DBA
Open the connections to Amazon RDS for PostgreSQL.	Remove the AWS DMS security groups from Amazon RDS for PostgreSQL, add production security groups, and point your applications to the new database.	DBA
Clean up the AWS DMS objects.	Remove the endpoints, replication tasks, replicati	SysAdmin, DBA

Task	Description	Skills required
	on instances, and the EC2 instance.	

Related resources

- AWS DMS documentation
- AWS SCT documentation
- Amazon RDS for PostgreSQL pricing
- Using an Oracle database as a source for AWS DMS
- Using a PostgreSQL database as a target for AWS DMS

Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT

Anil Kunapareddy, Amazon Web Services

Harshad Gohil, None

Summary

Managing Oracle databases on Amazon Elastic Compute Cloud (Amazon EC2) instances requires resources and can be costly. Moving these databases to an Amazon Relational Database Service (Amazon RDS) for MySQL DB instance will ease your job by optimizing the overall IT budget. Amazon RDS for MySQL also provides features like Multi-AZ, scalability, and automatic backups.

This pattern walks you through the migration of a source Oracle database on Amazon EC2 to a target Amazon RDS for MySQL DB instance. It uses AWS Database Migration Service (AWS DMS) to migrate the data, and AWS Schema Conversion Tool (AWS SCT) to convert the source database schema and objects to a format that's compatible with Amazon RDS for MySQL.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source database with instance and listener services running, in ARCHIVELOG mode
- A target Amazon RDS for MySQL database, with sufficient storage for data migration

Limitations

- AWS DMS does not create a schema on the target database; you have to do that. The schema name must already exist for the target. Tables from the source schema are imported to user/schema, which AWS DMS uses to connect to the target instance. You must create multiple replication tasks if you have to migrate multiple schemas.

Product versions

- All Oracle database editions for versions 10.2 and later, 11g and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and

Using a MySQL-Compatible Database as a Target for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.

- AWS DMS supports versions 5.5, 5.6, and 5.7 of MySQL.

Architecture

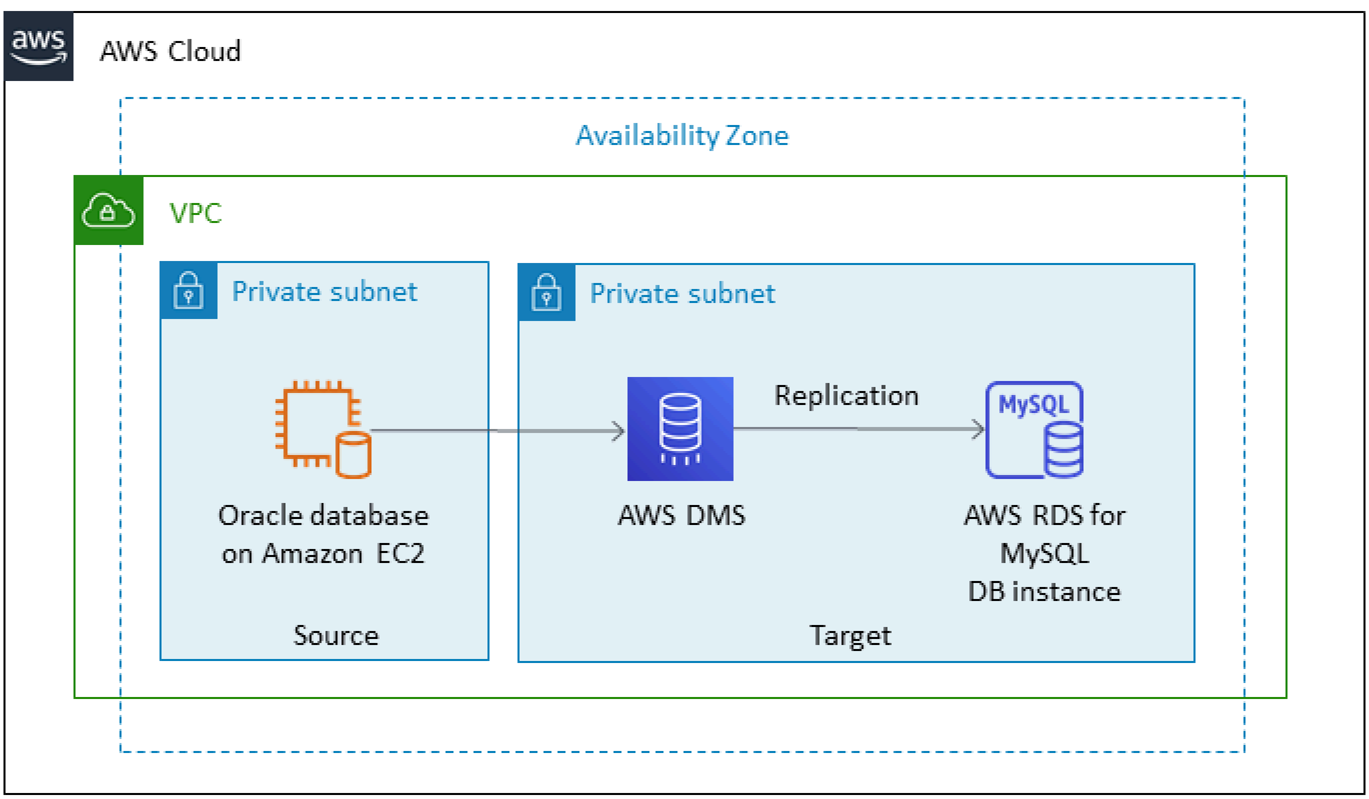
Source technology stack

- An Oracle database on an EC2 instance

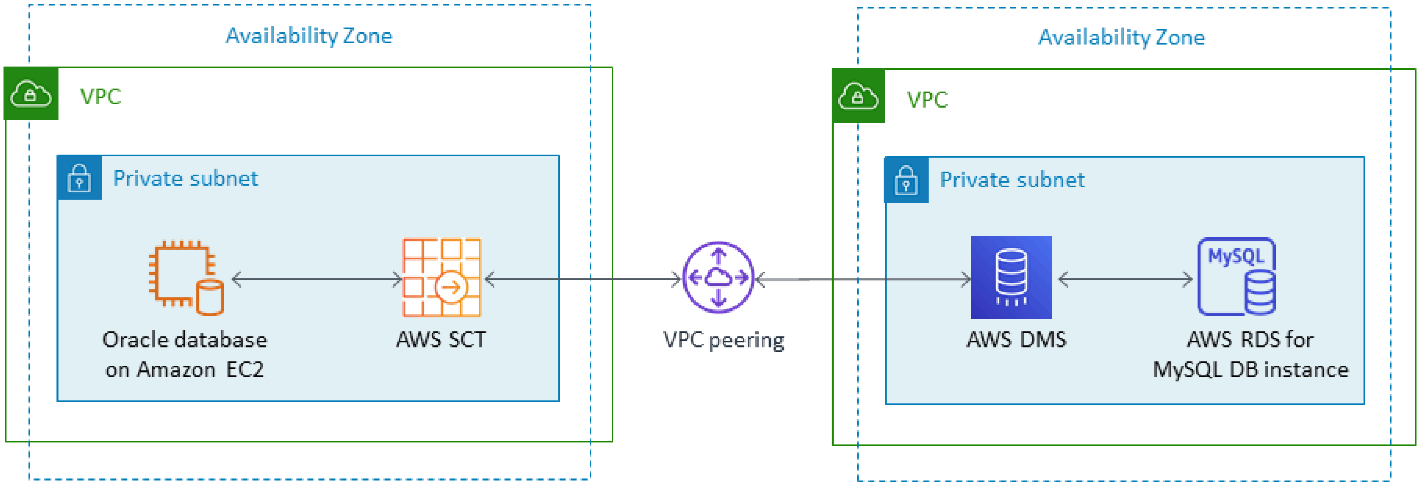
Target technology stack

- Amazon RDS for MySQL DB instance

Data migration architecture



Source and target architecture



Tools

- **AWS DMS** - [AWS Database Migration Service](#) (AWS DMS) is a web service you can use to migrate data from your database that is on-premises, on an Amazon RDS DB instance, or in a database on an EC2 instance, to a database on an AWS service such as Amazon RDS for MySQL or an EC2 instance. You can also migrate a database from an AWS service to an on-premises database. You can migrate data between heterogeneous or homogeneous database engines.
- **AWS SCT** - [AWS Schema Conversion Tool](#) (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects, including views, stored procedures, and functions, to a format that's compatible with the target database. After converting your database schema and code objects using AWS SCT, you can use AWS DMS to migrate data from the source database to the target database to complete your migration projects.

Epics

Plan the migration

Task	Description	Skills required
Identify the source and target database versions and engines.		DBA/Developer

Task	Description	Skills required
Identify the DMS replication instance.		DBA/Developer
Identify storage requireme nts such as storage type and capacity.		DBA/Developer
Identify network requireme nts such as latency and bandwidth.		DBA/Developer
Identify hardware requireme nts for the source and target server instances (based on Oracle compatibility list and capacity requirements).		DBA/Developer
Identify network access security requirements for source and target databases.		DBA/Developer
Install AWS SCT and Oracle drivers.		DBA/Developer
Determine a backup strategy.		DBA/Developer
Determine availability requirements.		DBA/Developer
Identify application migration and switch-over strategy.		DBA/Developer
Select the proper DB instance type based on capacity, storage, and network features.		DBA/Developer

Configure the environment

Task	Description	Skills required
Create a virtual private cloud (VPC). The source, target, and replication instance should be in the same VPC. It is also good to have these in the same Availability Zone.		Developer
Create the necessary security groups for database access.		Developer
Generate and configure a key pair.		Developer
Configure subnets, Availability Zones, and CIDR blocks.		Developer

Configure the source: Oracle database on EC2 instance

Task	Description	Skills required
Install Oracle Database on Amazon EC2 with required users and roles.		DBA
Perform the three steps in the next column to access Oracle from outside the EC2 instance.	1. Change the local host in `tnsnames` to the Amazon EC2 public DNS. 2. Change the local host in `listener` to the Amazon EC2 public DNS. 3. Stop and restart the listener.	DBA

Task	Description	Skills required
When Amazon EC2 is restarted, the public DNS changes. Make sure to update Amazon EC2 public DNS in 'tnsnames' and 'listener' or use an Elastic IP address.		DBA/Developer
Configure the EC2 instance security group so that the replication instance and required clients can access the source database.		DBA/Developer

Configure the target: Amazon RDS for MySQL

Task	Description	Skills required
Configure and start the Amazon RDS for MySQL DB instance.		Developer
Create the necessary tablespace in the Amazon RDS for MySQL DB instance.		DBA
Configure the security group so that the replicati on instance and required clients can access the target database.		Developer

Configure AWS SCT and create a schema in the target database

Task	Description	Skills required
Install AWS SCT and Oracle drivers.		Developer
Enter the appropriate parameters and connect to the source and target.		Developer
Generate a schema conversio n report.		Developer
Correct the code and schema as necessary, especially tablespaces and quotes, and run on the target database.		Developer
Validate the schema on source vs. target before migrating data.		Developer

Migrate data using AWS DMS

Task	Description	Skills required
For full-load and change data capture (CDC) or just CDC, you must set an extra connection attribute.		Developer
The user specified in the AWS DMS source Oracle database definitions must be granted all the required privilege s. For a complete list, see https://docs.aws.amazon.com		DBA/Developer

Task	Description	Skills required
/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Self-Managed.		
Enable supplemental logging in the source database.		DBA/Developer
For full-load and change data capture (CDC) or just CDC, enable ARCHIVELOG mode in the source database.		DBA
Create source and target endpoints, and test the connections.		Developer
When the endpoints are connected successfully, create a replication task.		Developer
Select CDC only (or) full load plus CDC in the task to capture changes for continuous replication only (or) full load plus ongoing changes, respectively.		Developer
Run the replication task and monitor Amazon CloudWatch logs.		Developer
Validate the data in the source and target databases.		Developer

Migrate your application and cut over

Task	Description	Skills required
Follow the steps for your application migration strategy.		DBA, Developer, App owner
Follow the steps for your application cutover/switch-over strategy.		DBA, Developer, App owner

Close the project

Task	Description	Skills required
Validate the schema and data in source vs. target databases.		DBA/Developer
Gather metrics around time to migrate, percent of manual vs. tool, cost savings, etc.		DBA/Developer/AppOwner
Review the project documents and artifacts.		DBA/Developer/AppOwner
Shut down temporary AWS resources.		DBA/Developer
Close out the project and provide feedback.		DBA/Developer/AppOwner

Related resources

- AWS DMS documentation
- AWS DMS website
- AWS DMS blog posts

- Strategies for Migrating Oracle Database to AWS
- Amazon RDS for Oracle FAQs
- Oracle FAQ
- Amazon EC2
- Amazon EC2 FAQs
- Licensing Oracle Software in the Cloud Computing Environment

Migrate from Oracle to Amazon DocumentDB using AWS DMS

Sashikanta Pattanayak and Munesh Siddappa, Amazon Web Services

Summary

This pattern provides guidance for migrating an Oracle database to an Amazon DocumentDB (with MongoDB compatibility) database by using AWS Database Migration Service (AWS DMS). This approach can be applied to an on-premises Oracle source database as well as an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance. This pattern uses an Amazon RDS Oracle DB source instance as an example.

Amazon DocumentDB (with MongoDB compatibility) is a fully managed, MongoDB-compatible document database service that makes it easy to store, query, and index JSON data.

The use case for this pattern is one-to-one replication of an Oracle database table to an Amazon DocumentDB collection. The pattern uses AWS DMS replication tasks to read the table structure of the Oracle database, create the corresponding collection in Amazon DocumentDB, and perform a full load migration. You can view and query your data in Amazon DocumentDB, the same as you would in MongoDB.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Familiarity with using Oracle databases
- Familiarity with using Amazon DocumentDB
- For the Oracle user, `SELECT ANY TABLE` privilege
- For the Amazon DocumentDB use, the privilege required to dump data

Limitations

The following limitations apply when using Amazon DocumentDB as a target for AWS DMS:

- In Amazon DocumentDB, collection names can't contain the dollar symbol ($). In addition, database names can't contain any Unicode characters.

- AWS DMS doesn't support merging of multiple source tables into a single Amazon DocumentDB collection.
- When AWS DMS processes changes from a source table that doesn't have a primary key, any large binary object (LOB) columns in that table are ignored.
- If the **Change table** option is enabled and AWS DMS encounters a source column named "_id", that column appears as "__id" (two underscores) in the change table.
- If you choose Oracle as a source endpoint, the Oracle source must have full supplemental logging enabled. Otherwise, if there are columns at the source that weren't changed, the data is loaded into Amazon DocumentDB as null values.

Product versions

- Amazon RDS for Oracle version 11.2.0.3 or later
- AWS DMS version 3.1.3 or later (for the latest version information, see Using Amazon DocumentDB as a Target for AWS DMS in the AWS DMS documentation)

Architecture

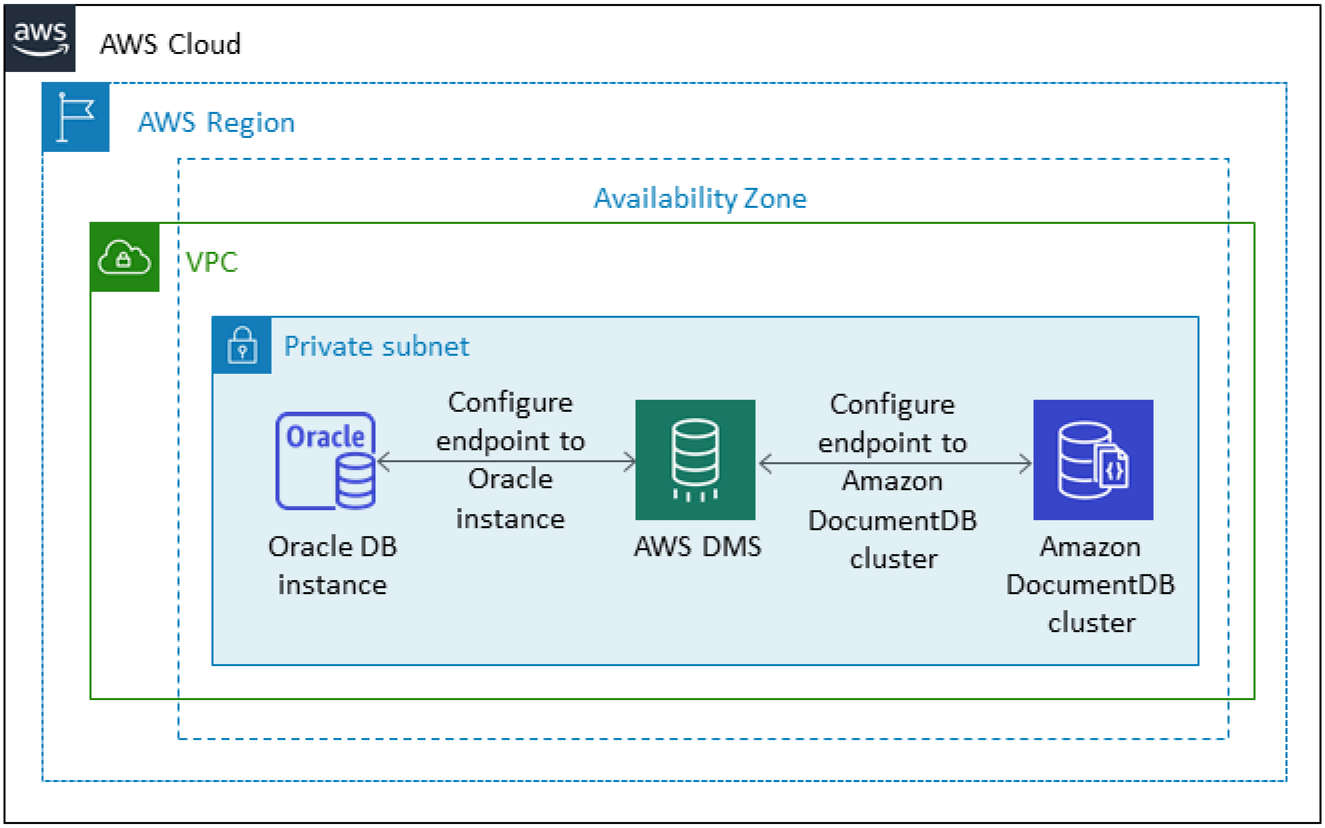
Source technology stack

- Amazon RDS for Oracle DB instance

Target technology stack

- Amazon DocumentDB

Source and target architecture



Tools

- **AWS DMS** – AWS Database Migration Service (AWS DMS) is a web service that you can use to migrate data from a source data store to a target data store. The AWS DMS User Guide specifies the Oracle source database versions and editions that are supported for use with AWS DMS. For additional information relevant to this pattern, see Using Amazon DocumentDB as a Target for AWS DMS.
- **Amazon EC2** – Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS cloud. Your Amazon DocumentDB cluster should be running in your default virtual private cloud (VPC). To interact with your Amazon DocumentDB cluster, you must launch an EC2 instance into your default VPC, in the same AWS Region where you created your Amazon DocumentDB cluster. For details, refer to Launch an Amazon EC2 instance in the Amazon DocumentDB documentation.

Epics

Plan the migration

Task	Description	Skills required
Validate the source and target database versions and engines.		AWS Admin
Choose the proper instance type (capacity, storage features, network features).		AWS Admin
Identify network/host access security requirements for the source and target databases.		AWS Admin
Create an outbound security group to the source and target databases.		AWS Admin
Create and configure an EC2 instance for Amazon DocumentDB.		AWS Admin

Configure infrastructure

Task	Description	Skills required
Create a VPC and subnets.		AWS Admin
Create security groups and network access control lists (ACLs).		AWS Admin

Task	Description	Skills required
Configure and start the source Amazon RDS for Oracle instance.		AWS Admin
Configure and start the Amazon DocumentDB instance.		AWS Admin

Prepare the source database

Task	Description	Skills required
Verify that the Oracle database can be connected using the connection details.		AWS Admin
Verify that the Oracle user has the SELECT ANY TABLE privilege.		AWS Admin

Prepare the target database

Task	Description	Skills required
Create the Amazon DocumentDB cluster by choosing the proper instance class and number of instances .		AWS Admin

Configure Amazon EC2

Task	Description	Skills required
Configure the EC2 instance.	To interact with your Amazon DocumentDB cluster, you must launch an EC2 instance into your default VPC, in the same AWS Region where you created your Amazon DocumentDB cluster. Configure the AWS Region, VPCs, Availability Zones, and subnets for the EC2 instance.	AWS Admin
Configure the key pair.	A public/private key pair allows you to connect securely to the EC2 instance after it launches.	AWS Admin
Set the bastion host CIDR ranges (optional).	Set the CIDR IP range that is allowed for external Secure Shell (SSH) access to the bastion host instances.	AWS Admin

Migrate data – full load

Task	Description	Skills required
Create an AWS DMS replicati on instance.		AWS Admin
Create source and target endpoints.		AWS Admin
Create AWS DMS replication tasks for a full load.		AWS Admin

Test the migration

Task	Description	Skills required
Connect to the Amazon DocumentDB cluster through the EC2 instance.		AWS Admin
Connect to the cluster using the mongo shell.	For instructions, see the Amazon DocumentDB links in the References and Help section.	AWS Admin
Verify the results of the migration.		AWS Admin

Related resources

- How AWS DMS Works
- Migrating to Amazon DocumentDB
- Using Amazon DocumentDB as a Target for AWS DMS
- Amazon DocumentDB overview
- Access and Use Your Amazon DocumentDB cluster Using the mongo Shell
- Migrate from MongoDB to Amazon DocumentDB using the offline method (blog post)
- How to use Amazon DocumentDB (with MongoDB compatibility) to build and manage applications at scale (blog post)

Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT

Veeranjaneyulu Grandhi and vinod kumar, Amazon Web Services

Summary

This pattern walks you through the steps for migrating an Oracle database on an Amazon Elastic Compute Cloud (Amazon EC2) instance to an Amazon Relational Database Service (Amazon RDS) for MariaDB DB instance. The pattern uses AWS Data Migration Service (AWS DMS) for data migration and AWS Schema Conversion Tool (AWS SCT) for schema conversion.

Managing Oracle databases on EC2 instances requires more resources and is more costly than using a database on Amazon RDS. Amazon RDS makes it easy to set up, operate, and scale a relational database in the cloud. Amazon RDS provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching, and backups.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A source Oracle database with instance and listener services up and running. This database should be in ARCHIVELOG mode.
- Familiarity with Using an Oracle Database as a Source for AWS DMS.
- Familiarity with Using Oracle as a Source for AWS SCT.

Limitations

- Database size limit: 64 TB

Product versions

- All Oracle database editions for versions 10.2 and later, 11g and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS and the AWS SCT version table in the AWS documentation.

- Amazon RDS supports MariaDB Server Community Server versions 10.3, 10.4, 10.5, and 10.6. For the latest list of supported versions, see the Amazon RDS documentation.

Architecture

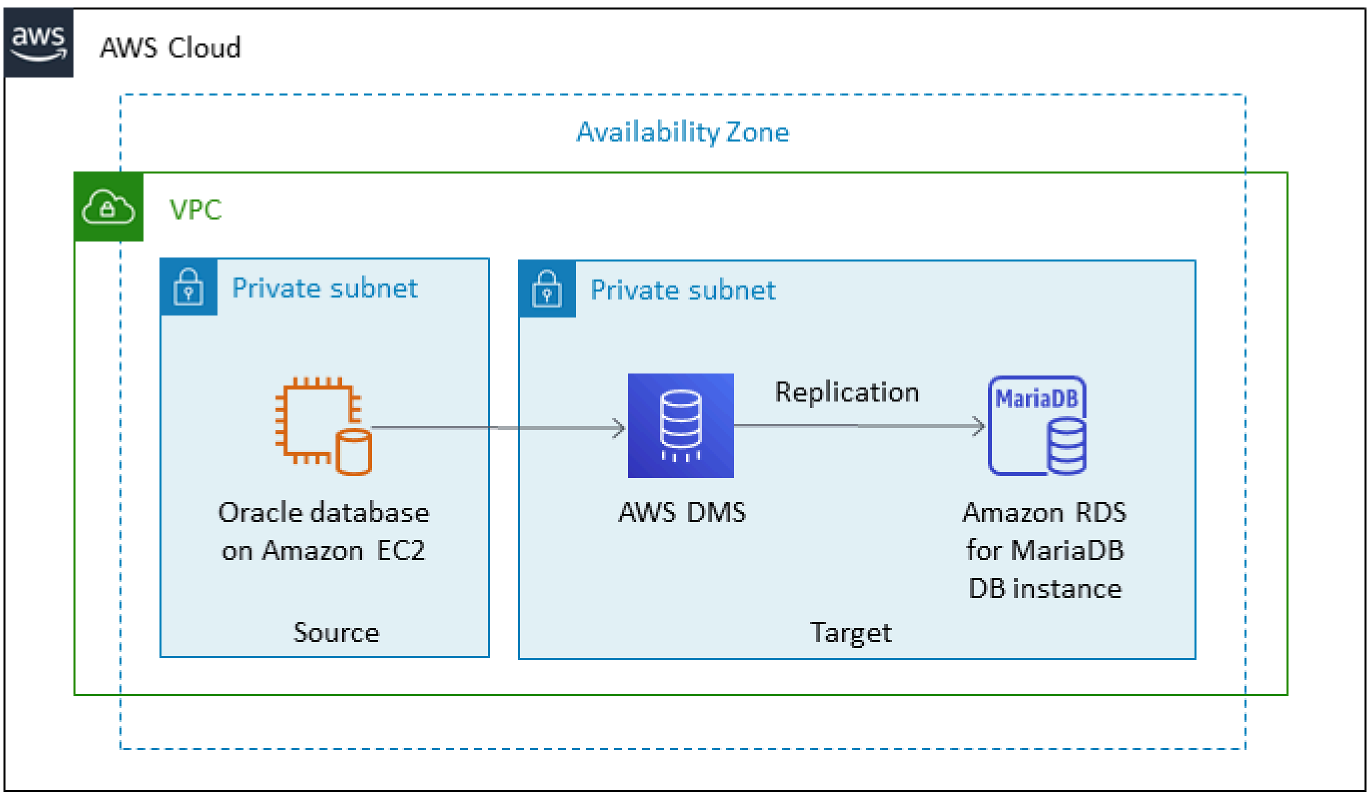
Source technology stack

- An Oracle database on an EC2 instance

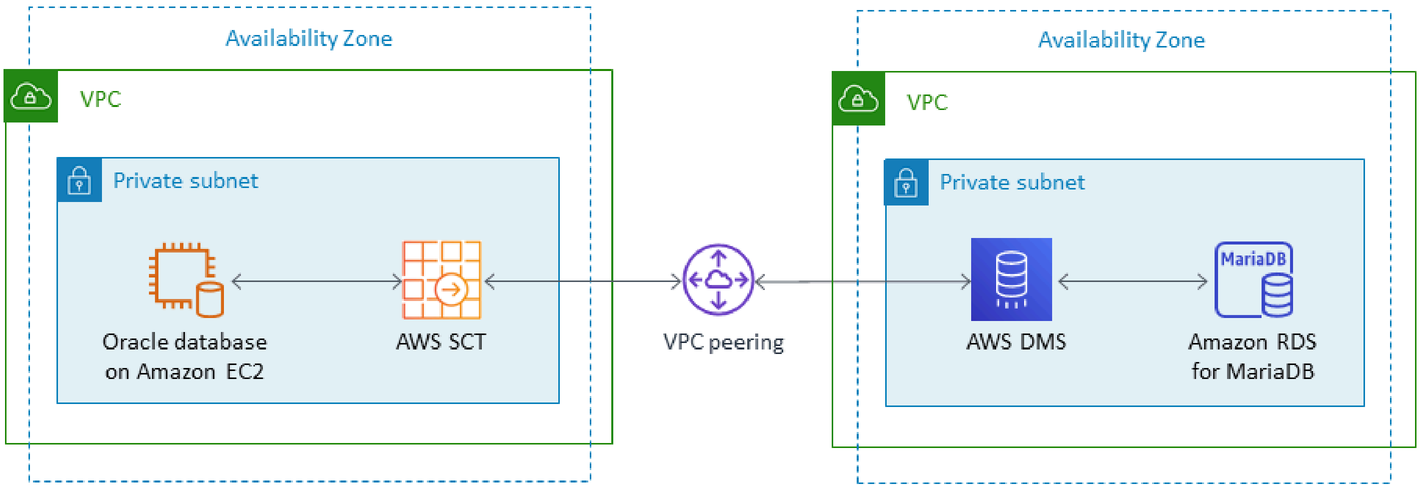
Target technology stack

- Amazon RDS for MariaDB

Data migration architecture



Target architecture



Tools

- AWS Schema Conversion Tool (AWS SCT) makes heterogeneous database migrations predictable by automatically converting the source database schema and a majority of the database code objects—including views, stored procedures, and functions—to a format compatible with the target database. After converting your database schema and code objects using AWS SCT, you can use AWS DMS to migrate data from the source database to the target database to complete your migration projects. For more information, see Using Oracle as a Source for AWS SCT in the AWS SCT documentation.
- AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from the most widely used commercial and open-source databases. AWS DMS supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora. To learn more about migrating Oracle databases, see Using an Oracle Database as a Source for AWS DMS in the AWS DMS documentation.

Epics

Plan for the migration

Task	Description	Skills required
Identify versions and database engines.	Identify the source and target database versions and engines.	DBA, Developer
Identify the replication instance.	Identify the AWS DMS replication instance.	DBA, Developer
Identify storage requireme nts.	Identify storage type and capacity.	DBA, Developer
Identify network requireme nts.	Identify network latency and bandwidth.	DBA, Developer
Identify hardware requireme nts.	Identify hardware requireme nts for the source and target server instances (based on the Oracle compatibility list and capacity requirements).	DBA, Developer
Identify security requireme nts.	Identify network-access security requirements for the source and target databases.	DBA, Developer
Install drivers.	Install the latest AWS SCT and Oracle drivers.	DBA, Developer
Determine a backup strategy.		DBA, Developer
Determine availability requirements.		DBA, Developer

Task	Description	Skills required
Choose an application migration/switchover strategy.		DBA, Developer
Select the instance type.	Select the proper instance type based on capacity, storage, and network features.	DBA, Developer

Configure the environment

Task	Description	Skills required
Create a virtual private cloud (VPC).	The source, target, and replication instances should be in the same VPC and in the same Availability Zone (recommended).	Developer
Create security groups.	Create the necessary security groups for database access.	Developer
Generate a key pair.	Generate and configure a key pair.	Developer
Configure other resources.	Configure subnets, Availabil ity Zones, and CIDR blocks.	Developer

Configure the source

Task	Description	Skills required
Launch the EC2 instance.	For instructions, see the [Amazon EC2 documentation]().	Developer

Task	Description	Skills required
Install the Oracle database.	Install the Oracle database on the EC2 instance, with required users and roles.	DBA
Follow the steps in the task description to access Oracle from outside of the EC2 instance.	1. Change the local host in `tnsnames` to the Amazon EC2 public DNS. 2. Change the local host in `listener` to the Amazon EC2 public DNS. 3. Stop and restart the listener.	DBA
Update the Amazon EC2 public DNS.	After the EC2 instance restarts, the public DNS changes. Make sure to update the Amazon EC2 public DNS in `tnsnames` and `listener`, or use an Elastic IP address.	DBA, Developer
Configure the EC2 instance security group.	Configure the EC2 instance security group so the replicati on instance and required clients can access the source database.	DBA, Developer

Configure the target Amazon RDS for MariaDB environment

Task	Description	Skills required
Start the RDS DB instance.	Configure and start the Amazon RDS for MariaDB DB instance.	Developer

Task	Description	Skills required
Create tablespaces.	Create any necessary tablespaces in the Amazon RDS MariaDB database.	DBA
Configure a security group.	Configure a security group so the replication instance and required clients can access the target database.	Developer

Configure AWS SCT

Task	Description	Skills required
Install drivers.	Install the latest AWS SCT and Oracle drivers.	Developer
Connect.	Enter appropriate parameter s and then connect to the source and target.	Developer
Generate a schema conversio n report.	Generate an AWS SCT schema conversion report.	Developer
Correct the code and schema as necessary.	Make any necessary correctio ns to the code and schema (especially tablespaces and quotation marks).	DBA, Developer
Validate the schema.	Validate the schema on the source versus the target before loading data.	Developer

Migrate data using AWS DMS

Task	Description	Skills required
Set a connection attribute.	For full-load and change data capture (CDC) or just for CDC, set an extra connection attribute. For more informati on, see the [Amazon RDS documentation]().	Developer
Enable supplemental logging.	Enable supplemental logging on the source database.	DBA, Developer
Enable archive log mode.	For full-load and CDC (or just for CDC), enable archive log mode on the source database.	DBA
Create and test endpoints.	Create source and target endpoints and test the connections. For more information, see the [Amazon DMS documentation]().	Developer
Create a replication task.	When the endpoints are connected successfully, create a replication task. For more information, see the [Amazon DMS documentation]().	Developer
Choose replication type.	Choose **CDC only** or **Full load plus CDC** in the task to capture changes for continuous replication only, or for full load and ongoing changes, respectively.	Developer

Task	Description	Skills required
Start and monitor the task.	Start the replication task and monitor Amazon CloudWatch logs. For more information, see the [Amazon DMS documentation]().	Developer
Validate the data.	Validate the data in the source and target databases.	Developer

Migrate applications and cut over to the target database

Task	Description	Skills required
Follow the chosen application migration strategy.		DBA, App owner, Developer
Follow the chosen application cutover/switchover strategy.		DBA, App owner, Developer

Close the project

Task	Description	Skills required
Validate the schema and data.	Ensure that the schema and data are validated successfully in the source versus the target before project closure.	DBA, Developer
Gather metrics.	Gather metrics for time to migrate, percentage of manual versus tool tasks, cost savings, and similar criteria.	DBA, App owner, Developer

Task	Description	Skills required
Review documentation.	Review the project documents and artifacts.	DBA, App owner, Developer
Shut down resources.	Shut down temporary AWS resources.	DBA, Developer
Close the project.	Close the migration project and provide any feedback.	DBA, App owner, Developer

Related resources

- MariaDB Amazon RDS overview
- Amazon RDS for MariaDB product details
- Using an Oracle Database as a Source for AWS DMS
- Strategies for Migrating Oracle Databases to AWS
- Licensing Oracle Software in the Cloud Computing Environment
- Amazon RDS for Oracle FAQs
- AWS DMS overview
- AWS DMS blog posts
- Amazon EC2 overview
- Amazon EC2 FAQs
- AWS SCT documentation

Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT

Sergey Dmitriev and Naresh Damera, Amazon Web Services

Summary

This pattern walks you through the migration of an on-premises Oracle database to an Amazon Relational Database Service (Amazon RDS) for MySQL DB instance. It uses AWS Database Migration Service (AWS DMS) to migrate the data, and AWS Schema Conversion Tool (AWS SCT) to convert the source database schema and objects to a format that's compatible with Amazon RDS for MySQL.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- All Oracle database editions for versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.
- AWS DMS currently supports MySQL versions 5.5, 5.6, and 5.7. For the latest list of supported versions, see Using a MySQL-Compatible Database as a Target for AWS DMS in the AWS documentation.

Architecture

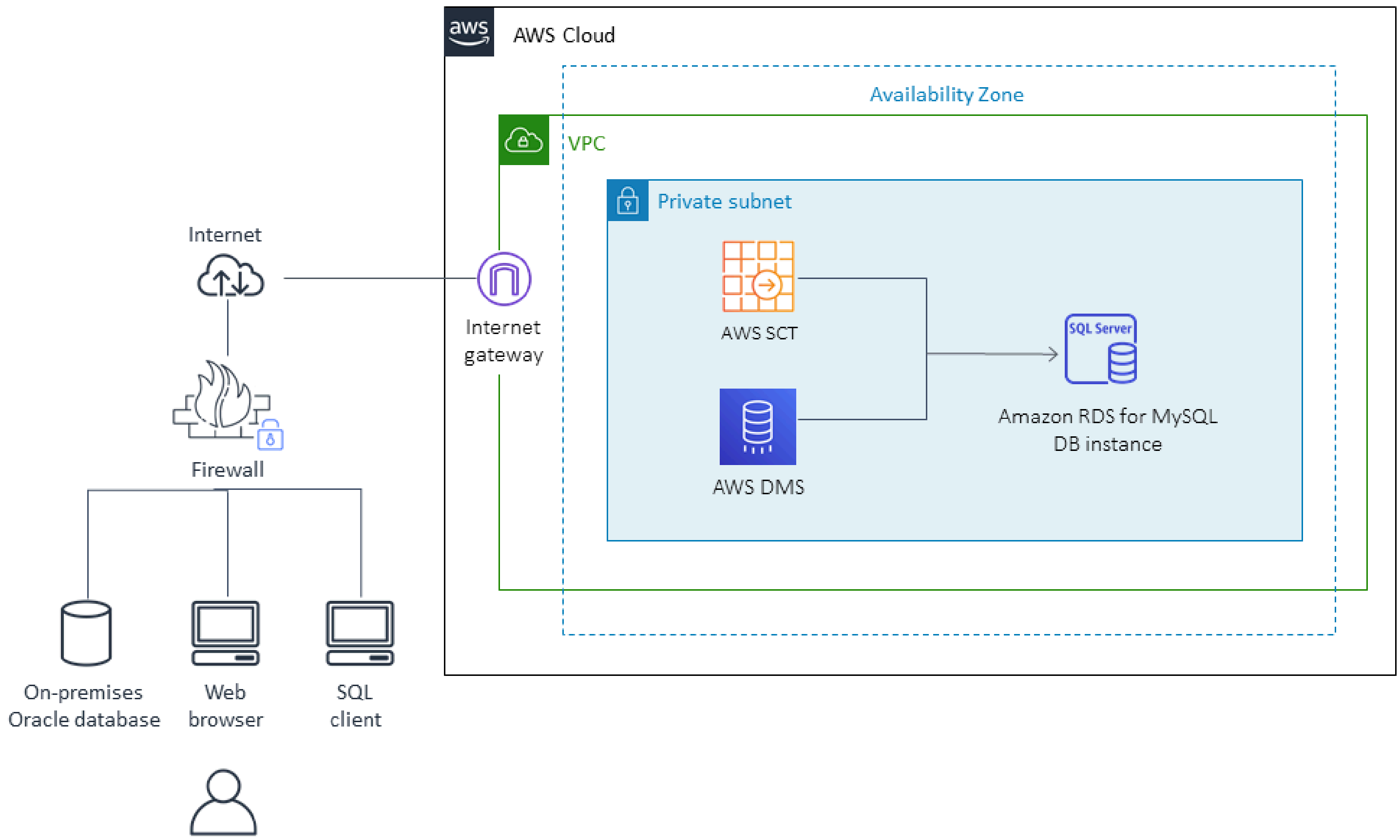
Source technology stack

- On-premises Oracle database

Target technology stack

- Amazon RDS for MySQL DB instance

Data migration architecture



Tools

- **AWS DMS** - [AWS Database Migration Services](https://docs.aws.amazon.com/dms/) (AWS DMS) helps you migrate relational databases, data warehouses, NoSQL databases, and other types of data stores. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups.
- **AWS SCT** - [AWS Schema Conversion Tool](https://docs.aws.amazon.com/SchemaConversionTool/) (AWS SCT) is used to convert your database schema from one database engine to another. The custom code that the tool converts includes views, stored procedures, and functions. Any code that the tool cannot convert automatically is clearly marked so that you can convert it yourself.

Epics

Plan the migration

Task	Description	Skills required
Validate the source and target database version and engine.		DBA
Identify the hardware requirements for the target server instance.		DBA, SysAdmin
Identify the storage requirements (storage type and capacity).		DBA, SysAdmin
Choose the proper instance type based on capacity, storage features, and network features.		DBA, SysAdmin
Identify the network access security requirements for the source and target databases.		DBA, SysAdmin
Identify the application migration strategy.		DBA, SysAdmin, App owner

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC) and subnets.		SysAdmin

Task	Description	Skills required
Create the security groups and network access control lists (ACLs).		SysAdmin
Configure and start an Amazon RDS DB instance.		DBA, SysAdmin

Migrate data

Task	Description	Skills required
Migrate the database schema by using AWS SCT.		DBA
Migrate data by using AWS DMS.		DBA

Migrate the application

Task	Description	Skills required
Use AWS SCT to analyze and convert the SQL code inside the application code.	For more information, see https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Converting.App.html.	App owner
Follow the application migration strategy.		DBA, SysAdmin, App owner

Cut over

Task	Description	Skills required
Switch the application clients over to the new infrastru cture.		DBA, SysAdmin, App owner

Close the project

Task	Description	Skills required
Shut down the temporary AWS resources.		DBA, SysAdmin
Review and validate the project documents.		DBA, SysAdmin
Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.		DBA, SysAdmin
Close out the project and provide feedback.		

Related resources

References

- AWS DMS documentation
- AWS SCT documentation
- Amazon RDS Pricing

Tutorial and videos

- Getting Started with AWS DMS

- Getting Started with Amazon RDS
- AWS DMS (video)
- Amazon RDS (video)

Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL by using an Oracle bystander and AWS DMS

Cady Motyka, Amazon Web Services

Summary

This pattern describes how you can migrate an on-premises Oracle database to either of the following PostgreSQL-compatible AWS database services with minimal downtime:

- Amazon Relational Database Service (Amazon RDS) for PostgreSQL
- Amazon Aurora PostgreSQL-Compatible Edition

The solution uses AWS Database Migration Service (AWS DMS) to migrate the data, AWS Schema Conversion Tool (AWS SCT) to convert the database schema, and an Oracle bystander database to help manage the migration. In this implementation, the downtime is restricted to however long it takes to create or validate all of the foreign keys on the database.

The solution also uses Amazon Elastic Compute Cloud (Amazon EC2) instances with an Oracle bystander database to help control the stream of data through AWS DMS. You can temporarily pause streaming replication from the on-premises Oracle database to the Oracle bystander to activate AWS DMS to catch up on data validation, or to use another data validation tool. The Amazon RDS for PostgreSQL DB instance or Aurora PostgreSQL-Compatible DB instance and the bystander database will have the same data when AWS DMS finishes migrating current changes.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle database in an on-premises data center with an Active Data Guard standby database configured
- AWS Direct Connect configured between the on-premises data center and AWS Secrets Manager for storing the database secrets
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, installed either on a local machine or on the EC2 instance where AWS SCT is installed
- Familiarity with using an Oracle database as a source for AWS DMS

- Familiarity with using a PostgreSQL database as a target for AWS DMS

Limitations

- Database size limit: 64 TB

Product versions

- AWS DMS supports all Oracle database editions for versions 10.2 and later (for versions 10.x), 11g and up to 12.2, 18c, and 19c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.
- AWS DMS supports PostgreSQL versions 9.4 and later (for versions 9.x), 10.x, 11.x, 12.x, and 13.x. For the latest information, see Using a PostgreSQL Database as a Target for AWS DMS in the AWS documentation.

Architecture

Source technology stack

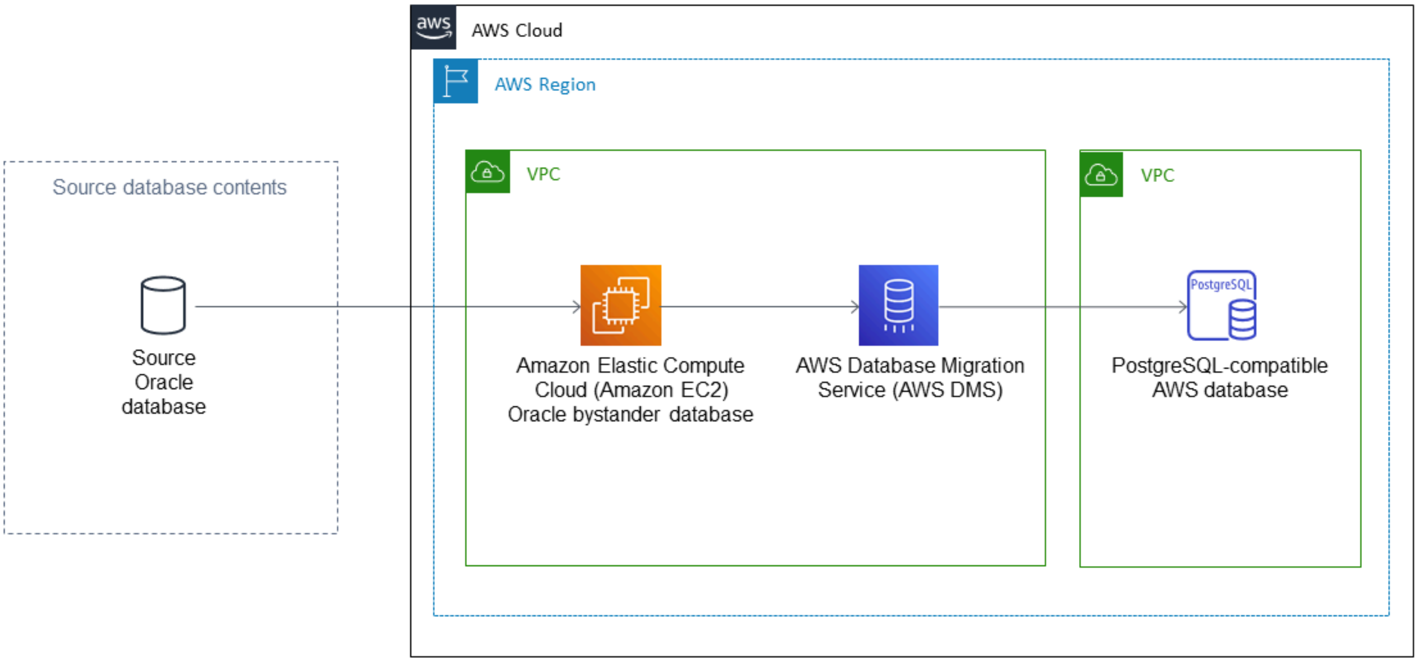
- An on-premises Oracle database
- An EC2 instance that holds a bystander for the Oracle database

Target technology stack

- Amazon RDS for PostgreSQL or Aurora PostgreSQL instance, PostgreSQL 9.3 and later

Target architecture

The following diagram shows an example workflow for migrating an Oracle database to a PostgreSQL-compatible AWS database by using AWS DMS and an Oracle bystander:



Tools

- [AWS Database Migration Service (AWS DMS)](#) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- [AWS Schema Conversion Tool (AWS SCT)](#) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.
- [Amazon Relational Database Service (Amazon RDS)](#) helps you set up, operate, and scale a relational database in the AWS Cloud.

Epics

Convert the Oracle database schema to PostgreSQL

Task	Description	Skills required
Set up AWS SCT.	Create a new report, and connect to Oracle as the source and PostgreSQL as the target. In **Project Settings**, go to the **SQL Scripting** tab.	DBA

Task	Description	Skills required
	Change the **Target SQL Script** to **Multiple Files**. These files will be used later and named the following: • create_database.sql • create_sequence.sql • create_table.sql • create_view.sql • create_function.sql	
Convert the Oracle database schema.	In the **Action** tab, choose **Generate Report**. Then, choose **Convert Schema** and choose **Save as SQL**.	DBA
Modify the scripts.	For example, you might want to modify the script if a number in the source schema has been converted to numeric format in PostgreSQ L, but you want to use **BIGINT** instead for better performan ce.	DBA

Create and configure the Amazon RDS DB instance

Task	Description	Skills required
Create the Amazon RDS DB instance.	In the correct AWS Region, create a new PostgreSQL DB instance. For more informati on, see [Creating a PostgreSQ L DB instance and connecting](#)	AWS SysAdmin, DBA

Task	Description	Skills required
	to a database on a PostgreSQL DB instance in the Amazon RDS documentation.	
Configure DB instance specifications.	Specify the DB engine version, DB instance class, Multi-AZ deployment, storage type, and allocated storage. Enter the DB instance identifier, a primary user name, and a primary password.	AWS SysAdmin, DBA
Configure network and security.	Specify the virtual private cloud (VPC), subnet group, public accessibility, Availability Zone preference, and security groups.	DBA, SysAdmin
Configure database options.	Specify the database name, port, parameter group, encryption, and KMS key.	AWS SysAdmin, DBA
Configure backups.	Specify the backup retention period, backup window, start time, duration, and whether to copy tags to snapshots.	AWS SysAdmin, DBA
Configure monitoring options.	Activate or deactivate enhanced monitoring and performance insights.	AWS SysAdmin, DBA
Configure maintenance options.	Specify auto minor version upgrade, maintenance window, and start day, time, and duration.	AWS SysAdmin, DBA

Task	Description	Skills required
Run the pre-migration scripts from AWS SCT.	On the Amazon RDS instance, run the following scripts generated by AWS SCT: • create_database.sql • create_sequence.sql • create_table.sql • create_view.sql • create_function.sql	AWS SysAdmin, DBA

Configure the Oracle bystander in Amazon EC2

Task	Description	Skills required
Set up the network for Amazon EC2.	Create the new VPC, subnets, internet gateway, route tables, and security groups.	AWS SysAdmin
Create the EC2 instance.	In the appropriate AWS Region, create a new EC2 instance. Select the Amazon Machine Image (AMI), choose the instance size, and configure instance details: number of instances (1), the VPC and subnet you created in the previous task, auto-assign public IP, and other options. Add storage, configure security groups, and launch. When prompted, create and save a key pair for the next step.	AWS SysAdmin

Task	Description	Skills required
Connect the Oracle source database to the EC2 instance.	Copy the IPv4 public IP address and DNS to a text file and connect by using SSH as follows: **ssh -i "your_file.pem" ec2-user@<your-IP-address-or-public-DNS>**.	AWS SysAdmin
Set up the initial host for a bystander in Amazon EC2.	Set up SSH keys, bash profile, ORATAB, and symbolic links. Create Oracle directories.	AWS SysAdmin, Linux Admin
Set up the database copy for a bystander in Amazon EC2	Use RMAN to create a database copy, enable supplemental logging, and create the standby control file. After copying is complete, place the database in recovery mode.	AWS SysAdmin, DBA
Set up Oracle Data Guard.	Modify your **listener.ora** file and start the listener. Set up a new archive destination. Place the bystander in recovery mode, replace temporary files to avoid future corruption, install a crontab if necessary to prevent the archive directory from running out of space, and edit the **manage-trclog-files-oracle.cfg** file for the source and standby.	AWS SysAdmin, DBA

Task	Description	Skills required
Prep the Oracle database to sync shipping.	Add the standby log files and change the recovery mode. Change the log shipping to **SYNC AFFIRM** on both the source primary and the source standby. Switch logs on primary, confirm via the Amazon EC2 bystander alert log that you are using the standby log files, and confirm that the redo stream is flowing in SYNC.	AWS SysAdmin, DBA

Migrate data with AWS DMS

Task	Description	Skills required
Create a replication instance in AWS DMS.	Complete the fields for the name, instance class, VPC (same as the Amazon EC2 instance), Multi-AZ, and public accessibility. Under **Advance**, specify allocated storage, subnet group, Availability Zone, VPC security groups, and AWS Key Management Service (AWS KMS) key.	AWS SysAdmin, DBA
Create the source database endpoint.	Specify the endpoint name, type, source engine (Oracle), server name (Amazon EC2 private DNS name), port, SSL mode, user name, password,	AWS SysAdmin, DBA

Task	Description	Skills required
	SID, VPC (specify the VPC that has the replication instance) , and replication instance. To test the connection, choose **Run Test**, and then create the endpoint. You can also configure the following advanced settings: **maxFileSi ze** and **numberDataTypeScal e**.	
Connect AWS DMS to Amazon RDS for PostgreSQL.	Create a migration security group for connections across VPCs.	AWS SysAdmin, DBA
Create the target database endpoint.	Specify the endpoint name, type, source engine (PostgreS QL), server name (Amazon RDS endpoint), port, SSL mode, user name, password, database name, VPC (specify the VPC that has the replicati on instance), and replicati on instance. To test the connection, choose **Run Test**, and then create the endpoint. You can also configure the following advanced settings: **maxFileSize** and **numberDat aTypeScale**.	AWS SysAdmin, DBA

Task	Description	Skills required
Create the AWS DMS replicati on task.	Specify the task name, replication instance, source and target endpoints, and replication instance. For migration type, choose **Migrate existing data and replicate ongoing changes**. Clear the **Start task on create** checkbox.	AWS SysAdmin, DBA
Configure the AWS DMS replication task settings.	For target table preparation mode, choose **Do nothing**. Stop task after full load completes (to create primary keys). Specify limited or full LOB mode, and activate control tables. Optionally, you can configure the **CommitRat e** advance setting.	DBA
Configure table mappings.	In the **Table mappings** section, create an **Include** rule for all tables in all schemas included in the migration, and then create an **Exclude** rule. Add three transformation rules to convert the schema, table, and column names to lowercase, and add any other rules needed for this specific migration.	DBA

Task	Description	Skills required
Start the task.	Start the replication task. Make sure that the full load is running. Run **ALTER SYSTEM SWITCH LOGFILE** on the primary Oracle database to kick-start the task.	DBA
Run the mid-migration scripts from AWS SCT.	In Amazon RDS for PostgreSQL, run the following scripts generated by AWS SCT: • create_index.sql • create_constraint.sql	DBA
Restart the task to continue change data capture (CDC).	Run **VACUUM** on the Amazon RDS for PostgreSQL DB instance, and restart the AWS DMS task to apply cached CDC changes.	DBA

Cut over to the PostgreSQL database

Task	Description	Skills required
Review the AWS DMS logs and validation tables for any errors.	Check and fix any replication or validation errors.	DBA
Stop all Oracle dependencies.	Stop all Oracle dependencies, shut down listeners on the Oracle database, and run **ALTER SYSTEM SWITCH LOGFILE**. Stop the AWS DMS task when it shows no activity.	DBA

Task	Description	Skills required
Run the post-migration scripts from AWS SCT.	In Amazon RDS for PostgreSQL, run the following scripts generated by AWS SCT: • create_foreign_key_constraint.sql • create_triggers.sql	DBA
Complete additional Amazon RDS for PostgreSQL steps.	Increment sequences to match Oracle if needed, run **VACUUM** and **ANALYZE**, and take a snapshot for compliance.	DBA
Open the connections to Amazon RDS for PostgreSQL.	Remove the AWS DMS security groups from Amazon RDS for PostgreSQL, add production security groups, and point your applications to the new database.	DBA
Clean up AWS DMS objects.	Remove the endpoints, replication tasks, replication instances, and the EC2 instance.	SysAdmin, DBA

Related resources

- [AWS DMS documentation]
- [AWS SCT documentation]
- [Amazon RDS for PostgreSQL pricing]

Migrate from Oracle Database to Amazon RDS for PostgreSQL by using Oracle GoldenGate

Dhairya Jindani, Sindhusha Paturu, and Rajeshkumar Sabankar, Amazon Web Services

Summary

This pattern shows how to migrate an Oracle database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL by using Oracle Cloud Infrastructure (OCI) GoldenGate.

By using Oracle GoldenGate, you can replicate data between your source database and one or more destination databases with minimal downtime.

> **Note**
>
> The source Oracle database can be either on-premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance. You can use a similar procedure when using on-premises replication tools.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Oracle GoldenGate license
- Java Database Connectivity (JDBC) driver to connect to the PostgreSQL database
- Schema and tables created with the AWS Schema Conversion Tool (AWS SCT) on the target Amazon RDS for PostgreSQL database

Limitations

- Oracle GoldenGate can replicate existing table data (initial load) and ongoing changes (change data capture) only

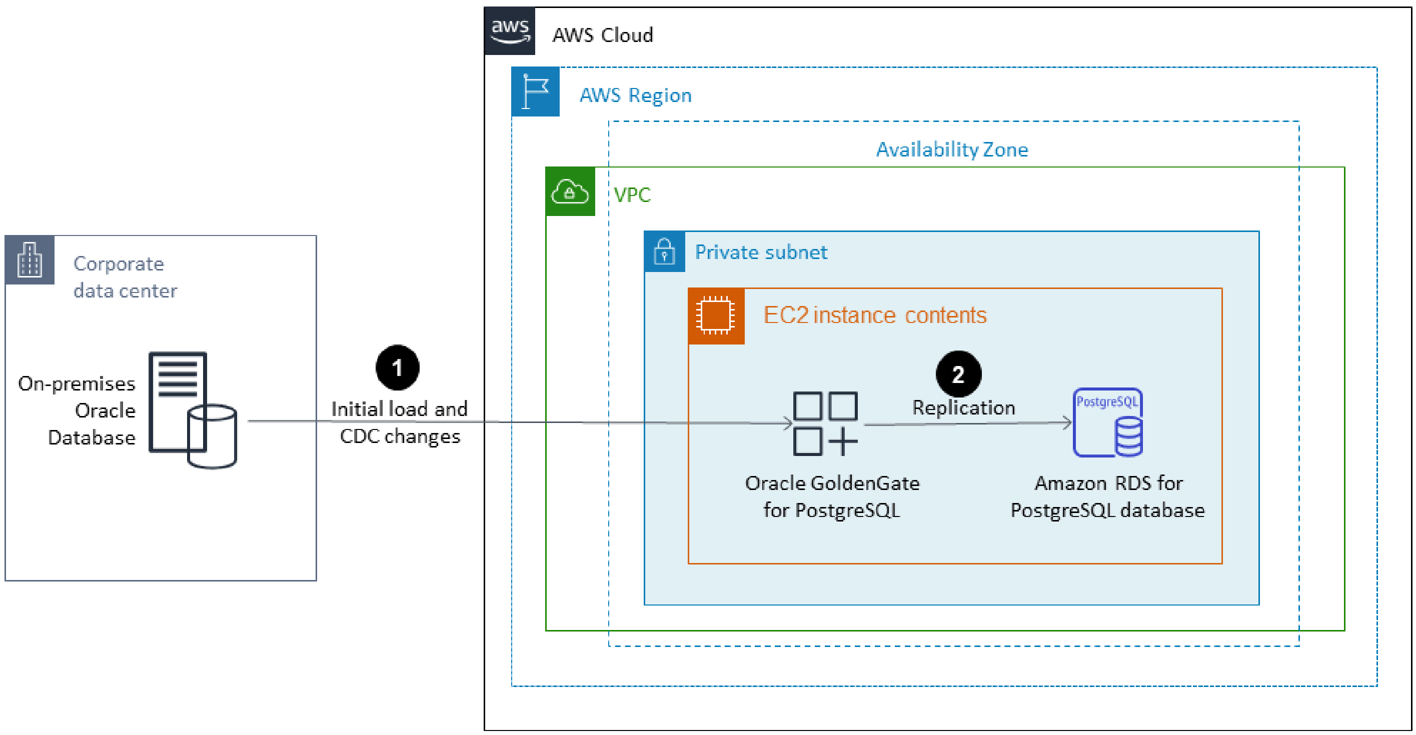
Product versions

- Oracle Database Enterprise Edition 10g or newer versions

- Oracle GoldenGate 12.2.0.1.1 for Oracle or newer versions
- Oracle GoldenGate 12.2.0.1.1 for PostgreSQL or newer versions

Architecture

The following diagram shows an example workflow for migrating an Oracle database to Amazon RDS for PostgreSQL by using Oracle GoldenGate:



The diagram shows the following workflow:

1. The Oracle GoldenGate [Extract process](#) runs against the source database to extract data.
2. The Oracle GoldenGate [Replicat process](#) delivers the extracted data to the target Amazon RDS for PostgreSQL database.

Tools

- Oracle GoldenGate helps you design, run, orchestrate, and monitor your data replication and stream data processing solutions in the Oracle Cloud Infrastructure.
- Amazon Relational Database Service (Amazon RDS) for PostgreSQL helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.

Epics

Download and install Oracle GoldenGate

Task	Description	Skills required
Download Oracle GoldenGat e.	Download the following versions of Oracle GoldenGat e: • Oracle GoldenGat e 12.2.0.1.1 for Oracle or a newer version • Oracle GoldenGat e 12.2.0.1.1 for PostgreSQ L or a newer version To download the software, see Oracle GoldenGate Downloads on the Oracle website.	DBA
Install Oracle GoldenGate for Oracle on the source Oracle Database server.	For instructions, see the Oracle GoldenGate documentation.	DBA
Install Oracle GoldenGate for PostgreSQL database on the Amazon EC2 instance.	For instructions, see the Oracle GoldenGate documentation.	DBA

Configure Oracle GoldenGate on the source and target databases

Task	Description	Skills required
Set up Oracle GoldenGate for Oracle Database on the source database.	For instructions, see the Oracle GoldenGate documentation.	DBA

Task	Description	Skills required
	Make sure that you configure the following: • Supplemental logging • Oracle GoldenGate users • Any required grants and permissions • Parameter files • Manager process • Directory • **GLOBALS** files • Oracle Wallet	
Set up Oracle GoldenGate for PostgreSQL on the target database.	For instructions, see [Part VI Using Oracle GoldenGate for PostgreSQL] on the Oracle website. Make sure that you configure the following: • Manager process • **GLOBALS** files • Oracle Wallet	DBA

Configure the data capture

Task	Description	Skills required
Set up the Extract process in the source database.	In the source Oracle Database, create an extract file to extract data.	DBA

Task	Description	Skills required
	For instructions, see ADD EXTRACT in the Oracle documentation. **Note** The extract file includes the creation of the extract parameter file and trail file directory.	
Set up a data pump to transfer the trail file from the source to the target database.	Create an EXTRACT parameter file and trail file directory by following the instructions in PARFILE in *Database Utilities* on the Oracle website. For more information, see What is a Trail? in *Fusion Middleware Understanding Oracle GoldenGate* on the Oracle website.	DBA

Task	Description	Skills required
Set up replication on the Amazon EC2 instance.	Create a replication parameter file and trail file directory. For more information about creating replication parameter files, see section [3.5 Validating a parameter file]() in the Oracle Database documentation. For more information about creating a trail file directory , see [Creating a trail]() in the Oracle Cloud documentation. **Important** Make sure that you add a checkpoint table entry in the GLOBALS file at the target. For more information, see [What is a Replicat?]() in *Fusion Middleware Understanding Oracle GoldenGate* on the Oracle website.	DBA

Configure the data replication

Task	Description	Skills required
In the source database, create a parameter file to extract data for the initial load.	Follow the instructions in [Creating a parameter file in GGSCI](#) in the Oracle Cloud documentation. **Important** Make sure that the Manager is running on the target.	DBA
In the target database, create a parameter file to replicate data for the initial load.	Follow the instructions in [Creating a parameter file in GGSCI](#) in the Oracle Cloud documentation. **Important** Make sure that you add and start the Replicat process.	DBA

Cut over to the Amazon RDS for PostgreSQL database

Task	Description	Skills required
Stop the Replicat process and make sure that the source and target databases are in sync.	Compare row counts between the source and target databases to make sure that the data replication was successful.	DBA

Task	Description	Skills required
Configure data definition language (DDL) support.	Run the DDL script for creating triggers, sequence, synonyms, and referential keys on PostgreSQL. **Note** You can use any standard SQL client application to connect to a database in your DB cluster. For example, you can use pgAdmin to connect to your DB instance.	DBA

Related resources

- Amazon RDS for PostgreSQL (*Amazon RDS User Guide*)
- Amazon EC2 documentation
- Oracle GoldenGate supported processing methods and databases (Oracle documentation)

Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT

Piyush Goyal and Brian motzer, Amazon Web Services

Summary

This pattern provides guidance for migrating Oracle databases to an Amazon Redshift cloud data warehouse in the Amazon Web Services (AWS) Cloud by using AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT). The pattern covers source Oracle databases that are on premises or installed on an Amazon Elastic Compute Cloud (Amazon EC2) instance. It also covers Amazon Relational Database Service (Amazon RDS) for Oracle databases.

Prerequisites and limitations

Prerequisites

- An Oracle database that is running in an on-premises data center or in the AWS Cloud
- An active AWS account
- Familiarity with using an Oracle database as a source for AWS DMS
- Familiarity with using an Amazon Redshift database as a target for AWS DMS
- Knowledge of Amazon RDS, Amazon Redshift, the applicable database technologies, and SQL
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, where AWS SCT is installed

Product versions

- For self-managed Oracle databases, AWS DMS supports all Oracle database editions for versions 10.2 and later (for versions 10.*x*), 11g and up to 12.2, 18c, and 19c. For Amazon RDS for Oracle databases that AWS manages, AWS DMS supports all Oracle database editions for versions 11g (versions 11.2.0.4 and later) and up to 12.2, 18c, and 19c. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Architecture

Source technology stack

One of the following:

- An on-premises Oracle database

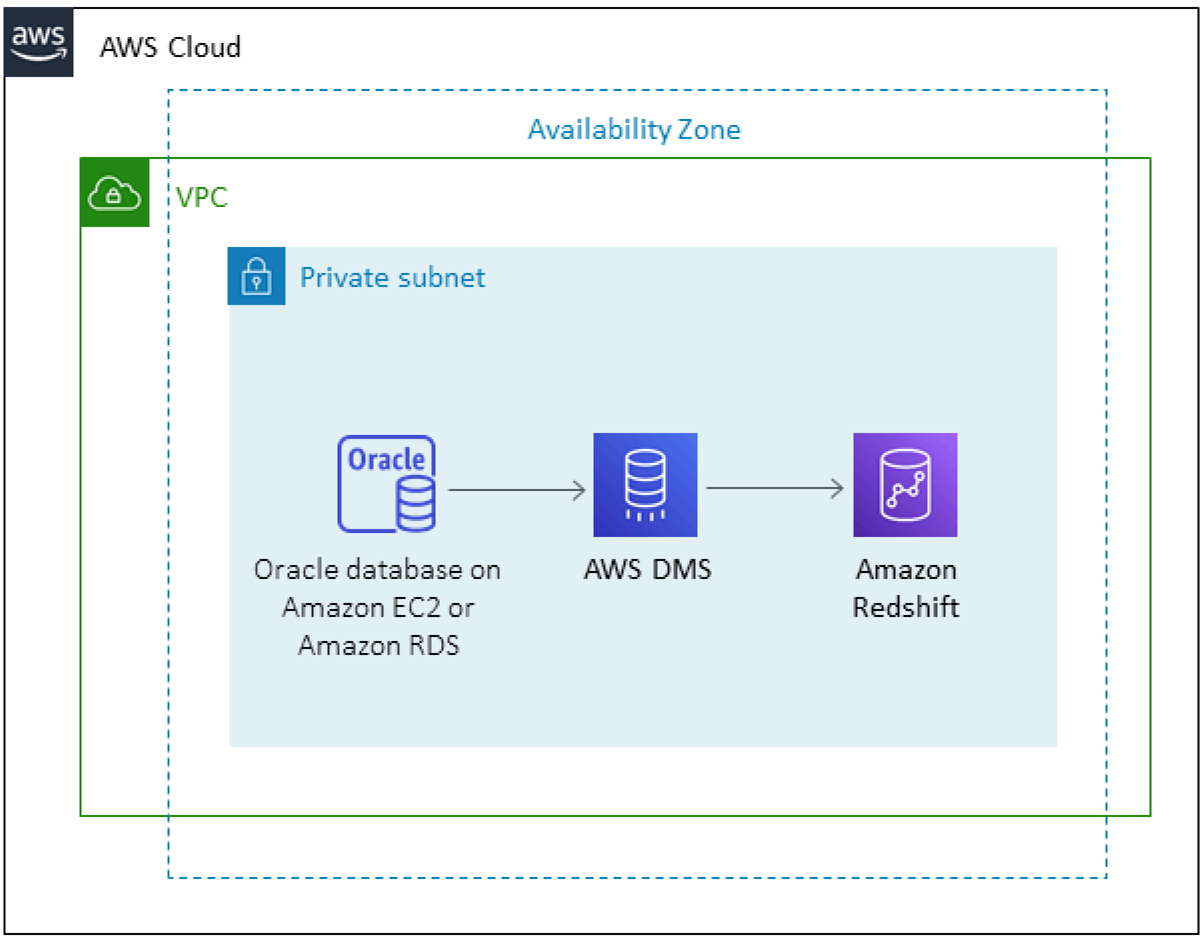
- An Oracle database on an EC2 instance
- An Amazon RDS for Oracle DB instance

Target technology stack

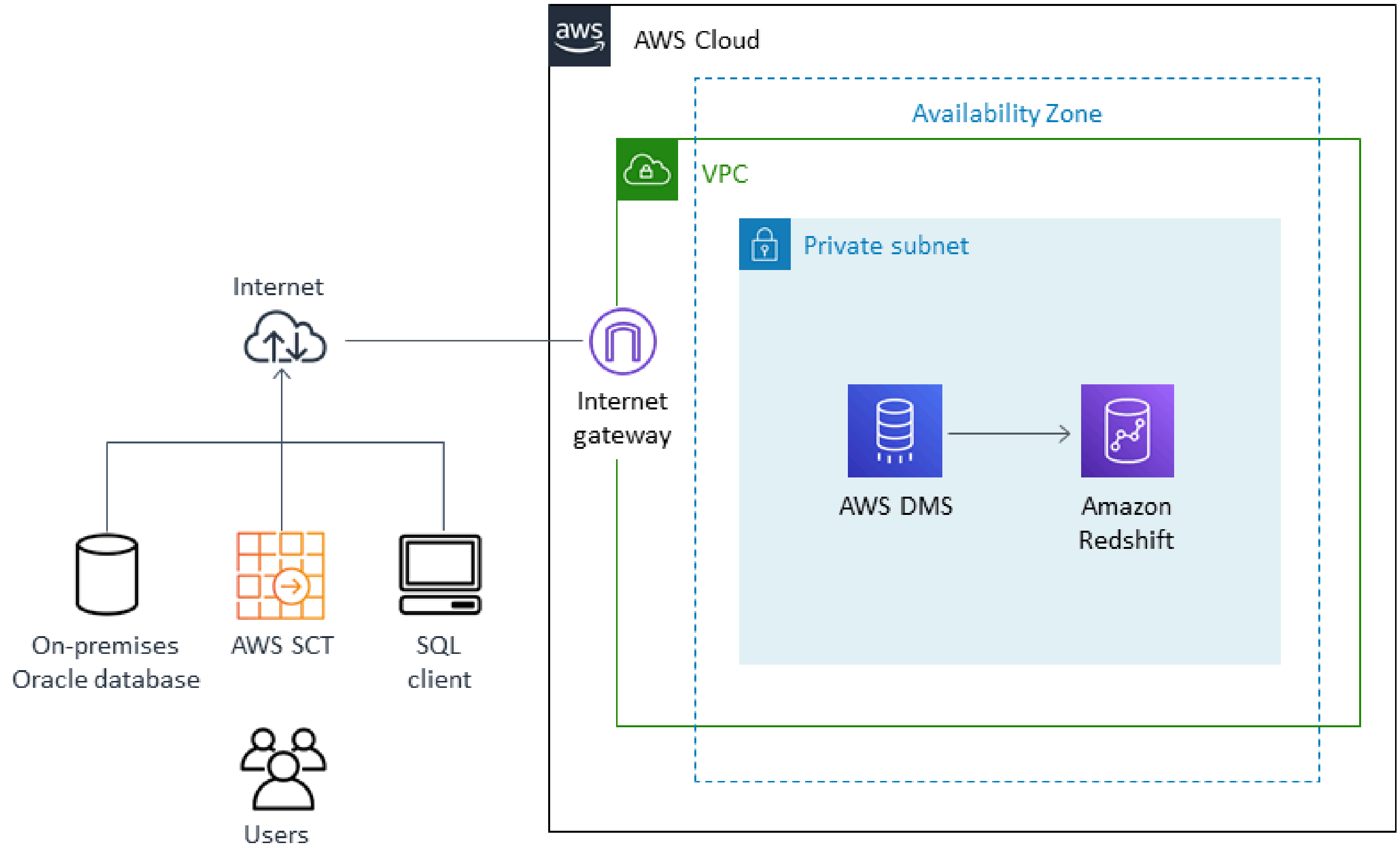
- Amazon Redshift

Target architecture

From an Oracle database running in the AWS Cloud to Amazon Redshift:



From an Oracle database running in an on-premises data center to Amazon Redshift:



Tools

- AWS DMS - AWS Data Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. AWS DMS can migrate your data to and from most widely used commercial and open-source databases.
- AWS SCT - AWS Schema Conversion Tool (AWS SCT) can be used to convert your existing database schema from one database engine to another. It supports various database engines, including Oracle, SQL Server, and PostgresSQL, as sources.

Epics

Prepare for the migration

Task	Description	Skills required
Validate the database versions.	Validate the source and target database versions and make	DBA

Task	Description	Skills required
	sure they are supported by AWS DMS. For informati on about supported Oracle Database versions, see Using an Oracle database as a source for AWS DMS. For information about using Amazon Redshift as a target, see Using an Amazon Redshift database as a target for AWS DMS.	
Create a VPC and security group.	In your AWS account, create a virtual private cloud (VPC), if it doesn't exist. Create a security group for outbound traffic to source and target databases. For more informati on, see the Amazon Virtual Private Cloud (Amazon VPC) documentation.	Systems administrator
Install AWS SCT.	Download and install the latest version of AWS SCT and its corresponding drivers. For more information, see Installing, verifying, and updating the AWS SCT.	DBA
Create a user for the AWS DMS task.	Create an AWS DMS user in the source database and grant it READ privileges. This user will be used by both AWS SCT and AWS DMS.	DBA

Task	Description	Skills required
Test the DB connectivity.	Test the connectivity to the Oracle DB instance.	DBA
Create a new project in AWS SCT.	Open the AWS SCT tool and create a new project.	DBA
Analyze the Oracle schema to be migrated.	Use AWS SCT to analyze the schema to be migrated, and generate a database migration assessment report. For more information, see [Creating a database migration assessment report] in the AWS SCT documentation.	DBA
Review the assessment report.	Review the report for migration feasibility. Some DB objects might require manual conversion. For more information about the report, see [Viewing the assessment report] in the AWS SCT documentation.	DBA

Prepare the target database

Task	Description	Skills required
Create an Amazon Redshift cluster.	Create an Amazon Redshift cluster within the VPC that you created previously. For more information, see [Amazon Redshift clusters]	DBA

Task	Description	Skills required
	in the Amazon Redshift documentation.	
Create database users.	Extract the list of users, roles, and grants from the Oracle source database. Create users in the target Amazon Redshift database and apply the roles from the previous step.	DBA
Evaluate database parameters.	Review the database options, parameters, network files, and database links from the Oracle source database, and evaluate their applicability to the target.	DBA
Apply any relevant settings to the target.	For more information about this step, see [Configuration reference] in the Amazon Redshift documentation.	DBA

Create objects in the target database

Task	Description	Skills required
Create an AWS DMS user in the target database.	Create an AWS DMS user in the target database and grant it read and write privileges. Validate the connectivity from AWS SCT.	DBA
Convert the schema, review the SQL report, and save any errors or warnings.	For more information, see [Converting database schemas]	DBA

Task	Description	Skills required
	using the AWS SCT in the AWS SCT documentation.	
Apply the schema changes to the target database or save them as a .sql file.	For instructions, see Saving and applying your converted schema in the AWS SCT in the AWS SCT documentation.	DBA
Validate the objects in the target database.	Validate the objects that were created in the previous step in the target database. Rewrite or redesign any objects that weren't successfully converted.	DBA
Disable foreign keys and triggers.	Disable any foreign key and triggers. These can cause data loading issues during the full load process when running AWS DMS.	DBA

Migrate data using AWS DMS

Task	Description	Skills required
Create an AWS DMS replication instance.	Sign in to the AWS Management Console, and open the AWS DMS console. In the navigation pane, choose **Replication instances**, **Create replication instance**. For detailed instructions, see step 1 in *Getting started with AWS DMS* in the AWS DMS documentation.	DBA

Task	Description	Skills required
Create source and target endpoints.	Create source and target endpoints, Test the connectio n from the replication instance to both source and target endpoints. For detailed instructions, see step 2 in *Getting started with AWS DMS* in the AWS DMS documenta tion.	DBA
Create a replication task.	Create a replication task and select the appropria te migration method. For detailed instructions, see step 3 in *Getting started with AWS DMS* in the AWS DMS documentation.	DBA
Start the data replication.	Start the replication task and monitor the logs for any errors.	DBA

Migrate your application

Task	Description	Skills required
Create application servers.	Create the new application servers on AWS.	Application owner
Migrate the application code.	Migrate the application code to the new servers.	Application owner
Configure the application server.	Configure the application server for the target database and drivers.	Application owner

Task	Description	Skills required
Optimize the application code.	Optimize the application code for the target engine.	Application owner

Cut over to the target database

Task	Description	Skills required
Validate users.	In the target Amazon Redshift database, validate users and grant them roles and privileges.	DBA
Validate that the application is locked.	Make sure that the application is locked, to prevent further changes.	Application owner
Validate the data.	Validate the data in the target Amazon Redshift database.	DBA
Enable foreign keys and triggers.	Enable foreign keys and triggers in the target Amazon Redshift database.	DBA
Connect to the new database.	Configure the application to connect to the new Amazon Redshift database.	Application owner
Perform final checks.	Perform a final, comprehensive system check before going live.	DBA, Application owner
Go live.	Go live with the target Amazon Redshift database.	DBA

Close the migration project

Task	Description	Skills required
Shut down temporary AWS resources.	Shut down temporary AWS resources such as the AWS DMS replication instance and the EC2 instance used for AWS SCT.	DBA, Systems administrator
Review documents.	Review and validate the migration project documents .	DBA, Systems administrator
Gather metrics.	Collect information about the migration project, such as the time to migrate, the percentage of manual versus tool tasks, and total cost savings.	DBA, Systems administrator
Close out the project.	Close out the project and provide feedback.	DBA, Systems administrator

Related resources

References

- AWS DMS user guide
- AWS SCT user guide
- Amazon Redshift getting started guide

Tutorials and videos

- Dive Deep into AWS SCT and AWS DMS (presentation from AWS re:Invent 2019)
- Getting Started with AWS Database Migration Service

Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT

Senthil Ramasamy, Amazon Web Services

Summary

This pattern describes how to migrate an Oracle database to Amazon Aurora PostgreSQL-Compatible Edition by using AWS Data Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT).

The pattern covers source Oracle databases that are on premises, Oracle databases that are installed on Amazon Elastic Compute Cloud (Amazon EC2) instances, and Amazon Relational Database Service (Amazon RDS) for Oracle databases. The pattern converts these databases to Aurora PostgreSQL-Compatible.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An Oracle database in an on-premises data center or in the AWS Cloud.
- SQL clients installed either on a local machine or on an EC2 instance.
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, installed on either a local machine or an EC2 instance where AWS SCT is installed.

Limitations

- Database size limit: 128 TB
- If the source database supports a commercial off-the-shelf (COTS) application or is vendor-specific, you might not be able to convert it to another database engine. Before using this pattern, confirm that the application supports Aurora PostgreSQL-Compatible.

Product versions

- For self-managed Oracle databases, AWS DMS supports all Oracle database editions for versions 10.2 and later (for versions 10.x), 11g, and up to 12.2, 18c, and 19c. For the latest list of supported Oracle database versions (both self-managed and Amazon RDS for Oracle), see Using an Oracle database as a source for AWS DMS and Using a PostgreSQL database as a target for AWS DMS.

- We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Oracle database versions supported by AWS SCT, see the AWS SCT documentation.
- Aurora supports the PostgreSQL versions listed in Amazon Aurora PostgreSQL releases and engine versions.

Architecture

Source technology stack

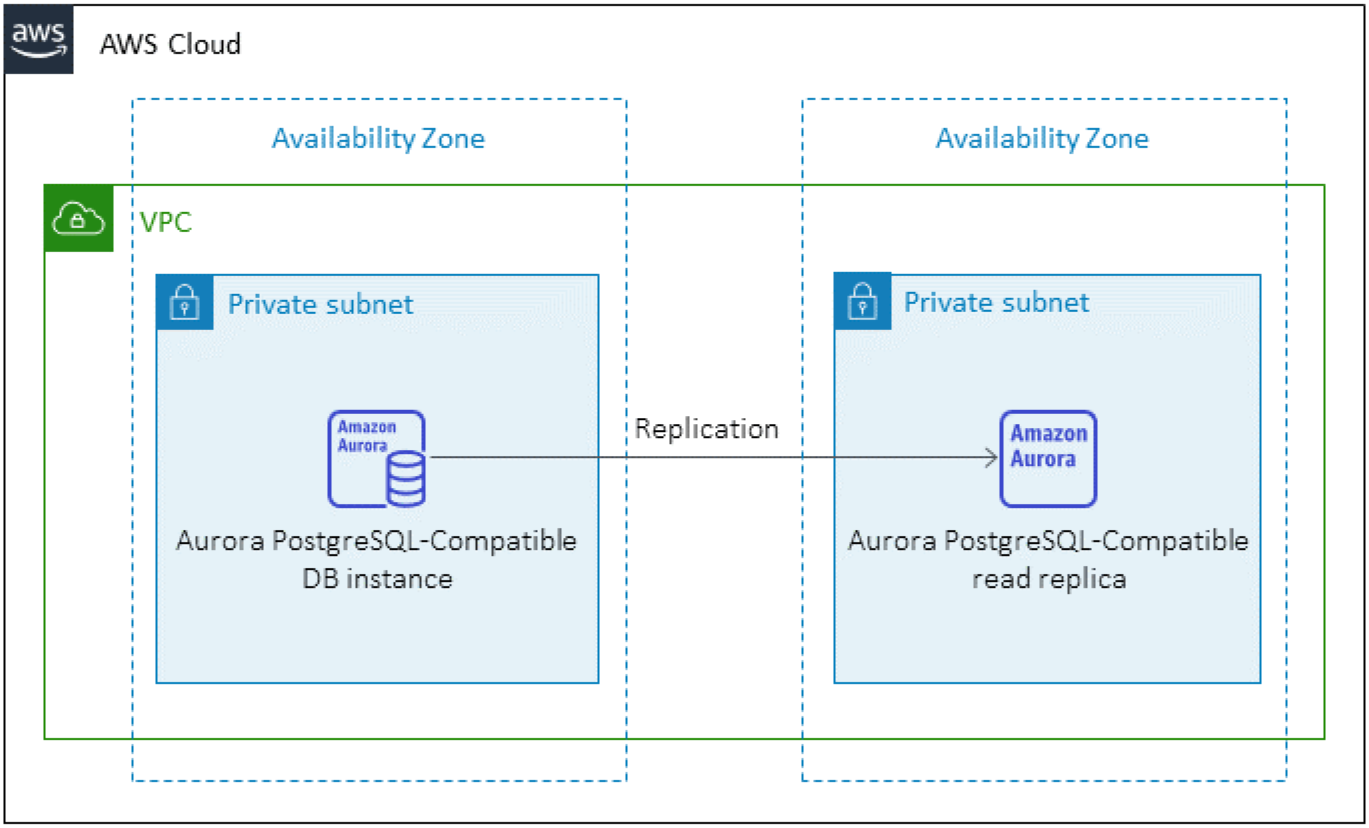
One of the following:

- An on-premises Oracle database
- An Oracle database on an EC2 instance
- An Amazon RDS for Oracle DB instance

Target technology stack

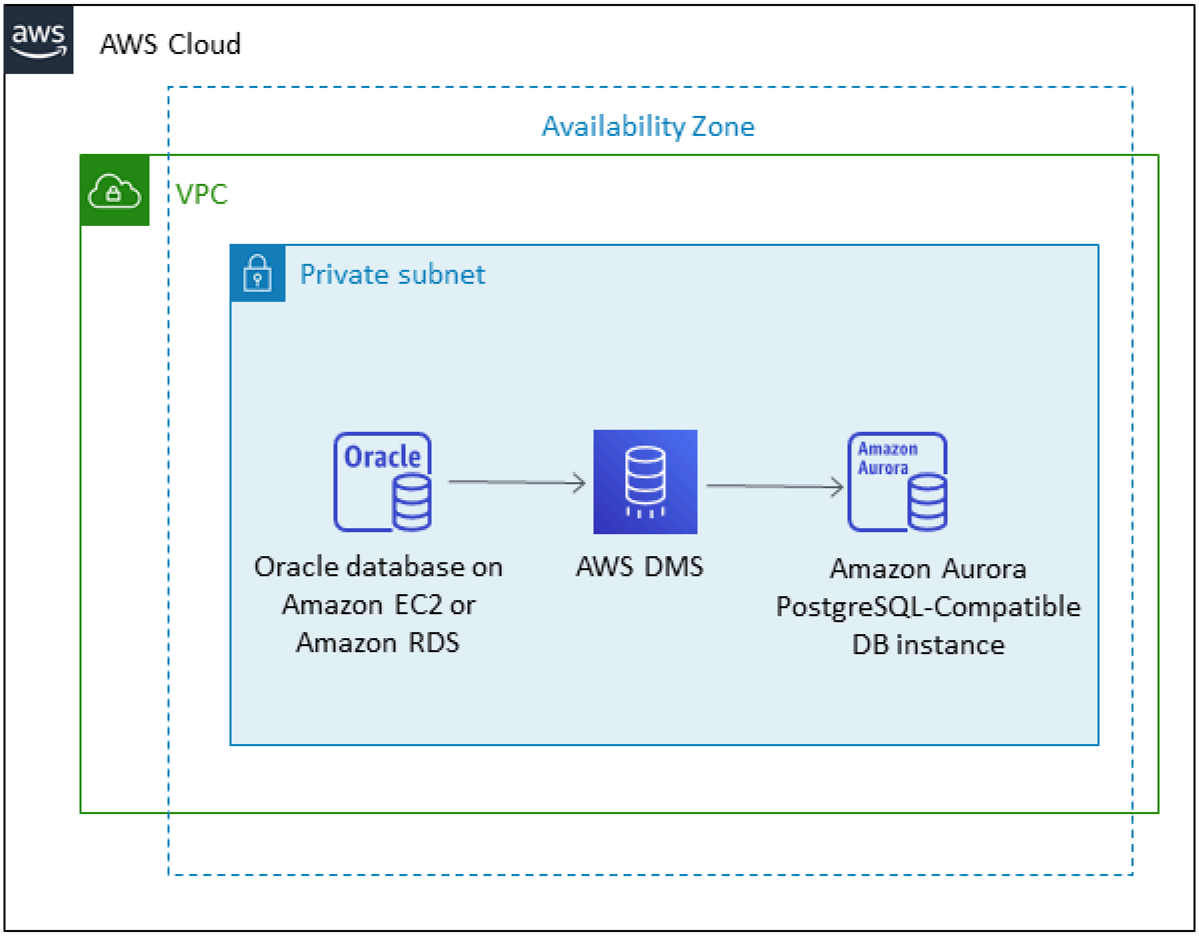
- Aurora PostgreSQL-Compatible

Target architecture

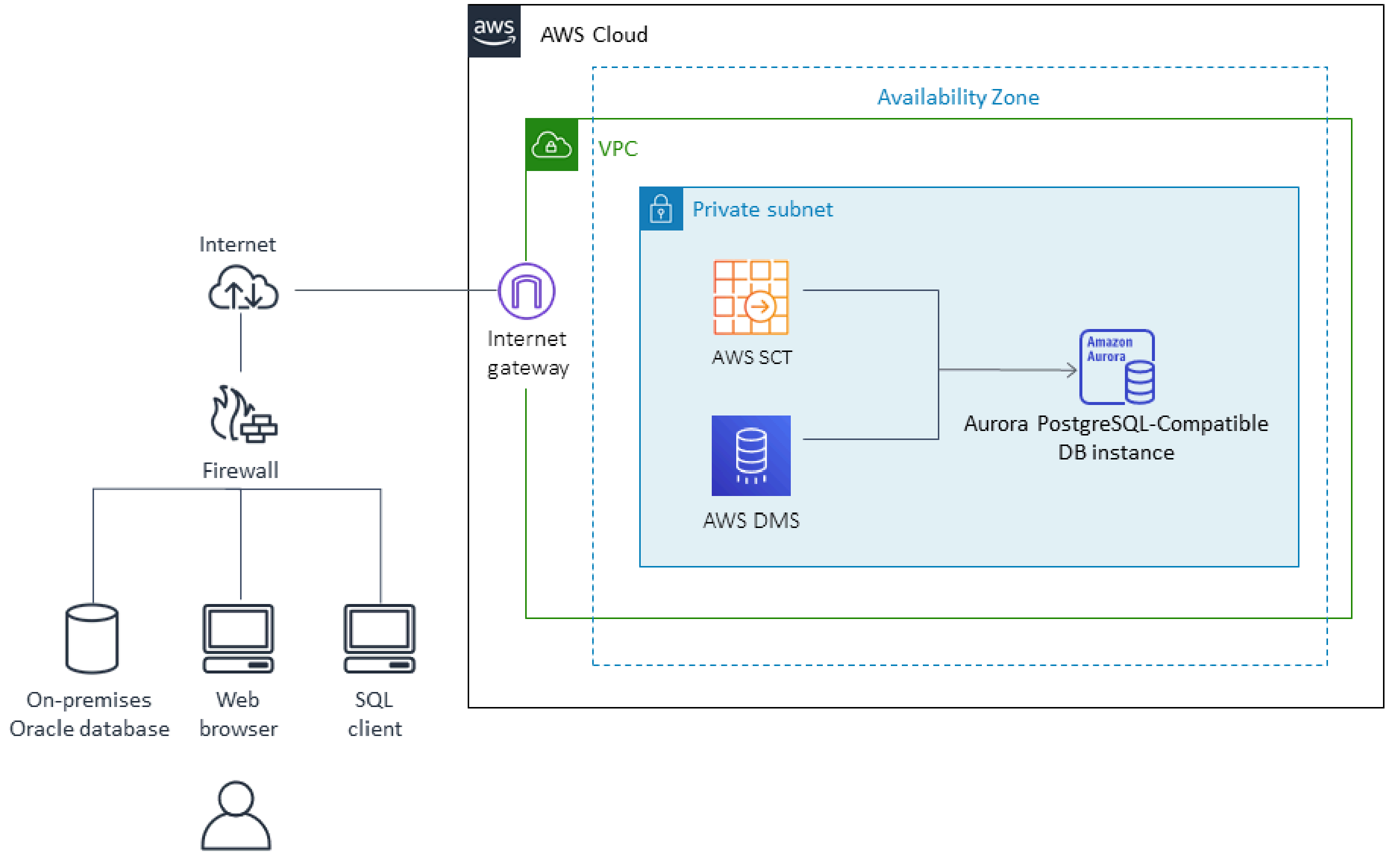


Data migration architecture

- From an Oracle database running in the AWS Cloud



- From an Oracle database running in an on-premises data center



Tools

- [AWS Database Migration Service (AWS DMS)](#) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- [AWS Schema Conversion Tool (AWS SCT)](#) supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format compatible with the target database.

Epics

Prepare for the migration

Task	Description	Skills required
Prepare the source database.	To prepare the source database, see [Using Oracle Database as a source for](#)	DBA

Task	Description	Skills required
	AWS SCT in the AWS SCT documentation.	
Create an EC2 instance for AWS SCT.	Create and configure an EC2 instance for AWS SCT, if required.	DBA
Download AWS SCT.	Download the latest version of AWS SCT and associated drivers. For more information, see Installing, verifying, and updating AWS SCT in the AWS SCT documentation.	DBA
Add users and permissions.	Add and validate the prerequisite users and permissions in the source database.	DBA
Create an AWS SCT project.	Create an AWS SCT project for the workload, and connect to the source database. For instructions, see Creating an AWS SCT project and Adding database servers in the AWS SCT documentation.	DBA

Task	Description	Skills required
Evaluate feasibility.	Generate an assessment report, which summarizes action items for schemas that can't be converted automatically and provides estimates for manual conversion efforts. For more information, see [Creating and reviewing the database migration assessment report]() in the AWS SCT documentation.	DBA

Prepare the target database

Task	Description	Skills required
Create a target Amazon RDS DB instance.	Create a target Amazon RDS DB instance, using Amazon Aurora as the database engine. For instructions, see [Creating an Amazon RDS DB instance]() in the Amazon RDS documentation.	DBA
Extract users, roles, and permissions.	Extract the list of users, roles, and permissions from the source database.	DBA
Map users.	Map the existing database users to the new database users.	App owner
Create users.	Create users in the target database.	DBA, App owner

Task	Description	Skills required
Apply roles.	Apply roles from the previous step to the target database.	DBA
Check options, parameters, network files, and database links.	Review the source database for options, parameters, network files, and database links, and then evaluate their applicability to the target database.	DBA
Apply settings.	Apply any relevant settings to the target database.	DBA

Transfer objects

Task	Description	Skills required
Configure AWS SCT connectiv ity.	Configure AWS SCT connectiv ity to the target database.	DBA
Convert the schema using AWS SCT.	AWS SCT automatically converts the source database schema and most of the custom code to a format that is compatible with the target database. Any code that the tool cannot convert automatically is clearly marked so that you can convert it manually.	DBA
Review the report.	Review the generated SQL report and save any errors and warnings.	DBA

Task	Description	Skills required
Apply automated schema changes.	Apply automated schema changes to the target database or save them as a .sql file.	DBA
Validate objects.	Validate that AWS SCT created the objects on the target.	DBA
Handle items that weren't converted.	Manually rewrite, reject, or redesign any items that failed to convert automatically.	DBA, App owner
Apply role and user permissio ns.	Apply the generated role and user permissions and review any exceptions.	DBA

Migrate the data

Task	Description	Skills required
Determine the method.	Determine the method for migrating data.	DBA
Create a replication instance.	Create a replication instance from the AWS DMS console. For more information, see [Working with an AWS DMS replication instance](#) in the AWS DMS documentation.	DBA
Create the source and target endpoints.	To create endpoints, follow the instructions in [Creating source and target endpoints](#)	DBA

Task	Description	Skills required
	in the AWS DMS documentation.	
Create a replication task.	To create a task, see Working with AWS DMS tasks in the AWS DMS documentation.	DBA
Start the replication task and monitor the logs.	For more information about this step, see Monitoring AWS DMS tasks in the AWS DMS documentation.	DBA

Migrate the application

Task	Description	Skills required
Analyze and convert SQL items in the application code.	Use AWS SCT to analyze and convert the SQL items in the application code. When you convert your database schema from one engine to another, you also need to update the SQL code in your applications to interact with the new database engine instead of the old one. You can view, analyze, edit, and save the converted SQL code.	App owner
Create application servers.	Create the new application servers on AWS.	App owner
Migrate the application code.	Migrate the application code to the new servers.	App owner

Task	Description	Skills required
Configure the application servers.	Configure the applicati on servers for the target database and drivers.	App owner
Fix code.	Fix any code that's specific to the source database engine in your application.	App owner
Optimize code.	Optimize your application code for the target database engine.	App owner

Cut over

Task	Description	Skills required
Cut over to the target database.	Perform the cutover to the new database.	DBA
Lock the application.	Lock the application from any further changes.	App owner
Validate changes.	Validate that all changes were propagated to the target database.	DBA
Redirect to the target database.	Point the new applicati on servers to the target database.	App owner
Check everything.	Perform a final, comprehen sive system check.	App owner
Go live.	Complete final cutover tasks.	App owner

Close the project

Task	Description	Skills required
Shut down temporary resources.	Shut down the temporary AWS resources such as the AWS DMS replication instance and the EC2 instance used for AWS SCT.	DBA, App owner
Update feedback.	Update feedback on the AWS DMS process for internal teams.	DBA, App owner
Revise process and templates.	Revise the AWS DMS process and improve the template if necessary.	DBA, App owner
Validate documents.	Review and validate the project documents.	DBA, App owner
Gather metrics.	Gather metrics to evaluate the time to migrate, percent of manual versus tool cost savings, and so on.	DBA, App owner
Close the project.	Close the migration project and provide feedback to stakeholders.	DBA, App owner

Related resources

References

- Using an Oracle Database as a Source for AWS DMS
- Using a PostgreSQL Database as a Target for AWS Database Migration Service
- Oracle Database 11g/12c to Amazon Aurora with PostgreSQL Compatibility (9.6.x) Migration Playbook

- Oracle Database 19c to Amazon Aurora with PostgreSQL Compatibility (12.4) Migration Playbook
- Migrating an Amazon RDS for Oracle database to Amazon Aurora PostgreSQL-Compatible Edition
- AWS Data Migration Service
- AWS Schema Conversion Tool
- Migrate from Oracle to Amazon Aurora
- Amazon RDS pricing

Tutorials and videos

- Database Migration Step-by-Step Walkthroughs
- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS Data Migration Service (video)
- Migrating an Oracle database to PostgreSQL (video)

Additional information

.

Migrate data from an on-premises Oracle database to Aurora PostgreSQL

Michelle Deng and Shunan Xiang, Amazon Web Services

Summary

This pattern provides guidance for data migration from an on-premises Oracle database to Amazon Aurora PostgreSQL-Compatible Edition. It targets an online data migration strategy with a minimal amount of downtime for multi-terabyte Oracle databases that contain large tables with high data manipulation language (DML) activities. An Oracle Active Data Guard standby database is used as the source to offload data migration from the primary database. The replication from the Oracle primary database to standby can be suspended during the full load to avoid ORA-01555 errors.

Table columns in primary keys (PKs) or foreign keys (FKs), with data type NUMBER, are commonly used to store integers in Oracle. We recommend that you convert these to INT or BIGINT in PostgreSQL for better performance. You can use the AWS Schema Conversion Tool (AWS SCT) to change the default data type mapping for PK and FK columns. (For more information, see the AWS blog post [Convert the NUMBER data type from Oracle to PostgreSQL](#).) The data migration in this pattern uses AWS Database Migration Service (AWS DMS) for both full load and change data capture (CDC).

You can also use this pattern to migrate an on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for PostgreSQL, or an Oracle database that's hosted on Amazon Elastic Compute Cloud (Amazon EC2) to either Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Oracle source database in an on-premises data center with Active Data Guard standby configured
- AWS Direct Connect configured between the on-premises data center and the AWS Cloud
- Familiarity with using an Oracle database as a source for AWS DMS
- Familiarity with using a PostgreSQL database as a target for AWS DMS

Limitations

- Amazon Aurora database clusters can be created with up to 128 TiB of storage. Amazon RDS for PostgreSQL database instances can be created with up to 64 TiB of storage. For the latest storage information, see Amazon Aurora storage and reliability and Amazon RDS DB instance storage in the AWS documentation.

Product versions

- AWS DMS supports all Oracle database editions for versions 10.2 and later (for versions 10.x), 11g and up to 12.2, 18c, and 19c. For the latest list of supported versions, see Using an Oracle Database as a Source for AWS DMS in the AWS documentation.

Architecture

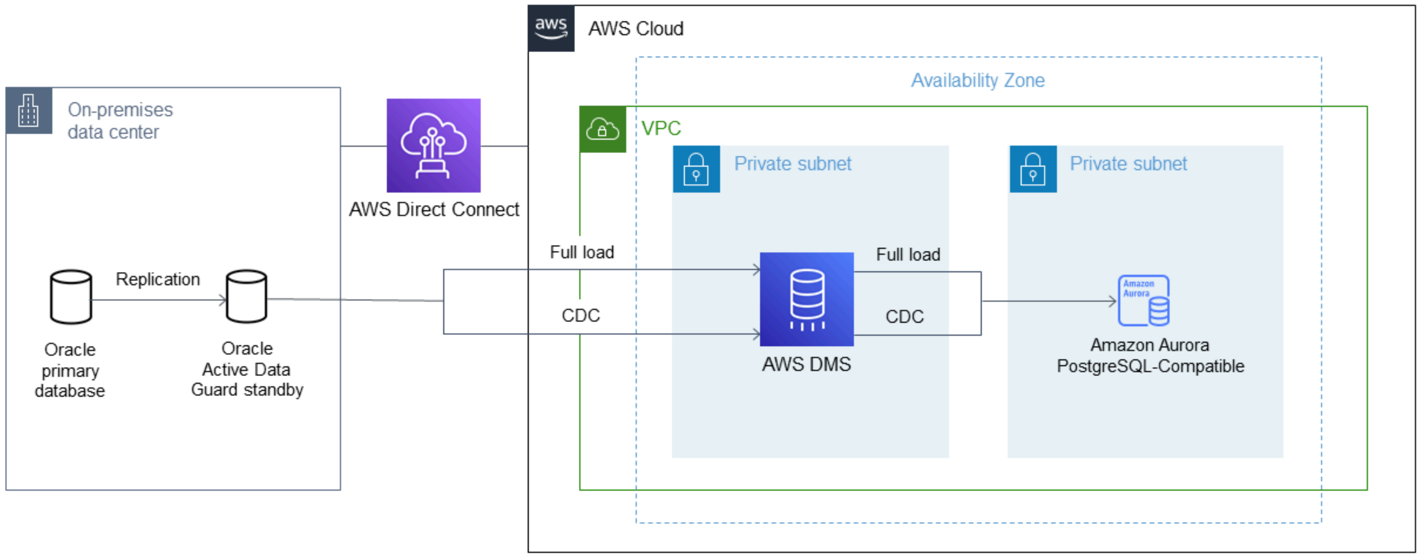
Source technology stack

- On-premises Oracle databases with Oracle Active Data Guard standby configured

Target technology stack

- Aurora PostgreSQL-Compatible

Data migration architecture



Tools

- **AWS DMS** - AWS Database Migration Service (AWS DMS) supports several source and target databases. See Using an Oracle Database as a Source for AWS DMS in the AWS DMS documentation for a list of supported Oracle source and target database versions and editions. If the source database is not supported by AWS DMS, you must select another method for migrating the data in Phase 6 (in the *Epics* section). **Important note:** Because this is a heterogeneous migration, you must first check to see whether the database supports a commercial off-the-shelf (COTS) application. If the application is COTS, consult the vendor to confirm that Aurora PostgreSQL-Compatible is supported before proceeding. For more information, see AWS DMS Step-by-Step Migration Walkthroughs in the AWS documentation.
- **AWS SCT** - The AWS Schema Conversion Tool (AWS SCT) facilitates heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. The custom code that the tool converts includes views, stored procedures, and functions. Any code that the tool cannot convert automatically is clearly marked so that you can convert it yourself.

Epics

Plan the migration

Task	Description	Skills required
Validate the source and target database versions.		DBA
Install AWS SCT and drivers.		DBA
Add and validate the AWS SCT prerequisite users and grants-source database.		DBA
Create an AWS SCT project for the workload, and connect to the source database.		DBA

Task	Description	Skills required
Generate an assessment report and evaluate feasibili ty.		DBA, App owner

Prepare the target database

Task	Description	Skills required
Create an Aurora PostgreSQL-Compatible target database.		DBA
Extract users, roles, and grants list from the source database.		DBA
Map the existing database users to the new database users.		App owner
Create users in the target database.		DBA
Apply roles from the previous step to the target Aurora PostgreSQL-Compatible database.		DBA
Review database options, parameters, network files, and database links from the source database, and evaluate their applicability to the target database.		DBA, App owner

Task	Description	Skills required
Apply any relevant settings to the target database.		DBA

Prepare for database object code conversion

Task	Description	Skills required
Configure AWS SCT connectiv ity to the target database.		DBA
Convert the schema in AWS SCT, and save the converted code as a .sql file.		DBA, App owner
Manually convert any database objects that failed to convert automatically.		DBA, App owner
Optimize the database code conversion.		DBA, App owner
Separate the .sql file into multiple .sql files based on the object type.		DBA, App owner
Validate the SQL scripts in the target database.		DBA, App owner

Prepare for data migration

Task	Description	Skills required
Create an AWS DMS replicati on instance.		DBA

Task	Description	Skills required
Create the source and target endpoints.	If the data type of the PKs and FKs is converted from NUMBER in Oracle to BIGINT in PostgreSQL, consider specifying the connection attribute `numberDataTypeScale=-2` when you create the source endpoint.	DBA

Migrate data – full load

Task	Description	Skills required
Create the schema and tables in the target database.		DBA
Create AWS DMS full-load tasks by either grouping tables or splitting a big table based on the table size.		DBA
Stop the applications on the source Oracle databases for a short period.		App owner
Verify that the Oracle standby database is synchronous with the primary database, and stop the replication from the primary database to the standby database.		DBA, App owner
Start applications on the source Oracle database.		App owner

Task	Description	Skills required
Start the AWS DMS full-load tasks in parallel from the Oracle standby database to the Aurora PostgreSQL-Compatible database.		DBA
Create PKs and secondary indexes after the full load is complete.		DBA
Validate the data.		DBA

Migrate data – CDC

Task	Description	Skills required
Create AWS DMS ongoing replication tasks by specifying a custom CDC start time or system change number (SCN) when the Oracle standby was synchronized with the primary database, and before the applications were restarted in the previous task.		DBA
Start AWS DMS tasks in parallel to replicate ongoing changes from the Oracle standby database to the Aurora PostgreSQL-Compatible database.		DBA

Task	Description	Skills required
Re-establish the replicati on from the Oracle primary database to the standby database.		DBA
Monitor the logs and stop the applications on the Oracle database when the target Aurora PostgreSQ L-Compatible database is almost synchronous with the source Oracle database.		DBA, App owner
Stop the AWS DMS tasks when the target is fully synchronized with the source Oracle database.		DBA
Create FKs and validate the data in the target database.		DBA
Create functions, views, triggers, sequences, and other object types in the target database.		DBA
Apply role grants in the target database.		DBA

Migrate the application

Task	Description	Skills required
Use AWS SCT to analyze and convert the SQL statements inside the application code.		App owner
Create new application servers on AWS.		App owner
Migrate the application code to the new servers.		App owner
Configure the application server for the target database and drivers.		App owner
Fix any code that's specific to the source database engine in the application.		App owner
Optimize the application code for the target database.		App owner

Cut over

Task	Description	Skills required
Point the new application server to the target database.		DBA, App owner
Perform sanity checks.		DBA, App owner
Go live.		DBA, App owner

Close the project

Task	Description	Skills required
Shut down temporary AWS resources.		DBA, Systems administrator
Review and validate the project documents.		DBA, App owner
Gather metrics for time to migrate, percentage of manual versus tool use, cost savings, and similar data.		DBA, App owner
Close out the project and provide feedback.		DBA, App owner

Related resources

References

- Oracle Database to Aurora PostgreSQL-Compatible: Migration Playbook
- Migrating an Amazon RDS for Oracle Database to Amazon Aurora MySQL
- AWS DMS website
- AWS DMS documentation
- AWS SCT website
- AWS SCT documentation
- Migrate from Oracle to Amazon Aurora

Tutorials

- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS Database Migration Service Step-by-Step Walkthroughs

Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS

Amit Kumar, Amazon Web Services

Summary

This pattern provides guidance for migrating an SAP Adaptive Server Enterprise (ASE) database to an Amazon Relational Database Service (Amazon RDS) DB instance that's running Microsoft SQL Server. The source database can be located in an on-premises data center or on an Amazon Elastic Compute Cloud (Amazon EC2) instance. The pattern uses AWS Database Migration Service (AWS DMS) to migrate data and (optionally) computer-aided software engineering (CASE) tools to convert the database schema.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An SAP ASE database in an on-premises data center or on an EC2 instance
- A target Amazon RDS for SQL Server database that's up and running

Limitations

- Database size limit: 64 TB

Product versions

- SAP ASE version 15.7 or 16.x only. For the latest information, see Using an SAP Database as a Source for AWS DMS.
- For Amazon RDS target databases, AWS DMS supports Microsoft SQL Server versions on Amazon RDS for the Enterprise, Standard, Web, and Express editions. For the latest information about supported versions, see the AWS DMS documentation. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.

Architecture

Source technology stack

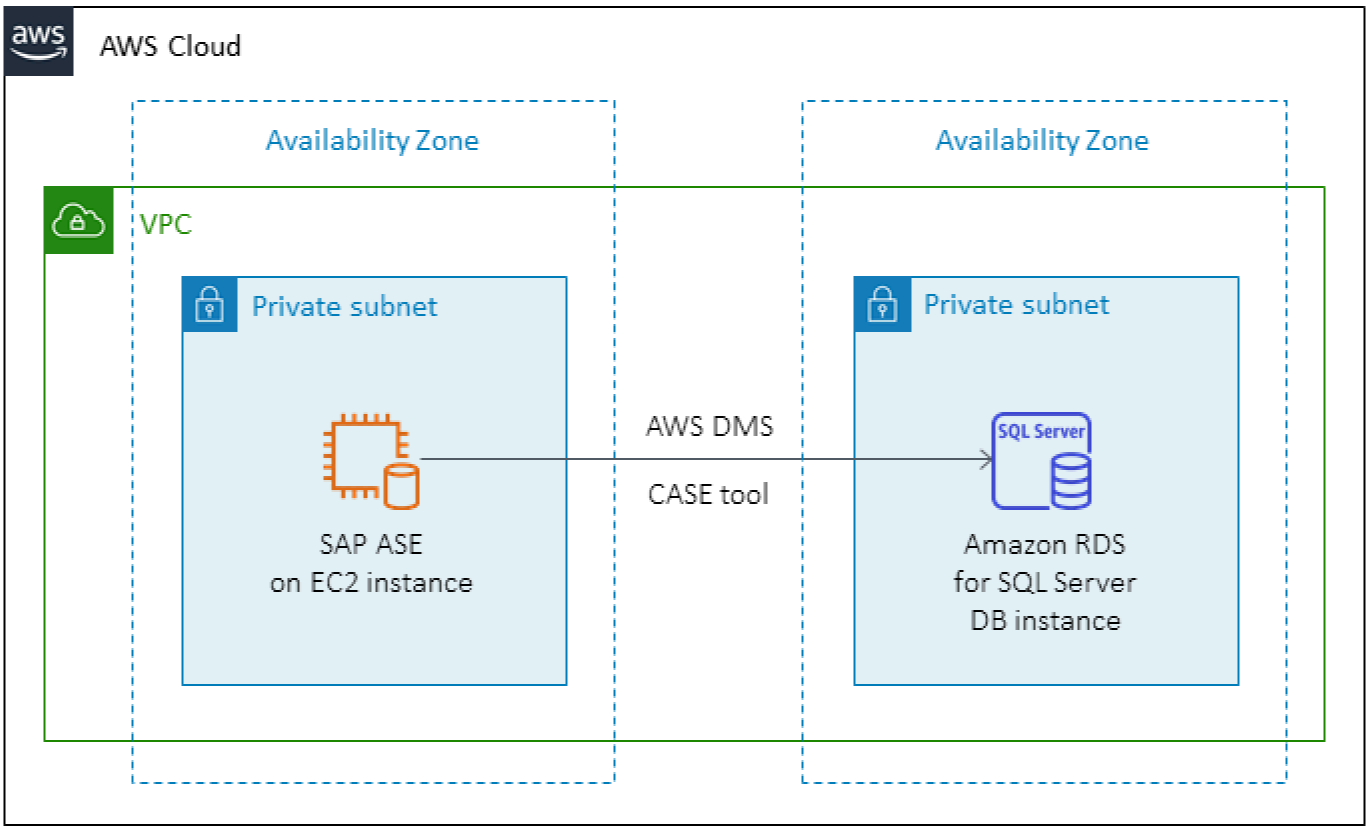
- An SAP ASE database that's on premises or on an Amazon EC2 instance

Target technology stack

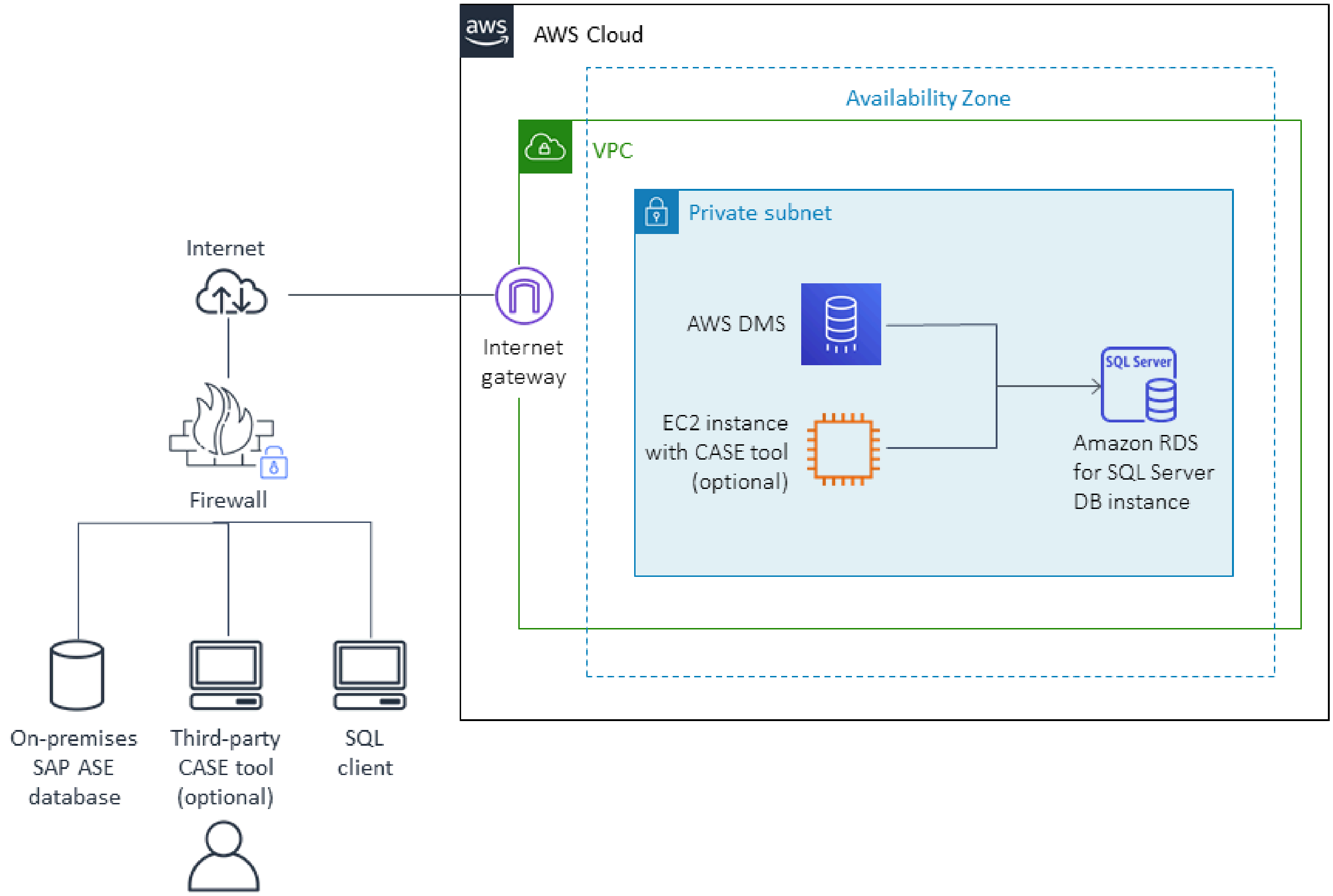
- An Amazon RDS for SQL Server DB instance

Source and target architecture

From an SAP ASE database on Amazon EC2 to an Amazon RDS for SQL Server DB instance:



From an on-premises SAP ASE database to an Amazon RDS for SQL Server DB instance:



Tools

- [AWS Database Migration Service](#) (AWS DMS) is a web service you can use to migrate data from your database that is on-premises, on an Amazon RDS DB instance, or in a database on an EC2 instance, to a database on an AWS service such as Amazon RDS for SQL Server or an EC2 instance. You can also migrate a database from an AWS service to an on-premises database. You can migrate data between heterogeneous or homogeneous database engines.
- For schema conversions, you can optionally use [erwin Data Modeler](#) or [SAP PowerDesigner](#).

Epics

Plan the migration

Task	Description	Skills required
Validate the source and target database versions.		DBA
Identify the storage requirements (storage type and capacity).		DBA, SysAdmin
Choose the proper instance type based on capacity, storage features, and network features.		DBA, SysAdmin
Identify the network access security requirements for the source and target databases.		DBA, SysAdmin
Identify the application migration strategy.		DBA, SysAdmin, App owner

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC) and subnets.		SysAdmin
Create security groups and network access control lists (ACLs).		SysAdmin
Configure and start an Amazon RDS DB instance.		SysAdmin

Migrate data - option 1

Task	Description	Skills required
Migrate the database schema manually or use a CASE tool such as erwin Data Modeler or SAP PowerDesigner.		DBA

Migrate data - option 2

Task	Description	Skills required
Migrate data using AWS DMS.		DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.		DBA, SysAdmin, App owner

Cut over

Task	Description	Skills required
Switch the application clients over to the new infrastru cture.		DBA, SysAdmin, App owner

Close the project

Task	Description	Skills required
Shut down the temporary AWS resources.		DBA, SysAdmin
Review and validate the project documents.		DBA, SysAdmin, App owner
Gather metrics such as time to migrate, percentage of manual versus automated tasks, and cost savings.		DBA, SysAdmin, App owner
Close out the project and provide feedback.		DBA, SysAdmin, App owner

Related resources

References

- AWS DMS website
- Amazon RDS Pricing
- Using a SAP ASE Database as a Source for AWS DMS
- Limitations for RDS Custom for SQL Server

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS DMS (video)
- Amazon RDS (video)

Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS

Marcelo Fernandes, Amazon Web Services

Summary

This pattern provides guidance for migrating an on-premises Microsoft SQL Server database to Amazon Redshift by using AWS Data Migration Service (AWS DMS).

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Microsoft SQL Server database in an on-premises data center
- Completed prerequisites for using an Amazon Redshift database as a target for AWS DMS, as discussed in the AWS DMS documentation

Product versions

- SQL Server 2005-2019, Enterprise, Standard, Workgroup, Developer, and Web editions. For the latest list of supported versions, see Using a Microsoft SQL Server Database as a Source for AWS DMS in the AWS documentation.

Architecture

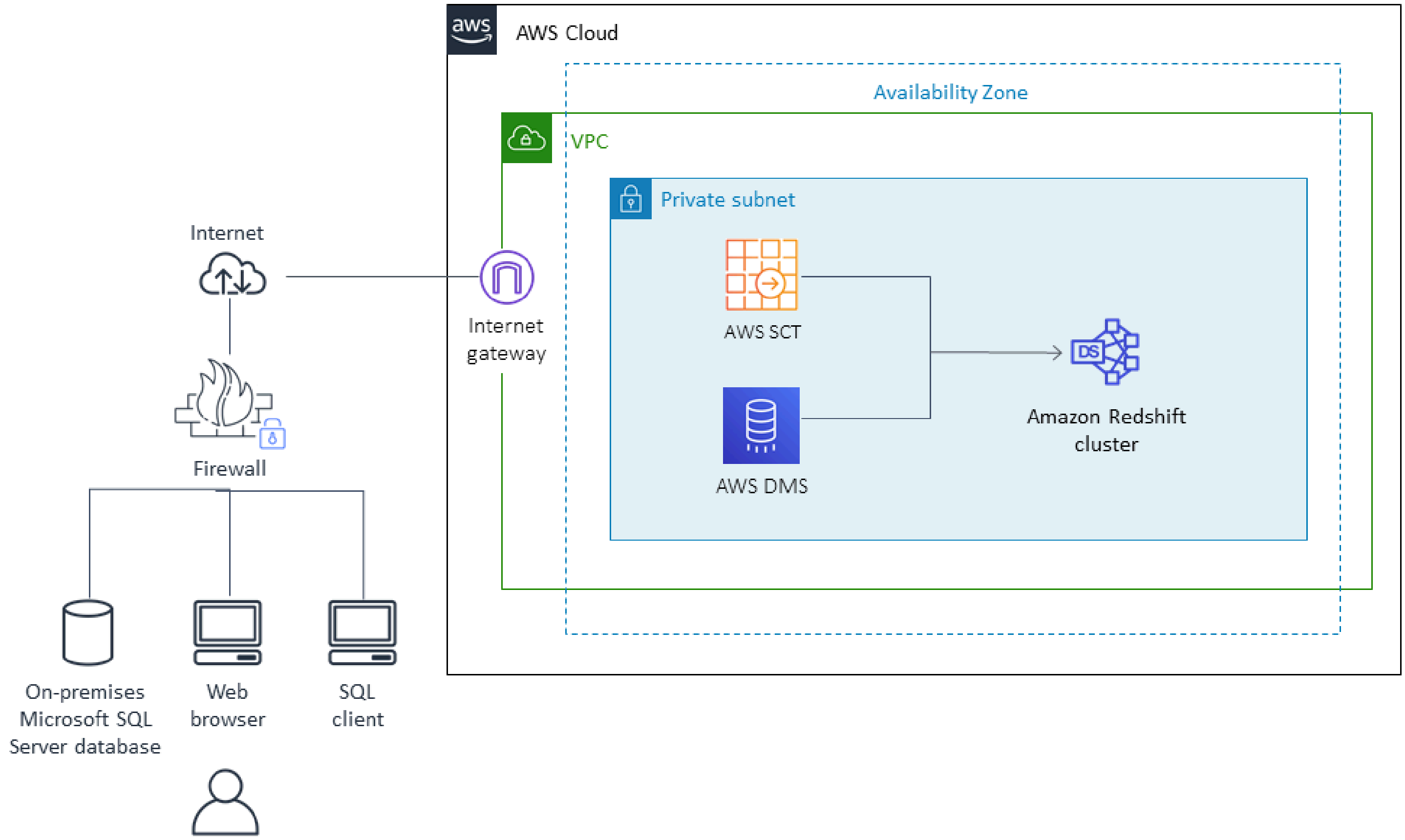
Source technology stack

- An on-premises Microsoft SQL Server database

Target technology stack

- Amazon Redshift

Data migration architecture



Tools

- [AWS DMS](#) is a data migration service that supports several types of source and target databases. For information about the Microsoft SQL Server database versions and editions that are supported for use with AWS DMS, see [Using a Microsoft SQL Server Database as a Source for AWS DMS](#) in the AWS DMS documentation. If AWS DMS doesn't support your source database, you must select an alternative method for data migration.

Epics

Plan the migration

Task	Description	Skills required
Validate the source and target database version and engine.		DBA

Task	Description	Skills required
Identify the hardware requirements for the target server instance.		DBA, Systems administrator
Identify the storage requirements (storage type and capacity).		DBA, Systems administrator
Choose the proper instance type based on capacity, storage features, and network features.		DBA, Systems administrator
Identify the network access security requirements for the source and target databases.		DBA, Systems administrator
Identify the application migration strategy.		DBA, App owner, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC).	For more information, see [Working with a DB instance in a VPC](#) in the AWS documenta tion.	Systems administrator
Create security groups.		Systems administrator
Configure and start an Amazon Redshift cluster.	For more information, see [Create a sample Amazon Redshift cluster](#) in the	DBA, Systems administrator

Task	Description	Skills required
	Amazon Redshift documenta tion.	

Migrate data

Task	Description	Skills required
Migrate the data from the Microsoft SQL Server database by using AWS DMS.		DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.		DBA, App owner, Systems administrator

Cut over

Task	Description	Skills required
Switch the application clients over to the new infrastru cture.		DBA, App owner, Systems administrator

Close the project

Task	Description	Skills required
Shut down the temporary resources.		DBA, Systems administrator

Task	Description	Skills required
Review and validate the project documents.		DBA, App owner, Systems administrator
Gather metrics such as time to migrate, percentage of manual versus automated tasks, and cost savings.		DBA, App owner, Systems administrator
Close out the project and provide feedback.		DBA, App owner, Systems administrator

Related resources

References

- AWS DMS documentation
- Amazon Redshift documentation
- Amazon Redshift Pricing

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started with Amazon Redshift
- Using an Amazon Redshift database as a target for AWS Database Migration Service
- AWS DMS (video)

Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents

Neha Thakur, Amazon Web Services

Summary

This pattern outlines steps for migrating an on-premises Microsoft SQL Server source database to an Amazon Redshift target database by using AWS Schema Conversion Tool (AWS SCT) data extraction agents. An agent is an external program that is integrated with AWS SCT but performs data transformation elsewhere and interacts with other AWS services on your behalf.

Prerequisites and limitations

Prerequisites

- A Microsoft SQL Server source database used for the data warehouse workload in an on-premises data center
- An active AWS account

Product versions

- Microsoft SQL Server version 2008 or later. For the latest list of supported versions, see [AWS SCT documentation](https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_Welcome.html).

Architecture

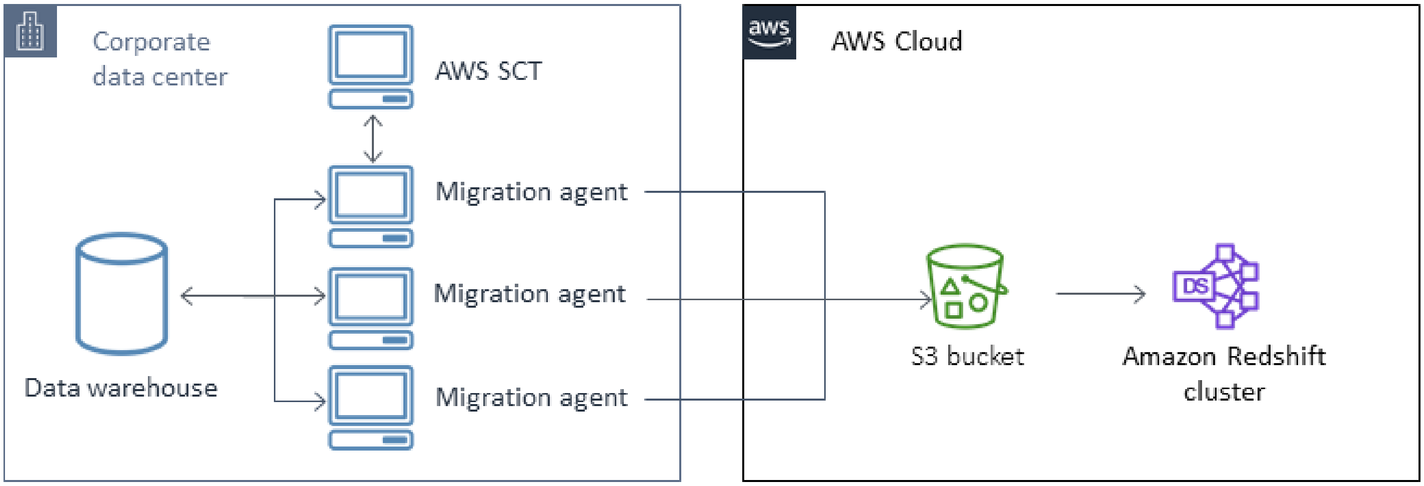
technology stackSource

- An on-premises Microsoft SQL Server database

technology stackTarget

- Amazon Redshift

Data migration architecture



Tools

- AWS Schema Conversion Tool (AWS SCT) handles heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database. When the source and target databases are very different, you can use an AWS SCT agent to perform additional data transformation. For more information, see Migrating Data from an On-Premises Data Warehouse to Amazon Redshift in the AWS documentation.

Best practices

- Best practices for AWS SCT
- Best practices for Amazon Redshift

Epics

Prepare for migration

Task	Description	Skills required
Validate the source and target database versions and engines.		DBA

Task	Description	Skills required
Identify hardware requireme nts for the target server instance.		DBA, SysAdmin
Identify storage requirements (storage type and capacity).		DBA, SysAdmin
Choose the proper instance type (capacity, storage features, network features).		DBA, SysAdmin
Identify network access security requirements for the source and target databases.		DBA, SysAdmin
Choose an application migration strategy.		DBA, SysAdmin, App owner

Configure infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC) and subnets.		SysAdmin
Create security groups.		SysAdmin
Configure and start the Amazon Redshift cluster.		SysAdmin

Migrate data

Task	Description	Skills required
Migrate the data using the AWS SCT data extraction agents.		DBA

Migrate applications

Task	Description	Skills required
Follow the chosen application migration strategy.		DBA, SysAdmin, App owner

Cut over to the target database

Task	Description	Skills required
Switch application clients over to the new infrastru cture.		DBA, SysAdmin, App owner

Close the project

Task	Description	Skills required
Shut down temporary AWS resources.		DBA, SysAdmin
Review and validate the project documents.		DBA, SysAdmin, App owner
Gather metrics such as time to migrate, percentage of		DBA, SysAdmin, App owner

Task	Description	Skills required
manual versus automated tasks, and cost savings.		
Close the project and provide any feedback.		DBA, SysAdmin, App owner

Related resources

References

- AWS SCT User Guide
- Using Data Extraction Agents
- Amazon Redshift Pricing

Tutorials and videos

- Getting Started with the AWS Schema Conversion Tool
- Getting Started with Amazon Redshift

Migrate legacy applications from Oracle Pro*C to ECPG

Sai Parthasaradhi and Mahesh Balumuri, Amazon Web Services

Summary

Most legacy applications that have embedded SQL code use the Oracle Pro*C precompiler to access the database. When you migrate these Oracle databases to Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition, you have to convert your application code to a format that's compatible with the precompiler in PostgreSQL, which is called ECPG. This pattern describes how to convert Oracle Pro*C code to its equivalent in PostgreSQL ECPG.

For more information about Pro*C, see the Oracle documentation. For a brief introduction to ECPG, see the Additional information section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible database
- An Oracle database running on premises

Tools

- The PostgreSQL packages listed in the next section.
- AWS CLI – The AWS Command Line Interface (AWS CLI) is an open-source tool for interacting with AWS services through commands in your command-line shell. With minimal configuration, you can run AWS CLI commands that implement functionality equivalent to that provided by the browser-based AWS Management Console from a command prompt.

Epics

Set the build environment on CentOS or RHEL

Task	Description	Skills required

Install PostgreSQL packages.	Install the required PostgreSQL packages by using the following commands.
	`yum update -y`
	`yum install -y yum-utils`
	`rpm -ivh https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/pgdg-redhat-repo-latest.noarch.rpm`
	`dnf -qy module disable postgresql`
		App developer, DevOps engineer

Install the header files and libraries.	Install the `postgresql12-devel` package, which contains header files and libraries, by using the following commands. Install the package in both the development and the runtime environments to avoid errors in the runtime environment.
	`dnf -y install postgresql12-devel`
	`yum install ncompress zip ghostscript jq unzip wget git -y`
		App developer, DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>For the development environment only, also run the following commands.<pre><code>yum install zlib-devel
 make -y
ln -s /usr/pgsql-12/
bin/ecpg /usr/bin/</code></pre></td><td></td></tr>
<tr><td>Configure the environment path variable.</td><td>Set the environment path for PostgreSQL client libraries.<pre><code>export PATH=$PATH:/usr/
pgsql-12/bin</code></pre></td><td>App developer, DevOps engineer</td></tr>
</table>

Task	Description	Skills required

Install additional software as necessary.	If required, install **pgLoader** as a replacement for **SQL*Loader** in Oracle.
	`wget -O /etc/yum.repos.d/pgloader-ccl.repo https://dl.packager.io/srv/opf/pgloader-ccl/master/installer/el/7.repo`
	`yum install pgloader-ccl -y`
	`ln -s /opt/pgloader-ccl/bin/pgloader /usr/bin/`
	If you're calling any Java applications from Pro*C modules, install Java.
	`yum install java -y`
	Install **ant** to compile the Java code.
	`yum install ant -y`
		App developer, DevOps engineer

Task	Description	Skills required

Install the AWS CLI.	Install the AWS CLI to run commands to interact with AWS services such as AWS Secrets Manager and Amazon Simple Storage Service (Amazon S3) from your applications.
	`cd /tmp/`
	`curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
	`unzip awscliv2.zip`
	`./aws/install -i /usr/local/aws-cli -b /usr/local/bin --update`
		App developer, DevOps engineer

Identify the programs to be converted.	Identify the applications that you want to convert from Pro*C to ECPG.	App developer, App owner

Convert Pro*C code to ECPG

Task	Description	Skills required
Remove unwanted headers.	Remove the `include` headers that are not required in PostgreSQL, such as `oci.h`, `oratypes`, and `sqlda`.	App owner, App developer
Update variable declarations.	Add `EXEC SQL` statements for all variable declarations	App developer, App owner

Task	Description	Skills required
	that are used as host variables. Remove the `EXEC SQL VAR` declarations such as the following from your application. `EXEC SQL VAR query IS STRING(2048);`	

Task	Description	Skills required

Update ROWNUM functionality.	The ROWNUM function isn't available in PostgreSQL. Replace this with the ROW_NUMBER window function in SQL queries.
	
	Pro*C code:
	
	`SELECT SUBSTR(RTRIM(FILE_NAME,'.txt'),12) INTO :gcpclFileseq FROM (SELECT FILE_NAME FROM DEMO_FILES_TABLE WHERE FILE_NAME LIKE '%POC%' ORDER BY FILE_NAME DESC) FL2 WHERE ROWNUM <=1 ORDER BY ROWNUM;`
	
	ECPG code:
	
	`SELECT SUBSTR(RTRIM(FILE_NAME,'.txt'),12) INTO :gcpclFileseq FROM (SELECT FILE_NAME , ROW_NUMBER() OVER (ORDER BY FILE_NAME DESC) AS ROWNUM FROM demo_schema.DEMO_FILES_TABLE WHERE FILE_NAME LIKE '%POC%' ORDER BY FILE_NAME DESC) FL2`
		App developer, App owner

Task	Description	Skills required
	``` WHERE ROWNUM <=1 ORDER BY ROWNUM; ```	

Update function parameters to use alias variables.	In PostgreSQL, function parameters can't be used as host variables. Overwrite them by using an alias variable.
	
	Pro*C code:
	
	``
	int processData(int
	 referenceId){
	  EXEC SQL char
	 col_val[100];
	  EXEC SQL select
	 column_name
	 INTO :col_val from
	 table_name where
	 col=:referenceId;
	}
	``
	
	ECPG code:
	
	``
	int processData(int
	 referenceIdParam){
	  EXEC SQL int reference
	Id = referenceIdParam;
	  EXEC SQL char
	 col_val[100];
	  EXEC SQL select
	 column_name
	 INTO :col_val from
	 table_name where
	 col=:referenceId;
	}
	``
		App developer, App owner

Task	Description	Skills required
Update struct types.	Define `struct` types in `EXEC SQL BEGIN` and `END` blocks with `typedef` if the `struct` type variables are used as host variables. If the `struct` types are defined in header (`.h`) files, include the files with `EXEC SQL` include statements. Pro*C code: Header file (`demo.h`) `struct s_partiti on_ranges { char sc_table_ group[31]; char sc_table_ name[31]; char sc_range_ value[10]; }; struct s_partiti on_ranges_ind { short ss_table_ group; short ss_table_ name; short ss_range_ value; };` ECPG code: Header file (`demo.h`)	App developer, App owner

Task	Description	Skills required

	``
	EXEC SQL BEGIN DECLARE
	 SECTION;
	typedef struct
	{
	  char sc_table_
	group[31];
	  char sc_table_
	name[31];
	  char sc_range_
	value[10];
	} s_partition_ranges;
	typedef struct
	{
	  short ss_table_
	group;
	  short ss_table_
	name;
	  short ss_range_
	value;
	} s_partition_ranges
	_ind;
	EXEC SQL END DECLARE
	 SECTION;
	``
	Pro*C file (demo.pc)
	``
	#include "demo.h"
	struct s_partiti
	on_ranges gc_partit
	ion_data[MAX_PART_
	TABLE] ;
	struct s_partiti
	on_ranges_ind
	 gc_partition_data_
	ind[MAX_PART_TABLE] ;
	``
	ECPG file (demo.pc)
	``
	exec sql include
	 "demo.h"
	``	

Task	Description	Skills required
	``` EXEC SQL BEGIN DECLARE SECTION; s_partition_ranges gc_partition_data[MAX_PART_TABLE] ; s_partition_ranges_ind gc_partition_data_ ind[MAX_PART_TABLE] ; EXEC SQL END DECLARE SECTION; ```	
Modify logic to fetch from cursors.	To fetch multiple rows from cursors by using array variables, change the code to use `FETCH FORWARD`. Pro*C code: ``` EXEC SQL char aPoeFiles [MAX_FILES][FILENA ME_LENGTH]; EXEC SQL FETCH filename_ cursor into :aPoeFile s; ``` ECPG code: ``` EXEC SQL char aPoeFiles [MAX_FILES][FILENA ME_LENGTH]; EXEC SQL int fetchSize = MAX_FILES; EXEC SQL FETCH FORWARD :fetchSiz e filename_cursor into :aPoeFiles; ```	App developer, App owner

Task	Description	Skills required

Modify package calls that don't have return values.	Oracle package functions that don’t have return values should be called with an indicator variable. If your application includes multiple functions that have the same name or if the unknown type functions generate runtime errors, typecast the values to the data types.
	
	Pro*C code:
	
	`void ProcessData (char *data , int id)`
	`{`
	`EXEC SQL EXECUTE BEGIN`
	`pkg_demo.process_data (:data, :id);`
	`END;`
	`END-EXEC;`
	`}`
	
	ECPG code:
	
	`void ProcessData (char *dataParam, int idParam )`
	`{`
	`EXEC SQL char *data = dataParam;`
	`EXEC SQL int id = idParam;`
		App developer, App owner

Task	Description	Skills required

	``
	        EXEC SQL short
	 rowInd;
	        EXEC SQL short
	 rowInd = 0;
	        EXEC SQL SELECT
	 pkg_demo.process_data
	 (
	
	 inp_data => :data::te
	xt,
	
	 inp_id => :id
	              )
	 INTO :rowInd;
	}
	``	

Task	Description	Skills required

Rewrite SQL_CURSOR variables.	Rewrite the SQL_CURSOR variable and its implement ation.
	
	Pro*C code:
	
	`/* SQL Cursor */`
	`SQL_CU`
	`RSOR      demo_cursor;`
	`EXEC SQL`
	` ALLOCATE :demo_cursor;`
	`EXEC SQL EXECUTE`
	`  BEGIN`
	`      pkg_demo.`
	`get_cursor(`
	`        demo_cur=`
	`>:demo_cursor`
	`      );`
	`  END;`
	`END-EXEC;`
	
	ECPG code:
	
	`EXEC SQL DECLARE`
	` demo_cursor CURSOR FOR`
	` SELECT`
	`         * from`
	`    pkg_demo.open_file`
	`name_rc(`
	`            demo_cur=`
	`>refcursor`
	`          ) ;`
	`EXEC SQL char open_file`
	`name_rcInd[100];`
	`# As the below function`
	` returns cursor_name`
	` as`
	`# return we need to`
	` use char[] type as`
	` indicator.`
		App developer, App owner

Task	Description	Skills required
	`EXEC SQL SELECT pkg_demo.get_cursor ( demo_cur=>'demo_cursor' ) INTO :open_filename_rcInd;`	
Apply common migration patterns.	• Change SQL queries so they're compatible with PostgreSQL. • Move anonymous blocks, when they aren't supported in ECPG, to the database. • Remove `dbms_application_info` logic, which isn't supported by PostgreSQL. • Move `EXEC SQL COMMIT` statements after the cursor close. If you commit queries while in the loop to fetch the records from the cursor, the cursor is closed and a cursor doesn't exist error is displayed. • For information about handling exceptions in ECPG and error codes, see [Error Handling](#) in the PostgreSQL documentation.	App developer, App owner

Task	Description	Skills required
Enable debugging, if required.	To run the ECPG program in debug mode, add the following command inside the main function block. `ECPGdebug(1, stderr);`	App developer, App owner

Compile ECPG programs

Task	Description	Skills required

Create an executable file for ECPG.	If you have an embedded SQL C source file named `prog1.pgc` , you can create an executable program by using the following sequence of commands.
	`ecpg prog1.pgc`
	`cc -I/usr/local/pgsql/include -c prog1.c`
	`cc -o prog1 prog1.o -L/usr/local/pgsql/lib -lecpg`
		App developer, App owner

Create a make file for compilation.	Create a make file to compile the ECPG program, as shown in the following sample file.
	`CFLAGS ::= $(CFLAGS) -I/usr/pgsql-12/include -g -Wall`
	`LDFLAGS ::= $(LDFLAGS) -L/usr/pgsql-12/lib -Wl,-rpath,/usr/pgsql-12/lib`
		App developer, App owner

Task	Description	Skills required

```
LDLIBS ::= $(LDLIBS) -
lecpg
PROGRAMS = test
.PHONY: all clean
%.c: %.pgc
      ecpg $<
all: $(PROGRAMS)
clean:
    rm -f $(PROGRAM
S) $(PROGRAMS:%=%.c)
 $(PROGRAMS:%=%.o)
```

Test the application

Task	Description	Skills required
Test the code.	Test the converted application code to make sure that it functions correctly.	App developer, App owner, Test engineer

Related resources

- ECPG - Embedded SQL in C (PostgreSQL documentation)
- Error Handling (PostgreSQL documentation)
- Why Use the Oracle Pro*C/C++ Precompiler (Oracle documentation)

Additional information

PostgreSQL has an embedded SQL precompiler, ECPG, which is equivalent to the Oracle Pro*C precompiler. ECPG converts C programs that have embedded SQL statements to standard C code by replacing the SQL calls with special function calls. The output files can then be processed with any C compiler tool chain.

Input and output files

ECPG converts each input file you specify on the command line to the corresponding C output file. If an input file name doesn’t have a file extension, .pgc is assumed. The file's extension is replaced by `.c` to construct the output file name. However, you can override the default output file name by using the `-o` option.

If you use a dash (`-`) as the input file name, ECPG reads the program from standard input and writes to standard output, unless you override that by using the `-o` option.

Header files

When the PostgreSQL compiler compiles the pre-processed C code files, it looks for the ECPG header files in the PostgreSQL `include` directory. Therefore, you might have to use the `-I` option to point the compiler to the correct directory (for example, `-I/usr/local/pgsql/include`).

Libraries

Programs that use C code with embedded SQL have to be linked against the `libecpg` library. For example, you can use the linker options `-L/usr/local/pgsql/lib -lecpg`.

Converted ECPG applications call functions in the `libpq` library through the embedded SQL library (`ecpglib`), and communicate with the PostgreSQL server by using the standard frontend/backend protocol.

Migrate virtual generated columns from Oracle to PostgreSQL

Veeranjaneyulu Grandhi, Rajesh Madiwale, and Ramesh Pathuri, Amazon Web Services

Summary

In version 11 and earlier, PostgreSQL doesn't provide a feature that is directly equivalent to an Oracle virtual column. Handling virtual generated columns while migrating from Oracle Database to PostgreSQL version 11 or earlier is difficult for two reasons:

- Virtual columns aren't visible during migration.
- PostgreSQL doesn't support the `generate` expression before version 12.

However, there are workarounds to emulate similar functionality. When you use AWS Database Migration Service (AWS DMS) to migrate data from Oracle Database to PostgreSQL version 11 and earlier, you can use trigger functions to populate the values in virtual generated columns. This pattern provides examples of Oracle Database and PostgreSQL code that you can use for this purpose. On AWS, you can use Amazon Relational Database Service (Amazon RDS) for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition for your PostgreSQL database.

Starting with PostgreSQL version 12, generated columns are supported. Generated columns can either be calculated from other column values on the fly, or calculated and stored. PostgreSQL generated columns are similar to Oracle virtual columns.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle database
- Target PostgreSQL databases (on Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible)
- PL/pgSQL coding expertise

Limitations

- Applies only to PostgreSQL versions before version 12.
- Applies to Oracle Database version 11g or later.

- Virtual columns are not supported in data migration tools.
- Applies only to columns defined in the same table.
- If a virtual generated column refers to a deterministic user-defined function, it cannot be used as a partitioning key column.
- The output of the expression must be a scalar value. It cannot return an Oracle supplied datatype, a user-defined type, `LOB`, or `LONG RAW`.
- Indexes that are defined against virtual columns are equivalent to function-based indexes in PostgreSQL.
- Table statistics must be gathered.

Tools

- [pgAdmin 4](#) is an open source management tool for PostgreSQL. This tool provides a graphical interface that simplifies the creation, maintenance, and use of database objects.
- [Oracle SQL Developer](#) is a free, integrated development environment for working with SQL in Oracle databases in both traditional and cloud deployments.

Epics

Create source and target database tables

Task	Description	Skills required
Create a source Oracle Database table.	In Oracle Database, create a table with virtual generated columns by using the following statement. `CREATE TABLE test.gene rated_column ( CODE NUMBER, STATUS VARCHAR2(12) DEFAULT 'PreOpen', FLAG CHAR(1) GENERATED ALWAYS AS (CASE UPPER(STATUS) WHEN 'OPEN' THEN 'N' ELSE`	DBA, App developer

Task	Description	Skills required
	``` 'Y' END) VIRTUAL VISIBLE); ``` In this source table, the data in the `STATUS` column is migrated through AWS DMS to the target database. The `FLAG` column, however, is populated by using `generate by` functionality, so this column isn't visible to AWS DMS during migration. To implement the functionality of `generated by`, you must use triggers and functions in the target database to populate the values in the `FLAG` column, as shown in the next epic.	

Task	Description	Skills required
Create a target PostgreSQL table on AWS.	Create a PostgreSQL table on AWS by using the following statement. `CREATE TABLE test.generated_column ( code integer not null, status character varying(12) not null , flag character(1) );` In this table, the `status` column is a standard column. The `flag` column will be a generated column based on the data in the `status` column.	DBA, App developer

Create a trigger function to handle the virtual column in PostgreSQL

Task	Description	Skills required
Create a PostgreSQL trigger.	In PostgreSQL, create a trigger. `CREATE TRIGGER tgr_gen_column AFTER INSERT OR UPDATE OF status ON test.generated_column FOR EACH ROW`	DBA, App developer

Task	Description	Skills required
	```EXECUTE FUNCTION test.tgf_gen_column();```	

Task	Description	Skills required
Create a PostgreSQL trigger function.	In PostgreSQL, create a function for the trigger. This function populates a virtual column that is inserted or updated by the application or AWS DMS, and validates the data. ```CREATE OR REPLACE FUNCTION test.tgf_gen_column() RETURNS trigger AS $VIRTUAL_COL$ BEGIN IF (TG_OP = 'INSERT') THEN IF (NEW.flag IS NOT NULL) THEN RAISE EXCEPTION 'ERROR: cannot insert into column "flag"' USING DETAIL = 'Column "flag" is a generated column.'; END IF; END IF; IF (TG_OP = 'UPDATE') THEN IF (NEW.flag::VARCHAR != OLD.flag::varchar) THEN RAISE EXCEPTION 'ERROR: cannot update column "flag"' USING DETAIL = 'Column "flag" is a generated column.'; END IF; END IF; IF TG_OP IN ('INSERT','UPDATE') THEN```	DBA, App developer

Task	Description	Skills required

	``
	IF (old.flag is NULL)
	 OR (coalesce(old.stat
	us,'') != coalesce(
	new.status,'')) THEN
	UPDATE test.gene
	rated_column
	SET flag = (CASE
	 UPPER(status) WHEN
	 'OPEN' THEN 'N' ELSE
	 'Y' END)
	WHERE code = new.code;
	END IF;
	END IF;
	RETURN NEW;
	END
	$VIRTUAL_COL$ LANGUAGE
	 plpgsql;
	``	

Test data migration by using AWS DMS

Task	Description	Skills required
Create a replication instance.	To create a replication instance, follow the [instructions](#) in the AWS DMS documentation. The replication instance should be in the same virtual private cloud (VPC) as your source and target databases.	DBA, App developer
Create source and target endpoints.	To create the endpoints, follow the [instructions](#) in the AWS DMS documentation.	DBA, App developer
Test the endpoint connections.	You can test the endpoint connections by specifying the	DBA, App developer

Task	Description	Skills required
	VPC and replication instance and choosing **Run test**.	
Create and start a full load task.	For instructions, see Creating a Task and Full-load task settings in the AWS DMS documentation.	DBA, App developer
Validate the data for the virtual column.	Compare the data in the virtual column in the source and target databases. You can validate the data manually or write a script for this step.	DBA, App developer

Related resources

- Getting started with AWS Database Migration Service (AWS DMS documentation)
- Using an Oracle database as a source for AWS DMS (AWS DMS documentation)
- Using a PostgreSQL database as a target for AWS DMS (AWS DMS documentation)
- Generated columns in PostgreSQL (PostgreSQL documentation)
- Trigger functions (PostgreSQL documentation)
- Virtual columns in Oracle Database (Oracle documentation)

Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible

Rakesh Raghav and anuradha chintha, Amazon Web Services

Summary

As part of your migration journey from Oracle to Amazon Aurora PostgreSQL-Compatible Edition on the Amazon Web Services (AWS) Cloud, you might encounter multiple challenges. For example, migrating code that relies on the Oracle `UTL_FILE` utility is always a challenge. In Oracle PL/SQL, the `UTL_FILE` package is used for file operations, such as read and write, in conjunction with the underlying operating system. The `UTL_FILE` utility works for both server and client machine systems.

Amazon Aurora PostgreSQL-Compatible is a managed database offering. Because of this, it isn't possible to access files on the database server. This pattern walks you through the integration of Amazon Simple Storage Service (Amazon S3) and Amazon Aurora PostgreSQL-Compatible to achieve a subset of `UTL_FILE` functionality. Using this integration, we can create and consume files without using third-party extract, transform, and load (ETL) tools or services.

Optionally, you can set up Amazon CloudWatch monitoring and Amazon SNS notifications.

We recommend thoroughly testing this solution before implementing it in a production environment.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Database Migration Service (AWS DMS) expertise
- Expertise in PL/pgSQL coding
- An Amazon Aurora PostgreSQL-Compatible cluster
- An S3 bucket

Limitations

This pattern doesn't provide the functionality to act as a replacement for the Oracle `UTL_FILE` utility. However, the steps and sample code can be enhanced further to achieve your database modernization goals.

Product versions

- Amazon Aurora PostgreSQL-Compatible Edition 11.9

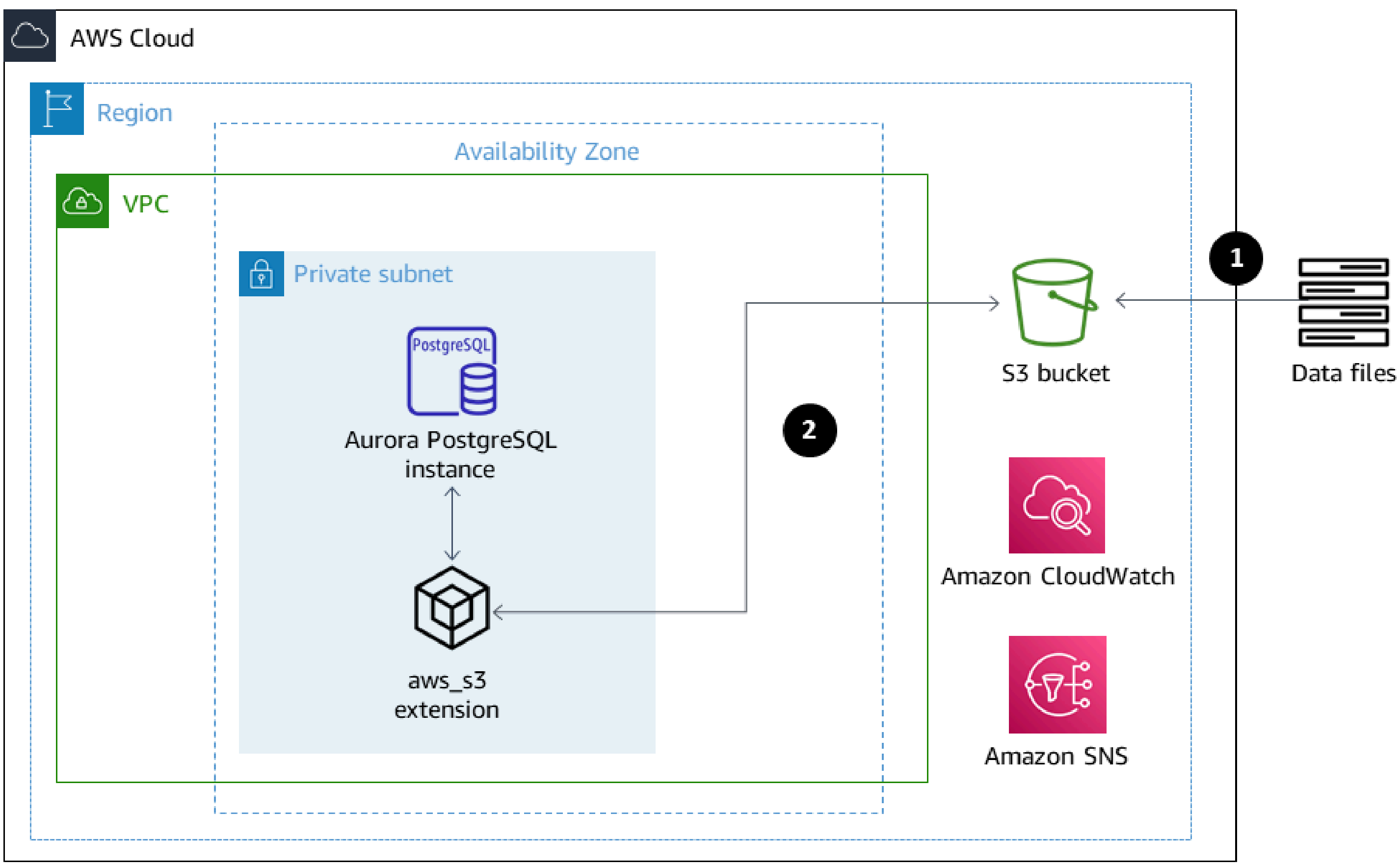
Architecture

Target technology stack

- Amazon Aurora PostgreSQL-Compatible
- Amazon CloudWatch
- Amazon Simple Notification Service (Amazon SNS)
- Amazon S3

Target architecture

The following diagram shows a high-level representation of the solution.



1. Files are uploaded from the application into the S3 bucket.

2. The `aws_s3` extension accesses the data, using PL/pgSQL, and uploads the data to Aurora PostgreSQL-Compatible.

Tools

- Amazon Aurora PostgreSQL-Compatible – Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine. It combines the speed and reliability of high-end commercial databases with the cost-effectiveness of open-source databases.
- AWS CLI – The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. With only one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.
- Amazon CloudWatch – Amazon CloudWatch monitors Amazon S3 resources and use.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. In this pattern, Amazon S3 provides a storage layer to receive and store files for consumption and transmission to and from the Aurora PostgreSQL-Compatible cluster.
- aws_s3 – The `aws_s3` extension integrates Amazon S3 and Aurora PostgreSQL-Compatible.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients. In this pattern, Amazon SNS is used to send notifications.
- pgAdmin – pgAdmin is an open-source management tool for Postgres. pgAdmin 4 provides a graphical interface for creating, maintaining, and using database objects.

Code

To achieve the required functionality, the pattern creates multiple functions with naming similar to `UTL_FILE`. The *Additional information* section contains the code base for these functions.

In the code, replace `testaurorabucket` with the name of your test S3 bucket. Replace `us-east-1` with the AWS Region where your test S3 bucket is located.

Epics

Integrate Amazon S3 and Aurora PostgreSQL-Compatible

Task	Description	Skills required
Set up IAM policies.	Create AWS Identity and Access Management (IAM) policies that grant access to the S3 bucket and objects in it. For the code, see the *Additional information* section.	AWS administrator, DBA
Add Amazon S3 access roles to Aurora PostgreSQL.	Create two IAM roles: one role for read and one role for write access to Amazon S3. Attach the two roles to the Aurora PostgreSQL-Compatible cluster: • One role for the S3Export feature • One role for the S3Import feature For more information, see the Aurora PostgreSQL-Compatible documentation on importing and exporting data to Amazon S3.	AWS administrator, DBA

Set up the extensions in Aurora PostgreSQL-Compatible

Task	Description	Skills required
Create the aws_commons extension.	The `aws_commons` extension is a dependency of the `aws_s3` extension.	DBA, Developer
Create the aws_s3 extension.	The `aws_s3` extension interacts with Amazon S3.	DBA, Developer

Validate Amazon S3 and Aurora PostgreSQL-Compatible integration

Task	Description	Skills required
Test importing files from Amazon S3 into Aurora PostgreSQL.	To test importing files into Aurora PostgreSQL-Compatible, create a sample CSV file and upload it into the S3 bucket. Create a table definition based on the CSV file, and load the file into the table by using the `aws_s3.table_import_from_s3` function.	DBA, Developer
Test exporting files from Aurora PostgreSQL to Amazon S3.	To test exporting files from Aurora PostgreSQL-Compatible, create a test table, populate it with data, and then export the data by using the `aws_s3.query_export_to_s3` function.	DBA, Developer

To mimic the UTL_FILE utility, create wrapper functions

Task	Description	Skills required
Create the utl_file_utility schema.	The schema keeps the wrapper functions together. To create the schema, run the following command. `CREATE SCHEMA utl_file_utility;`	DBA, Developer
Create the file_type type.	To create the `file_type` type, use the following code. `CREATE TYPE utl_file_utility.file_type AS ( p_path character varying(30), p_file_name character varying );`	DBA/Developer
Create the init function.	The `init` function initializes common variable such as `bucket` or `region`. For the code, see the *Additional information* section.	DBA/Developer
Create the wrapper functions.	Create the wrapper functions `fopen`, `put_line`, and `fclose`. For code, see the *Additional information* section.	DBA, Developer

Test the wrapper functions

Task	Description	Skills required
Test the wrapper functions in write mode.	To test the wrapper functions in write mode, use the code provided in the *Additional information* section.	DBA, Developer
Test the wrapper functions in append mode.	To test the wrapper functions in append mode, use the code provide in the *Additional information* section.	DBA, Developer

Related resources

- [Amazon S3 integration]
- [Amazon S3]
- [Aurora]
- [Amazon CloudWatch]
- [Amazon SNS]

Additional information

Set up IAM policies

Create the following policies.

Policy name	JSON
S3IntRead	`{ "Version": "2012-10-17", "Statement": [ { "Sid": "S3integrationtest ", "Effect": "Allow",`

```
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
         "arn:aws:s3:::testaurorabuc
ket/*",
         "arn:aws:s3:::testaurorabuc
ket"
            ]
        }
    ]
}
```

S3IntWrite

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3integrationtest
",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",

                "s3:ListBucket"
            ],
            "Resource": [
    "arn:aws:s3:::testaurorabucket/
*",                "arn:aws:s3:::test
aurorabucket"
            ]
        }
    ]
}
```

Create the init function

To initialize common variables, such as `bucket` or `region`, create the `init` function by using the following code.

```
CREATE OR REPLACE FUNCTION utl_file_utility.init(
```

```
    )
    RETURNS void
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
BEGIN
      perform set_config
      ( format( '%s.%s','UTL_FILE_UTILITY', 'region' )
      , 'us-east-1'::text
      , false );

      perform set_config
      ( format( '%s.%s','UTL_FILE_UTILITY', 's3bucket' )
      , 'testaurorabucket'::text
      , false );
END;
$BODY$;
```

Create the wrapper functions

Create the `fopen`, `put_line`, and `fclose` wrapper functions.

fopen

```
CREATE OR REPLACE FUNCTION utl_file_utility.fopen(
    p_file_name character varying,
    p_path character varying,
    p_mode character DEFAULT 'W'::bpchar,
    OUT p_file_type utl_file_utility.file_type)
    RETURNS utl_file_utility.file_type
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
declare
    v_sql character varying;
    v_cnt_stat integer;
    v_cnt integer;
    v_tabname character varying;
    v_filewithpath character varying;
    v_region character varying;
```

```
    v_bucket character varying;

BEGIN
    /*initialize common variable */
    PERFORM utl_file_utility.init();
    v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 'region' ) );
    v_bucket :=  current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 's3bucket' ) );

    /* set tabname*/
    v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
 length(p_file_name) else strpos(p_file_name,'.') - 1 end );
    v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
 concat_ws('/',p_path,p_file_name) end ;
    raise notice 'v_bucket %, v_filewithpath % , v_region %', v_bucket,v_filewithpath,
 v_region;

    /* APPEND MODE HANDLING; RETURN EXISTING FILE DETAILS IF PRESENT ELSE CREATE AN
 EMPTY FILE */
    IF p_mode = 'A' THEN
        v_sql := concat_ws('','create temp table if not exists ', v_tabname,' (col1
 text)');
        execute v_sql;

        begin
        PERFORM aws_s3.table_import_from_s3
            ( v_tabname,
            '',
            'DELIMITER AS ''#''',
            aws_commons.create_s3_uri
            (     v_bucket,
                v_filewithpath ,
                v_region)
            );
        exception
            when others then
             raise notice 'File load issue ,%',sqlerrm;
             raise;
        end;
        execute concat_ws('','select count(*) from ',v_tabname) into v_cnt;

        IF v_cnt > 0
        then
            p_file_type.p_path := p_path;
            p_file_type.p_file_name := p_file_name;
```

```
        else
            PERFORM aws_s3.query_export_to_s3('select ''''',
                            aws_commons.create_s3_uri(v_bucket, v_filewithpath,
 v_region)
                              );

            p_file_type.p_path := p_path;
            p_file_type.p_file_name := p_file_name;
        end if;
        v_sql := concat_ws('','drop table ', v_tabname);
        execute v_sql;
    ELSEIF p_mode = 'W' THEN
            PERFORM aws_s3.query_export_to_s3('select ''''',
                            aws_commons.create_s3_uri(v_bucket, v_filewithpath,
 v_region)
                              );
            p_file_type.p_path := p_path;
            p_file_type.p_file_name := p_file_name;
    END IF;

EXCEPTION
        when others then
            p_file_type.p_path := p_path;
            p_file_type.p_file_name := p_file_name;
            raise notice 'fopenerror,%',sqlerrm;
            raise;
END;
$BODY$;
```

put_line

```
CREATE OR REPLACE FUNCTION utl_file_utility.put_line(
    p_file_name character varying,
    p_path character varying,
    p_line text,
    p_flag character DEFAULT 'W'::bpchar)
    RETURNS boolean
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
/**************************************************************************
```

```
* Write line, p_line in windows format to file, p_fp - with carriage return
* added before new line.
**************************************************************************/
declare
    v_sql varchar;
    v_ins_sql varchar;
    v_cnt INTEGER;
    v_filewithpath character varying;
    v_tabname  character varying;
    v_bucket character varying;
    v_region character varying;

BEGIN
 PERFORM utl_file_utility.init();

/* check if temp table already exist */

 v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
 length(p_file_name) else strpos(p_file_name,'.') - 1 end );

 v_sql := concat_ws('','select count(1) FROM pg_catalog.pg_class c LEFT JOIN
 pg_catalog.pg_namespace n ON n.oid = c.relnamespace where n.nspname like ''pg_temp_
%'''
                         ,' AND pg_catalog.pg_table_is_visible(c.oid) AND
 Upper(relname) = Upper( '''
                         ,  v_tabname ,''' ) ');

 execute v_sql into v_cnt;

  IF v_cnt = 0 THEN
         v_sql := concat_ws('','create temp table ',v_tabname,' (col text)');
        execute v_sql;
        /* CHECK IF APPEND MODE */
        IF upper(p_flag) = 'A' THEN
            PERFORM utl_file_utility.init();
            v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY',
 'region' ) );
            v_bucket :=  current_setting( format( '%s.%s', 'UTL_FILE_UTILITY',
 's3bucket' ) );

            /* set tabname*/
            v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
 concat_ws('/',p_path,p_file_name) end ;
```

```
            begin
               PERFORM aws_s3.table_import_from_s3
                     ( v_tabname,
                          '',
                       'DELIMITER AS ''#''',
                        aws_commons.create_s3_uri
                           ( v_bucket,
                               v_filewithpath,
                               v_region    )
                    );
            exception
                when others then
                    raise notice  'Error Message : %',sqlerrm;
                    raise;
            end;
        END IF;
    END IF;
    /* INSERT INTO TEMP TABLE */
    v_ins_sql := concat_ws('','insert into ',v_tabname,' values(''',p_line,''')');
    execute v_ins_sql;
    RETURN TRUE;
    exception
            when others then
                raise notice  'Error Message : %',sqlerrm;
                raise;
END;
$BODY$;
```

fclose

```
CREATE OR REPLACE FUNCTION utl_file_utility.fclose(
    p_file_name character varying,
    p_path character varying)
    RETURNS boolean
    LANGUAGE 'plpgsql'

    COST 100
    VOLATILE
AS $BODY$
DECLARE
    v_filewithpath character varying;
    v_bucket character varying;
    v_region character varying;
```

```
    v_tabname character varying;
    v_sql character varying;
BEGIN
      PERFORM utl_file_utility.init();

    v_region := current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 'region' ) );
    v_bucket :=  current_setting( format( '%s.%s', 'UTL_FILE_UTILITY', 's3bucket' ) );

    v_tabname := substring(p_file_name,1,case when strpos(p_file_name,'.') = 0 then
 length(p_file_name) else strpos(p_file_name,'.') - 1 end );
    v_filewithpath := case when NULLif(p_path,'') is null then p_file_name else
 concat_ws('/',p_path,p_file_name) end ;

    raise notice 'v_bucket %, v_filewithpath % , v_region %', v_bucket,v_filewithpath,
 v_region ;

    /* exporting to s3 */
    perform aws_s3.query_export_to_s3
        (concat_ws('','select * from ',v_tabname,'  order by ctid asc'),
            aws_commons.create_s3_uri(v_bucket, v_filewithpath, v_region)
        );
    v_sql := concat_ws('','drop table ', v_tabname);
    execute v_sql;
    RETURN TRUE;
EXCEPTION
       when others then
     raise notice 'error fclose %',sqlerrm;
     RAISE;
END;
$BODY$;
```

Test your setup and wrapper functions

Use the following anonymous code blocks to test your setup.

Test the write mode

The following code writes a file named `s3inttest` in the S3 bucket.

```
do $$
declare
l_file_name varchar := 's3inttest' ;
l_path varchar := 'integration_test' ;
```

```
l_mode char(1) := 'W';
l_fs utl_file_utility.file_type ;
l_status boolean;

begin
select * from
utl_file_utility.fopen( l_file_name, l_path , l_mode ) into l_fs ;
raise notice 'fopen : l_fs : %', l_fs;

select * from
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket: for
 test purpose', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;

select * from utl_file_utility.fclose( l_file_name , l_path ) into l_status ;
raise notice 'fclose : l_status %', l_status;

end;
$$
```

Test the append mode

The following code appends lines onto the `s3inttest` file that was created in the previous test.

```
do $$
declare
l_file_name varchar := 's3inttest' ;
l_path varchar := 'integration_test' ;
l_mode char(1) := 'A';
l_fs utl_file_utility.file_type ;
l_status boolean;

begin
select * from
utl_file_utility.fopen( l_file_name, l_path , l_mode ) into l_fs ;
raise notice 'fopen : l_fs : %', l_fs;


select * from
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket: for
 test purpose : append 1', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;

select * from
```

```
utl_file_utility.put_line( l_file_name, l_path ,'this is test file:in s3bucket : for
 test purpose : append 2', l_mode ) into l_status ;
raise notice 'put_line : l_status %', l_status;

select * from utl_file_utility.fclose( l_file_name , l_path ) into l_status ;
raise notice 'fclose : l_status %', l_status;

end;
$$
```

Amazon SNS notifications

Optionally, you can set up Amazon CloudWatch monitoring and Amazon SNS notifications on the S3 bucket. For more information, see Monitoring Amazon S3 and Setting up Amazon SNS Notifications.

Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL

Venkatramana Chintha and Eduardo Valentim, Amazon Web Services

Summary

This pattern describes a step-by-step approach to validate objects after migrating an Oracle database to Amazon Aurora PostgreSQL-Compatible Edition.

This pattern outlines usage scenarios and steps for database object validation; for more detailed information, see Validating database objects after migration using AWS SCT and AWS DMS on the AWS Database blog.

Prerequisites and limitations

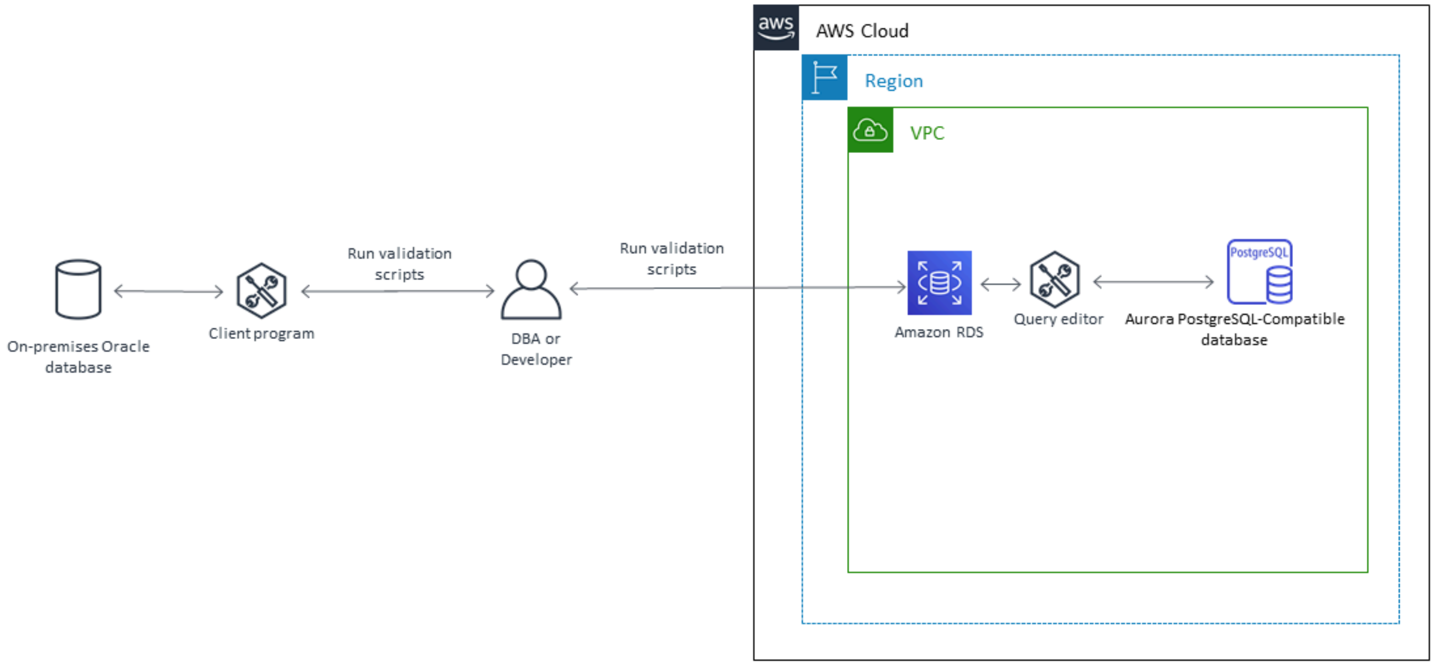
Prerequisites

- An active AWS account.
- An on-premises Oracle database that was migrated to an Aurora PostgreSQL-Compatible database.
- Sign-in credentials that have the AmazonRDSDataFullAccess policy applied, for the Aurora PostgreSQL-Compatible database.
- This pattern uses the query editor for Aurora Serverless DB clusters, which is available in the Amazon Relational Database Service (Amazon RDS) console. However, you can use this pattern with any other query editor.

Limitations

- Oracle SYNONYM objects are not available in PostgreSQL but can be partially validated through **views** or SET search_path queries.
- The Amazon RDS query editor is available only in certain AWS Regions and for certain MySQL and PostgreSQL versions.

Architecture



Tools

Tools

- Amazon Aurora PostgreSQL-Compatible Edition – Aurora PostgreSQL-Compatible is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases.
- Amazon RDS – Amazon Relational Database Service (Amazon RDS) makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.
- Query Editor for Aurora Severless – Query editor helps you run SQL queries in the Amazon RDS console. You can run any valid SQL statement on the Aurora Serverless DB cluster, including data manipulation and data definition statements.

To validate the objects, use the full scripts in the "Object validation scripts" file in the "Attachments" section. Use the following table for reference.

Oracle object	Script to use

Packages	Query 1
Tables	Query 3
Views	Query 5
Sequences	Query 7
Triggers	Query 9
Primary keys	Query 11
Indexes	Query 13
Check constraints	Query 15
Foreign keys	Query 17

PostgreSQL object	Script to use
Packages	Query 2
Tables	Query 4
Views	Query 6
Sequences	Query 8
Triggers	Query 10
Primary keys	Query 12
Indexes	Query 14
Check constraints	Query 16
Foreign keys	Query 18

Epics

Validate objects in the source Oracle database

Task	Description	Skills required
Run the "packages" validation query in the source Oracle database.	Download and open the "Object validation scripts" file from the "Attachments" section. Connect to the source Oracle database through your client program. Run the "Query 1" validations script from the "Object validation scripts" file. Important: Enter your Oracle user name instead of "your_schema" in the queries. Make sure you record your query results.	Developer, DBA
Run the "tables" validation query.	Run the "Query 3" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "views" validation query.	Run the "Query 5" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "sequences" count validation.	Run the "Query 7" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "triggers" validation query.	Run the "Query 9" script from the "Object validation scripts"	Developer, DBA

Task	Description	Skills required
	file. Make sure you record your query results.	
Run the "primary keys" validation query.	Run the "Query 11" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "indexes" validation query.	Run the "Query 13" validation script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "check constraints" validation query.	Run the "Query 15" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "foreign keys" validation query.	Run the "Query 17" validation script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA

Validate objects in the target Aurora PostgreSQL-Compatible database

Task	Description	Skills required
Connect to the target Aurora PostgreSQL-Compatible database by using the query editor.	Sign in to the AWS Management Console and open the Amazon RDS console. In the upper-right corner, choose the AWS Region in which you created	Developer, DBA

Task	Description	Skills required
	the Aurora PostgreSQL-Compatible database. In the navigation pane, choose "Databases," and choose the target Aurora PostgreSQL-Compatible database. In "Actions," choose "Query." Important: If you haven't connected to the database before, the "Connect to database" page opens. You then need to enter your database information, such as user name and password.	
Run the "packages" validation query.	Run the "Query 2" script from the "Object validation scripts" file in the "Attachments" section. Make sure you record your query results.	Developer, DBA
Run the "tables" validation query.	Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the "Query 4" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA

Task	Description	Skills required
Run the "views" validation query.	Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the "Query 6" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "sequences" count validation.	Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the "Query 8" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "triggers" validation query.	Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the "Query 10" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "primary keys" validation query.	Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the "Query 12" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA

Task	Description	Skills required
Run the "indexes" validation query.	Return to the query editor for the Aurora PostgreSQL-Compatible database, and run the "Query 14" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "check constraints" validation query.	Run the "Query 16" script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA
Run the "foreign keys" validation query.	Run the "Query 18" validatio n script from the "Object validation scripts" file. Make sure you record your query results.	Developer, DBA

Compare source and target database validation records

Task	Description	Skills required
Compare and validate both query results.	Compare the query results of the Oracle and Aurora PostgreSQL-Compatible databases to validate all objects. If they all match, then all objects have been successfully validated.	Developer, DBA

Related resources

- [Validating database objects after a migration using AWS SCT and AWS DMS]()

- [Amazon Aurora Features: PostgreSQL-Compatible Edition]

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Rehost

Topics

- Accelerate the discovery and migration of Microsoft workloads to AWS
- Create an approval process for firewall requests during a rehost migration to AWS
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account
- Migrate a Couchbase Server database to Amazon EC2
- Migrate Db2 for LUW to Amazon EC2 by using log shipping to reduce outage time
- Migrate Db2 for LUW to Amazon EC2 with high availability disaster recovery
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using Application Migration Service
- Migrate VMware VMs with HCX Automation by using PowerCLI
- Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud
- Migrate an on-premises Go web application to AWS Elastic Beanstalk by using the binary method
- Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP
- Migrate an on-premises VM to Amazon EC2 by using AWS Application Migration Service
- Migrate small sets of data from on premises to Amazon S3 using AWS SFTP
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk
- Migrate an on-premises Oracle database to Oracle on Amazon EC2
- Migrate an on-premises Oracle database to Amazon EC2 by using Oracle Data Pump
- Migrate RHEL BYOL systems to AWS License-Included instances by using AWS MGN
- Migrate an on-premises SAP ASE database to Amazon EC2
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2
- Migrate an on-premises MySQL database to Amazon EC2
- Reduce homogeneous SAP migration cutover time by using Application Migration Service
- Rehost on-premises workloads in the AWS Cloud: migration checklist
- Set up Multi-AZ infrastructure for a SQL Server Always On FCI by using Amazon FSx
- Use BMC Discovery queries to extract migration data for migration planning

Accelerate the discovery and migration of Microsoft workloads to AWS

Ali Alzand, Amazon Web Services

Summary

This pattern shows you how to use the Migration Validator Toolkit PowerShell module to discover and migrate your Microsoft workloads to AWS. The module works by performing multiple checks and validations for common tasks associated with any Microsoft workload. For example, the module checks for instances that might have multiple disks attached to it or instances that use many IP addresses. For a full list of checks that the module can perform, see the Checks section on the module's GitHub page.

The Migration Validator Toolkit PowerShell module can help your organization reduce the time and effort involved in discovering what applications and services are running on your Microsoft workloads. The module can also help you identify the configurations of your workloads so that you can find out if your configurations are supported on AWS. The module also provides recommendations for next steps and mitigation actions, so that you can avoid any misconfigurations before, during, or after your migration.

Prerequisites and limitations

Prerequisites

- Local administrator account
- PowerShell 4.0

Limitations

- Works only on Microsoft Windows Server 2012 R2 or later

Tools

Tools

- PowerShell 4.0

Code repository

The Migration Validator Toolkit PowerShell module for this pattern is available in the GitHub migration-validator-toolkit-for-microsoft-workloads repository.

Epics

Run the Migration Validator Toolkit PowerShell module on a single target

Task	Description	Skills required
Download, extract, import, and invoke the module.	Choose one of the following methods to download and deploy the module: • Run the PowerShell script • Download and extract the .zip file • Clone the GitHub repository **Run the PowerShell script** In PowerShell, run the following example code: `#MigrationValidatorToolkit $uri = 'https://github.com/aws-samples/migration-validator-toolkit-for-microsoft-workloads/archive/refs/heads/main.zip' $destination = (Get-Location).Path if ((Test-Path -Path "$destination\MigrationValidatorToolkit.zip" -PathType Leaf) -or (Test-Path -Path "$destination\Migr`	System Administrator

Task	Description	Skills required

	``
	ationValidatorTool
	kit")) {
	    write-host "File
	 $destination\Migra
	tionValidatorToolk
	it.zip or folder
	 $destination\Migra
	tionValidatorToolkit
	 found, exiting"
	}else {
	    Write-host "Enable
	 TLS 1.2 for this
	 PowerShell session
	 only."
	    [Net.ServicePointM
	anager]::SecurityP
	rotocol = [Net.Secu
	rityProtocolType]:
	:Tls12
	    $webClient =
	 New-Object System.Ne
	t.WebClient
	    Write-host
	 "Downloading Migration
	ValidatorToolkit.zip"
	    $webClient.Downloa
	dFile($uri, "$destina
	tion\MigrationVali
	datorToolkit.zip")
	    Write-host
	 "MigrationValidato
	rToolkit.zip download
	 successfully"
	    Add-Type -Assembly
	 "system.io.compres
	sion.filesystem"
	    [System.IO.Compres
	sion.ZipFile]::Ext
	ractToDirectory("$
	destination\Migrat
	ionValidatorToolki
	t.zip","$destinati
	``	

Task	Description	Skills required

	`on\MigrationValidatorToolkit")`
	`    Write-host "Extracting MigrationValidatorToolkit.zip complete successfully"`
	`    Import-Module "$destination\MigrationValidatorToolkit\migration-validator-toolkit-for-microsoft-workloads-main\MigrationValidatorToolkit.psm1"; Invoke-MigrationValidatorToolkit`
	`}`
	
	The code downloads the module from a .zip file. Then, the code extracts, imports, and invokes the module.
	
	Download and extract the .zip file
	
	1. Download the [.zip file](#) (download).
	2. Extract the .zip file.
	3. Follow the steps in the *Invoke the module manually* story of this guide.
	
	Clone the GitHub repository
	
	1. To clone the GitHub [migration-validato](#)	

Task	Description	Skills required
	[r-toolkit-for-microsoft-wor kloads](#) repository, run the following Git command in a terminal window: `git clone https://github.com/aws-samples/migration-validator-toolkit-for-microsoft-workloads.git` 2. Follow the steps in the *Invoke the module manually* story of this guide.	

Task	Description	Skills required

Invoke the module manually.	1. Go to the directory where the downloaded module is stored.
	2. To generate the output of your choice, run one of the following commands as an administrator in PowerShell:
	
	Format-Table format:
	`Import-Module .\MigrationValidatorToolkit.psm1;Invoke-MigrationValidatorToolkit`
	
	Format-List format:
	`Import-Module .\MigrationValidatorToolkit.psm1;Invoke-MigrationValidatorToolkit -List`
	
	Out-GridView format:
	`Import-Module .\MigrationValidatorToolkit.psm1;Invoke-MigrationValidatorToolkit -GridView`
	
	ConvertTo-Csv format:
	`Import-Module .\MigrationValidatorToolkit`
		System Administrator

Task	Description	Skills required
	`.psm1;Invoke-MigrationValidatorToolkit -csv`	

Run the Migration Validator Toolkit PowerShell module on multiple targets

Task	Description	Skills required
Download the .zip file or clone the GitHub repository.	Choose one of the following options: • Download the zip file. (download). • To clone the GitHub migration-validator-toolkit-for-microsoft-workloads repository, run the following Git command in a terminal window: `git clone https://github.com/aws-samples/migration-validator-toolkit-for-microsoft-workloads.git`	System Administrator
Update the server.csv list.	If you downloaded the .zip file, follow these steps: 1. Extract the .zip file. 2. Go to the `MigrationValidatorToolkit\Inputs\` directory.	System Administrator

Task	Description	Skills required
	3. Update `serverlist.csv` with the host name of your target computers.	
Invoke the module.	You can use any computer within the domain that uses a domain user that has administrator access to target computers. 1. Download the source code as a .zip file and extract the file. 2. As an administrator in PowerShell, run the following command: `Import-Module .\MigrationValidatorToolkit.psm1;Invoke-DomainComputers` The output .csv file is saved in `MigrationValidatorToolkit\Outputs\folder` with the prefix name `DomainComputers_MigrationAutomations_YYYY-MM-DDTHH-MM-SS`.	System Administrator

Troubleshooting

Issue	Solution
`MigrationValidatorToolkit` writes information about executions, commands, and errors to log files on the running host.	You can view log files manually in the following location: 1. Go to the `MigrationValidatorToolkit\logs\` directory. 2. Locate the log file. The format of the log file name is: `ComputerName_MigrationValidatorToolkit_YYYY-MM-SSTHH-MM-SS.log`

Related resources

- Options, tools, and best practices for migrating Microsoft workloads to AWS (AWS Prescriptive Guidance)
- Microsoft migration patterns (AWS Prescriptive Guidance)
- Free Cloud Migration Services on AWS (AWS documentation)
- Predefined post-launch actions (Application marketing documentation)

Additional information

Frequently asked questions

Where can I run the Migration Validator Toolkit PowerShell module?

You can run the module on Microsoft Windows Server 2012 R2 or later.

When do I run this module?

We recommend that you run the module during the assess phase of the migration journey.

Does the module modify my existing servers?

No. All actions in this module are read-only.

How long does it take to run the module?

It typically takes 1–5 minutes to run the module, but it depends on the resource allocation of your server.

What permissions does the module need to run?

You must run the module from a local administrator account.

Can I run the module on physical servers?

Yes, as long as the operating system is Microsoft Windows Server 2012 R2 or later.

How do I run the module at scale for multiple servers?

To run the module on multiple domain-joined computers at scale, follow the steps in the *Run the Migration Validator Toolkit PowerShell module on multiple targets* epic of this guide. For non domain-joined computers, use a remote invocation or run the module locally by following the steps in the *Run the Migration Validator Toolkit PowerShell module on a single target* epic of this guide.

Create an approval process for firewall requests during a rehost migration to AWS

Srikanth Rangavajhala, Amazon Web Services

Summary

If you want to use AWS Application Migration Service or Cloud Migration Factory on AWS for a rehost migration to the AWS Cloud, one of the prerequisites is that you must keep TCP ports 443 and 1500 open. Typically, opening these firewall ports requires approval from your information security (InfoSec) team.

This pattern outlines the process to obtain a firewall request approval from an InfoSec team during a rehost migration to the AWS Cloud. You can use this process to avoid rejections of your firewall request by the InfoSec team, which can become expensive and time consuming. The firewall request process has two review and approval steps between AWS migration consultants and leads who work with your InfoSec and application teams to open the firewall ports.

This pattern assumes that you are planning a rehost migration with AWS consultants or migration specialists from your organization. You can use this pattern if your organization doesn't have a firewall approval process or firewall request blanket approval form. For more information about this, see the *Limitations* section of this pattern. For more information on network requirements for Application Migration Service, see Network requirements in the Application Migration Service documentation.

Prerequisites and limitations

Prerequisites

- A planned rehost migration with AWS consultants or migration specialists from your organization
- The required port and IP information to migrate the stack
- Existing and future state architecture diagrams
- Firewall information about the on-premises and destination infrastructure, ports, and zone-to-zone traffic flow
- A firewall request review checklist (attached)
- A firewall request document, configured according to your organization's requirements
- A contact list for the firewall reviewers and approvers, including the following roles:
  - **Firewall request submitter** – AWS migration specialist or consultant. The firewall request submitter can also be a migration specialist from your organization.

- **Firewall request reviewer –** Typically, this is the single point of contact (SPOC) from AWS.
- **Firewall request approver** – An InfoSec team member.

Limitations

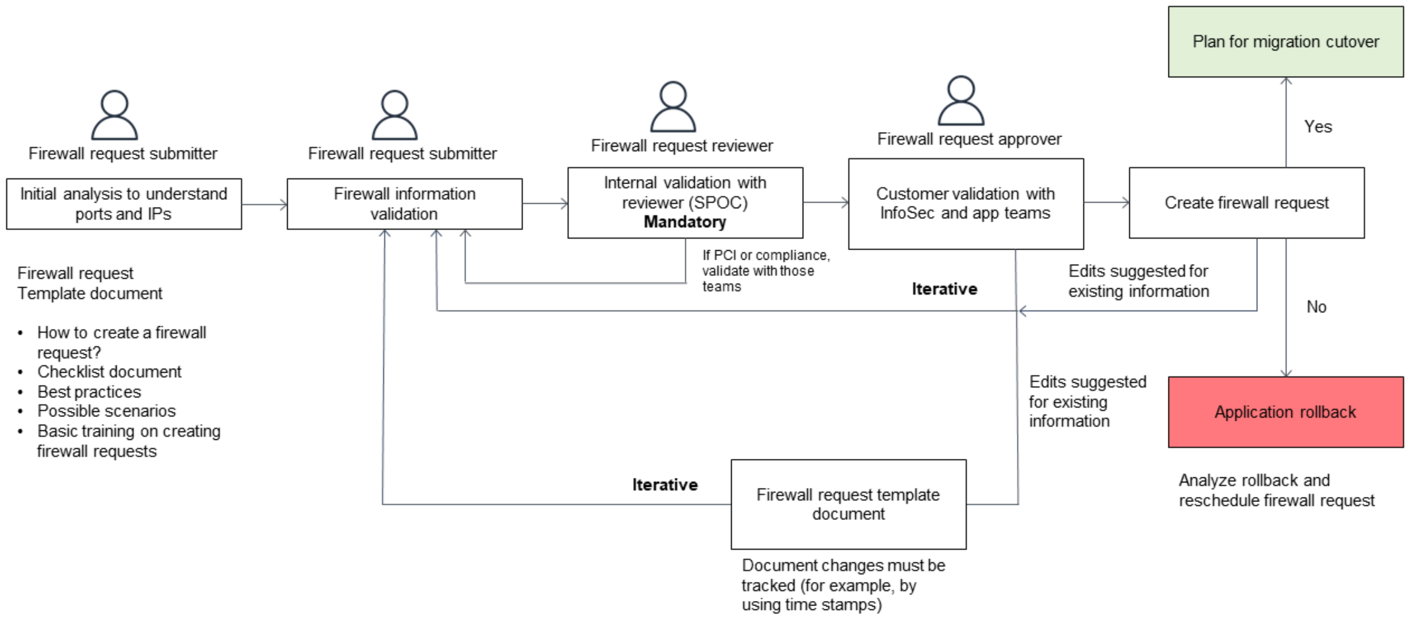
- This pattern describes a generic firewall request approval process. Requirements can vary for individual organizations.
- Make sure that you track changes to your firewall request document.

The following table shows the use cases for this pattern.

Does your organization have an existing firewall approval process?	Does your organization have an existing firewall request form?	Suggested action
Yes	Yes	Collaborate with AWS consultants or your migration specialists to implement your organization's process.
No	Yes	Use this pattern's firewall approval process. Use either an AWS consultant or a migration specialist from your organization to submit the firewall request blanket approval form.
No	No	Use this pattern's firewall approval process. Use either an AWS consultant or a migration specialist from your organization to submit the firewall request blanket approval form.

Architecture

The following diagram shows the steps for the firewall request approval process.



Tools

You can use scanner tools such as Palo Alto Networks or SolarWinds to analyze and validate firewalls and IP addresses.

Epics

Analyze the firewall request

Task	Description	Skills required
Analyze the ports and IP addresses.	The firewall request submitter completes an initial analysis to understand the required firewall ports and IP addresses. After this is complete, they request that your InfoSec team open the required ports and map the IP addresses.	AWS Cloud engineer, migration specialist

Validate the firewall request

Task	Description	Skills required
Validate the firewall informati on.	The AWS Cloud engineer schedules a meeting with your InfoSec team. During this meeting, the engineer examines and validates the firewall request information. Typically, the firewall request submitter is the same person as the firewall requester. This validation phase can become iterative based on the feedback given by the approver if anything is observed or recommended.	AWS Cloud engineer, migration specialist
Update the firewall request document.	After the InfoSec team shares their feedback, the firewall request document is edited, saved, and re-uploaded. This document is updated after each iteration. We recommend that you store this document in a version-controlled storage folder. This means that all changes are tracked and correctly applied.	AWS Cloud engineer, migration specialist

Submit the firewall request

Task	Description	Skills required
Submit the firewall request.	After the firewall request approver has approved the firewall blanket approval request, the AWS Cloud engineer submits the firewall request. The request specifies the ports that must be open and IP addresses that are required to map and update the AWS account. You can make suggestions or provide feedback after the firewall request is submitted . We recommend that you automate this feedback process and send any edits through a defined workflow mechanism.	AWS Cloud engineer, migration specialist

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Ingest and migrate EC2 Windows instances into an AWS Managed Services account

Anil Kunapareddy and Venkatramana Chintha, Amazon Web Services

Summary

This pattern explains the step-by-step process of migrating and ingesting Amazon Elastic Compute Cloud (Amazon EC2) Windows instances into an Amazon Web Services (AWS) Managed Services (AMS) account. AMS can help you manage the instance more efficiently and securely. AMS provides operational flexibility, enhances security and compliance, and helps you optimize capacity and reduce costs.

This pattern starts with an EC2 Windows instance that you have migrated to a staging subnet in your AMS account. A variety of migration services and tools are available to perform this task, such as AWS Application Migration Service.

To make a change to your AMS-managed environment, you create and submit a request for change (RFC) for a particular operation or action. Using an AMS workload ingest (WIGS) RFC, you ingest the instance into the AMS account and create a custom Amazon Machine Image (AMI). You then create the AMS-managed EC2 instance by submitting another RFC to create an EC2 stack. For more information, see AMS Workload Ingest in the AMS documentation.

Prerequisites and limitations

Prerequisites

- An active, AMS-managed AWS account
- An existing landing zone
- Permissions to make changes in the AMS-managed VPC
- An Amazon EC2 Windows instance in a staging subnet in your AMS account
- Completion of the general prerequisites for migrating workloads using AMS WIGS
- Completion of the Windows prerequisites for migrating workloads using AMS WIGS

Limitations

- This pattern is for EC2 instances operating Windows Server. This pattern doesn't apply to instances running other operating systems, such as Linux.

Architecture

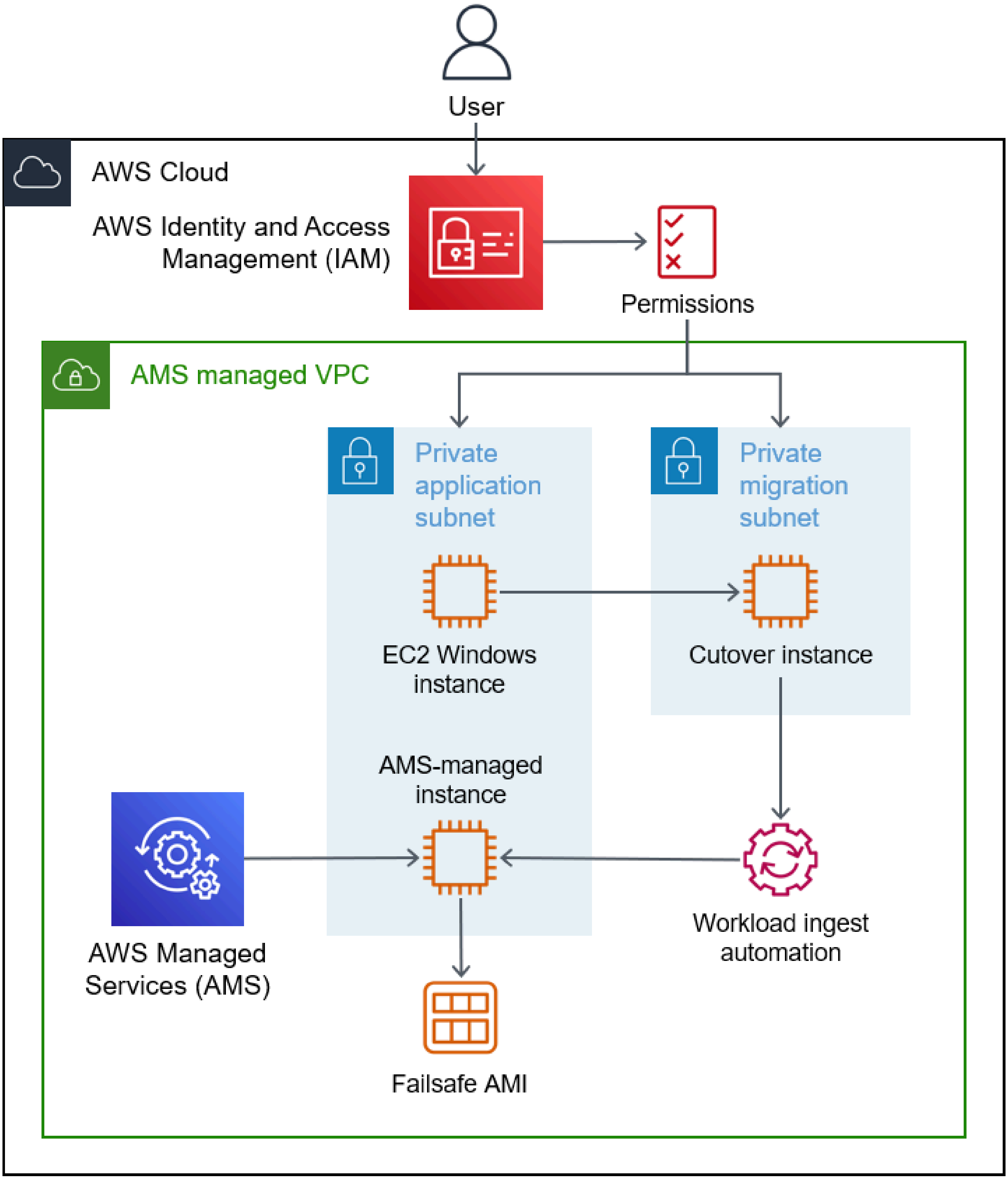
Source technology stack

Amazon EC2 Windows instance in a staging subnet in your AMS account

Target technology stack

Amazon EC2 Windows instance managed by AWS Managed Services (AMS)

Target architecture


User
AWS Cloud
AWS Identity and Access Management (IAM)
Permissions
AMS managed VPC
Private application subnet
Private migration subnet
EC2 Windows instance
Cutover instance
AMS-managed instance
AWS Managed Services (AMS)
Workload ingest automation
Failsafe AMI

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Managed Services (AMS) helps you operate more efficiently and securely by providing ongoing management of your AWS infrastructure, including monitoring, incident management, security guidance, patch support, and backup for AWS workloads.

Other services

- PowerShell is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS.

Epics

Configure settings on the instance

Task	Description	Skills required
Change the DNS Client settings.	1. On the source EC2 instance, open Command Prompt as an administr ator, type `gpedit.msc` , and then press **Enter**. 2. In the Local Group Policy Editor, navigate to **Computer Configuration**, **Administrative Templates** ,**Network**, **DNS Client**. 3. For **Primary DNS suffix**, choose **Not configured**.	Migration engineer

Task	Description	Skills required
	4. For **Primary DNS suffix devolution**, choose **Not configured**.	
Change the Windows Update settings.	1. In the Local Group Policy Editor, navigate to **Computer Configuration**, **Administrative Templates**, **Windows Components**, **Windows Update**. 2. For **Specify intranet Microsoft update service location**, choose **Not configured**. 3. For **Configure Automatic Updates**, choose **Not configured**. 4. For **Automatic Updates detection frequency**, choose **Not configured**. 5. Close the Local Group Policy Editor.	Migration engineer
Enable the firewall.	1. On the source EC2 instance, open Command Prompt as an administrator, type `services.msc` , and then press **Enter**. 2. In Windows Services, enable **Firewall**. 3. Close Windows Services.	Migration engineer

Prepare the instance for AMS WIGS

Task	Description	Skills required
Clean up and prepare the instance.	1. Using a bastion host and local credentials, create a Remote Desktop Protocol (RDP) connection to the EC2 instance in the staging subnet. 2. Remove all legacy software, antivirus software, and backup solutions that aren't required in AMS.	Migration engineer

Repair the sppnp.dll file.	1. Go to `C:\Windows\System32\sppnp.dll` .
	2. Rename `sppnp.dll` to `sppnp_old.dll` .
	3. Using PowerShell and administrator credentials, enter the following commands:
	`dism /online /cleanup-image /restorehealth`
	`sfc /scannow`
	4. Restart the EC2 Windows instance.
		Migration engineer

Run the pre-WIG validation script.	1. Download the Windows WIGS Pre-ingestion Validation zip file (`windows-prewings-`	Migration engineer

Task	Description	Skills required
	`valication.zip`) from [Migrating workloads : Windows pre-ingestion validation]() in the AMS documentation. 2. Run the Windows pre-WIG validation script and verify the results. 3. If the validation fails, fix the issue, and rerun the validation script until the validation succeeds.	
Create the failsafe AMI.	After the pre-WIG validation passes, create a pre-ingestion AMI as follows: 1. Choose **Deployment**, **Advanced stack components**, **AMI**, **Create**. 2. During creation, add a tag `Key=Name, Value=APPLICATION-ID_IngestReady` . 3. Wait until AMI is created before proceeding. For more information, see [AMI | Create]() in the AMS documentation.	Migration engineer

Ingest and validate the instance

Task	Description	Skills required
Submit the RFC to create the workload ingest stack.	Submit a request for change (RFC) to start the AMS WIGS. For instructions, see [Workload Ingest Stack: Creating](#) in the AMS documentation. This starts the workload ingestion and installs all the software required by AMS, including backup tools, Amazon EC2 management software, and antivirus software.	Migration engineer

Validate successful migration.	After the workload ingestion is complete, you can see the AMS-managed instance and AMS-ingested AMI.
	
	1. Log in to the AMS-manag ed instance with domain credentials.
	2. Validate the domain joining as follows:
	  a. In Windows Explorer, right-click **This PC**, and then choose **Properties**.
	  b. In the Device Specifica tion section, confirm that the domain appears in the **Full device name**.
	3. Validate the source and target disk drives.
		Migration engineer

Launch the instance in the target AMS account

Task	Description	Skills required
Submit the RFC to create an EC2 stack.	1. Using the AMS-ingested AMI of the Windows instance, prepare an RFC for an EC2 stack according to the instructions in [Create EC2 stack instance](#) in AMS documentation. In the EC2 stack RFC, provide all the parameters, including the server name, tags, target VPC, target subnet, instance type, target security groups, ingestion AMI, and role. 2. Submit the RFC for the EC2 stack, and then wait for the instance to be successfully created.	Migration engineer

Related resources

AWS Prescriptive Guidance

- Automate pre-workload ingestion activities for AWS Managed Services on Windows
- Automatically create an RFC in AMS using Python

AMS documentation

- AMS Workload Ingest
- How Migration Changes Your Resource
- Migrating Workloads: Standard Process

Marketing resources

- AWS Managed Services
- AWS Managed Services FAQs
- AWS Managed Services Resources
- AWS Managed Services Features

Migrate a Couchbase Server database to Amazon EC2

Subhani Shaik, Amazon Web Services

Summary

This pattern describes how you can migrate Couchbase Server from an on-premises environment to Amazon Elastic Compute Cloud (Amazon EC2) on AWS.

Couchbase Server is a distributed NoSQL (JSON document) database that provides relational database capabilities. Migrating a Couchbase Server database to AWS can provide increased scalability, improved performance, cost efficiency, enhanced security, simplified management, and global reach, which can benefit applications that require high availability and low latency data access. You also gain access to advanced features through AWS managed services.

Couchbase Server on AWS provides the following key features:

- Memory-first architecture
- High availability, disaster recovery, and load balancing
- Multi-master, multi-Region deployment for optimal performance

For more information about key benefits, see the Additional information section and the Couchbase website.

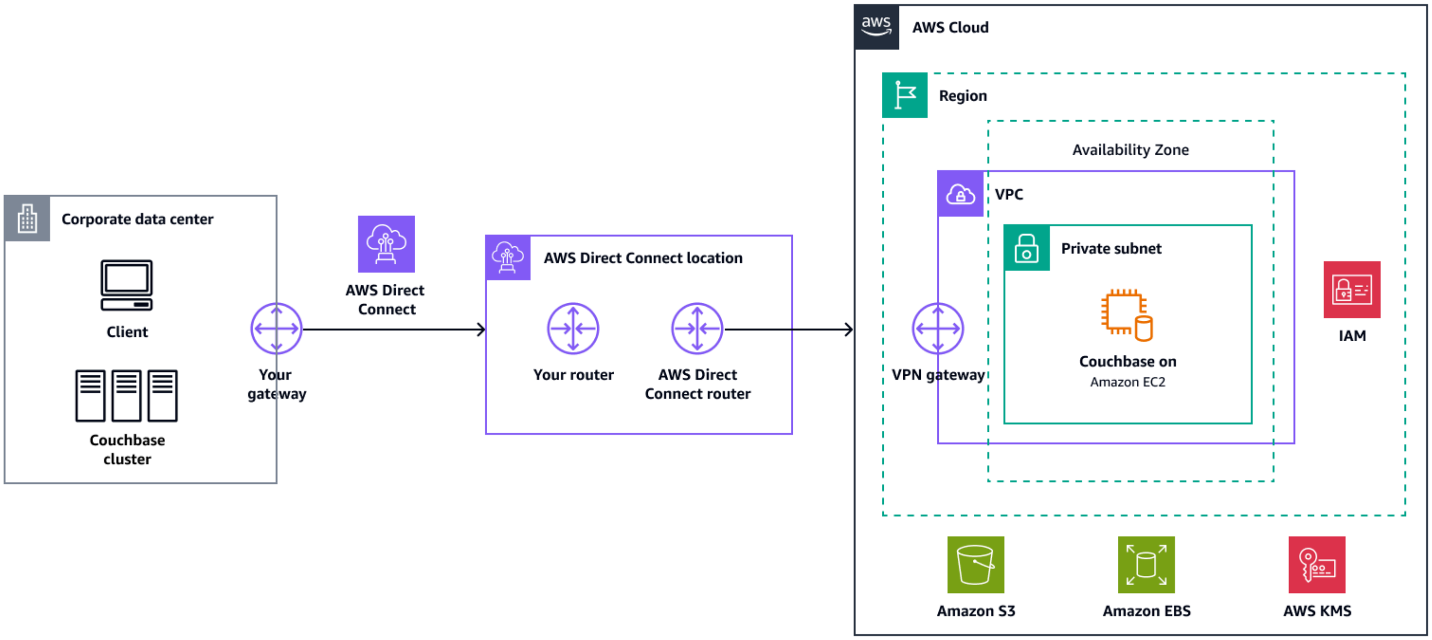
Prerequisites and limitations

Prerequisites

- An active AWS account with a virtual private cloud (VPC), two Availability Zones, private subnets, and a security group. For instructions, see Create a VPC in the Amazon Virtual Private Cloud (Amazon VPC) documentation.
- Connectivity enabled between source and target environments. For information about the TCX ports used by Couchbase Server, see the Couchbase documentation.

Architecture

The following diagram shows the high-level architecture for migrating Couchbase Server to AWS.



From the on-premises Couchbase cluster, data moves through a customer gateway by using [AWS Direct Connect](). The data passes through a router and an AWS Direct Connect route and reaches the VPC through an [AWS Virtual Private Network (AWS VPN)]() gateway. The VPC contains an EC2 instance that is running Couchbase Server. The AWS infrastructure also includes [AWS Identity and Access Management (IAM)]() for access control, [AWS Key Management Service (AWS KMS)]() for data encryption, [Amazon Elastic Block Store (Amazon EBS)]() for block storage, and [Amazon Simple Storage Service (Amazon S3)]() for data storage.

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Direct Connect links your internal network to a AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.

Best practices

- Installing and configuring Couchbase on different operating platforms
- Best practices for deploying Couchbase Server on AWS
- Creating a Couchbase cluster

- Performance best practices for Couchbase applications
- Security best practices for Couchbase Server
- Storage best practices for Couchbase Server databases

Epics

Deploy an Amazon EC2 instance for Couchbase Server

Task	Description	Skills required
Open the Amazon EC2 console.	Sign in to the AWS Management Console and open the Amazon EC2 console.	DevOps engineer, Couchbase administrator
Deploy an Amazon EC2 instance.	Launch an EC2 instance that matches the on-premises Couchbase Server configura tions. For more informati on about how to deploy an EC2 instance, see Launch an Amazon EC2 instance in the Amazon EC2 documentation.	DevOps engineer, Couchbase administrator

Install and configure Couchbase Server on Amazon EC2

Task	Description	Skills required
Install a Couchbase cluster.	Review Couchbase Server deployment guidelines before you install Couchbase Server on Amazon EC2. To install Couchbase Server, see the Couchbase Server documentation	Couchbase administrator

Task	Description	Skills required
Configure the cluster.	To configure the cluster, see Cluster Configuration Options in the Couchbase documentation.	Couchbase administrator

Add a new node and rebalance the Couchbase cluster

Task	Description	Skills required
Add a node for the EC2 instance.	Add the newly deployed EC2 instance that has Couchbase installed to the existing on-premises cluster. For instructions, see Add a Node and Rebalance in the Couchbase Server documentation.	Couchbase administrator
Rebalance the cluster.	The rebalancing process makes the newly added node with the EC2 instance an active member of the Couchbase cluster. For instructions, see Add a Node and Rebalance in the Couchbase Server documentation	Couchbase administrator

Reconfigure connections

Task	Description	Skills required
Remove the on-premises nodes and rebalance.	You can now remove the on-premises nodes from the cluster. After removing the	Couchbase administrator

Task	Description	Skills required
	nodes, follow the rebalance process to redistribute data, indexes, event processing, and query processing among available nodes in the cluster. For instructions, see Remove a Node and Rebalance in the Couchbase Server documenta tion.	
Update connection parameter s.	Update your application's connection parameters to use the new Amazon EC2 IP address, so your application can connect to the new node.	Couchbase application developer

Related resources

- Couchbase Server Services
- Deploy Couchbase Server Using AWS Marketplace
- Connect to Couchbase Server
- Manage Buckets
- Cross Data Center Replication (XDCR)
- Couchbase Inc. License Agreement

Additional information

Key benefits

Migrating your Couchbase database to AWS provides the following advantages:

Scalability. You can scale your Couchbase cluster up or down based on demand without having to manage physical hardware, so you can easily accommodate fluctuating data volumes and application usage. AWS provides:

- Vertical and horizontal scaling options
- Global deployment capabilities
- Load balancing across AWS Regions
- Database scaling solutions
- Content delivery optimization

Performance optimization. AWS provides a high-performance network infrastructure and optimized instance types to ensure fast data access and low latency for your Couchbase database.

- High performance computing (HPC) options
- Global content delivery through Amazon CloudFront
- Multiple storage options
- Advanced database services, including Amazon Relational Database Service (Amazon RDS) and Amazon DynamoDB
- Low-latency connections with AWS Direct Connect

Cost optimization. Select the appropriate instance type and configuration to balance performance and cost based on your workload. Pay only for the resources you use. This can potentially reduce your operational costs by eliminating the need to manage on-premises hardware and taking advantage of AWS Cloud economies of scale.

- Reserved instances can help you plan ahead and reduce your costs substantially when you use Couchbase on AWS.
- Automatic scaling prevents over-provisioning and helps you optimize your utilization and cost efficiencies.

Enhanced security. Benefit from the robust security features on AWS, such as data encryption, access controls, and security groups to help protect the sensitive data you store in Couchbase. Additional benefits:

- The AWS Shared Responsibility Model clearly differentiates between the security *of* the cloud (AWS responsibility) and security *in* the cloud (customer responsibility).
- AWS compliance supports major security standards.
- AWS provides advanced encryption options.

- AWS Identity and Access Management (IAM) helps you manage secure access to your resources.

Simplified management. AWS provides managed services for Couchbase, so you can focus on application development instead of managing the underlying infrastructure.

Global reach. You can deploy your Couchbase cluster across multiple AWS Regions to achieve low latency for users around the world. You can deploy your databases entirely in the cloud or in a hybrid environment. You can safeguard your data with built-in enterprise-grade security and fast, efficient bidirectional synchronization of data from the edge to the cloud. At the same time, you can simplify development with a consistent programming model for building web and mobile apps.

Business continuity:

- **Data backup and recovery**. In case of an issue, you can use AWS Backup to ensure data resiliency and easy recovery. For disaster recovery options, see the AWS Well-Architected Framework documentation.
- **Couchbase multi-Region deployment**: To deploy a Couchbase database in a multi-Region AWS environment, you can subscribe to Couchbase Server in AWS Marketplace, use AWS CloudFormationtemplates to create separate Couchbase clusters in each Region, and then configure cross-Region replication to synchronize data across Regions. This configuration ensures high availability and geographic redundancy across multiple Regions. For more information, see Deploy Couchbase Server Using AWS Marketplace in the Couchbase documentation.

Infrastructure agility:

- Rapid resource provisioning and deprovisioning
- Global infrastructure reach
- Automatic scaling based on demand
- Infrastructure as Code (IaC) for consistent deployments
- Multiple instance types that are optimized for different workloads

Innovation enablement:

- Access to the latest technologies, including AI/ML, IoT, and analytics
- Managed services, which reduce operational overhead
- Modern application development practices

- Serverless computing options

Operational excellence:

- Centralized monitoring and logging
- Automated resource management
- Predictive maintenance capabilities
- Enhanced visibility into resource usage
- Streamlined deployment processes

Modernization opportunities:

- Microservices architecture
- DevOps practices implementation
- Cloud-native application development
- Legacy application modernization

Competitive advantages:

- Faster time to market
- Improved customer experience
- Data-driven decision-making
- Enhanced business intelligence

Migrate Db2 for LUW to Amazon EC2 by using log shipping to reduce outage time

Feng Cai, Ambarish Satarkar, and Saurabh Sharma, Amazon Web Services

Summary

When customers migrate their IBM Db2 for LUW (Linux, UNIX, and Windows) workloads to Amazon Web Services (AWS), using Amazon Elastic Compute Cloud (Amazon EC2) with the Bring Your Own License (BYOL) model is the fastest way. However, migrating large amounts of data from on-premises Db2 into AWS can be a challenge, especially when the outage window is short. Many customers try to set the outage window to less than 30 minutes, which leaves little time for the database itself.

This pattern covers how to accomplish a Db2 migration with a short outage window by using transaction log shipping. This approach applies to Db2 on a little-endian Linux platform.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Db2 instance running on EC2 instance that matches the on-premises file system layouts
- An Amazon Simple Storage Service (Amazon S3) bucket accessible to the EC2 instance
- An AWS Identity and Access Management (IAM) policy and role to make programmatic calls to Amazon S3
- Synchronized time zone and system clocks on Amazon EC2 and the on-premises server
- The on-premises network connected to AWS through AWS Site-to-Site VPN or AWS Direct Connect

Limitations

- The Db2 on-premises instance and Amazon EC2 must be on the same platform family.
- The Db2 on-premises workload must be logged. To block any unlogged transaction, set `blocknonlogged=yes` in the database configuration.

Product versions

- Db2 for LUW version 11.5.9 and later

Architecture

Source technology stack

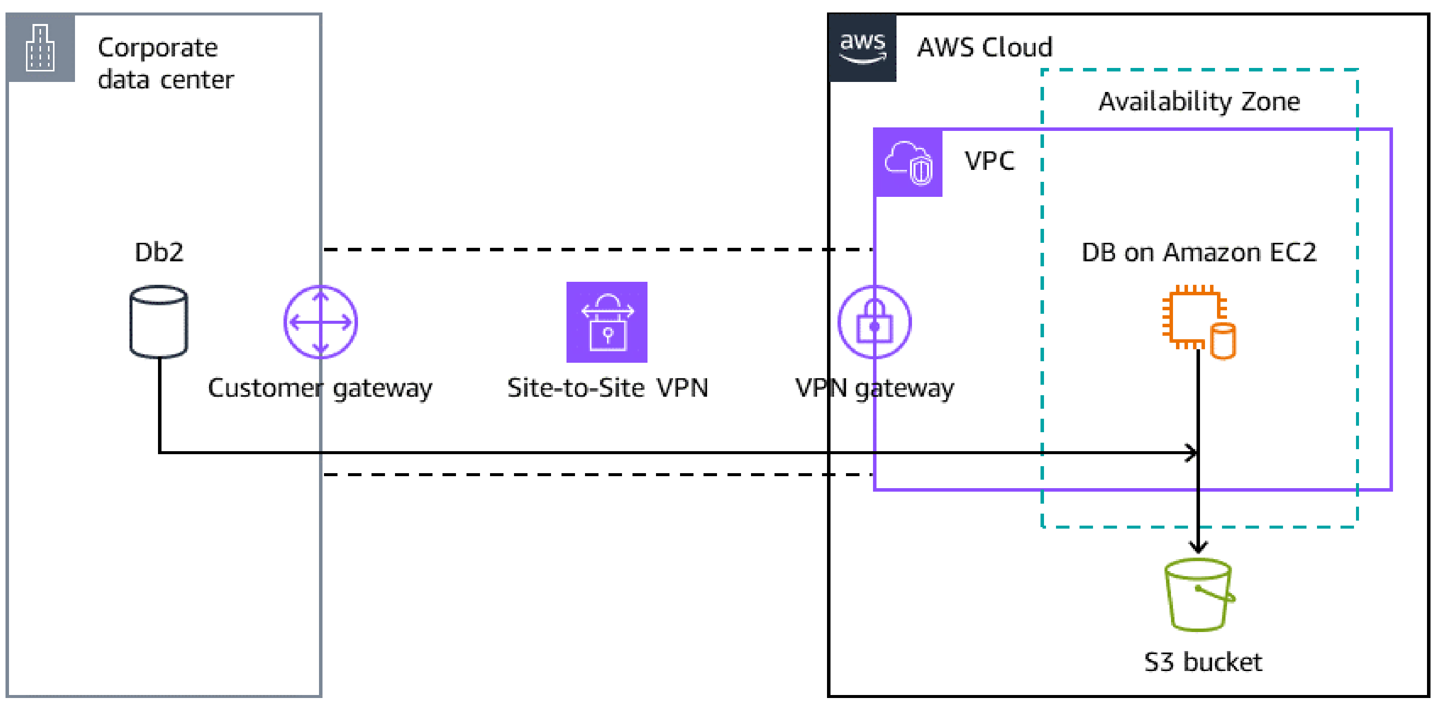
- Db2 on Linux x86_64

Target technology stack

- Amazon EBS
- Amazon EC2
- AWS Identity and Access Management (IAM)
- Amazon S3
- AWS Site-to-Site VPN or Direct Connect

Target architecture

The following diagram shows one Db2 instance running on-premises with a virtual private network (VPN) connection to Db2 on Amazon EC2. The dotted lines represent the VPN tunnel between your data center and the AWS Cloud.



Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Direct Connect links your internal network to a Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Site-to-Site VPN helps you pass traffic between instances that you launch on AWS and your own remote network.

Other tools

- db2cli is the Db2 interactive CLI command.

Best practices

- On the target database, use gateway endpoints for Amazon S3 to access the database backup image and log files in Amazon S3.
- On the source database, use AWS PrivateLink for Amazon S3 to send the database backup image and log files to Amazon S3.

Epics

Set environment variables

Task	Description	Skills required
Set environment variables.	This pattern uses the following names: • Instance name: `db2inst1` • Database name: `SAMPLE` You can change them to fit your environment.	DBA

Configure the on-premises Db2 server

Task	Description	Skills required

Set up the AWS CLI.	To download and install the latest version of the AWS CLI, run the following commands:
	`$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
	`unzip awscliv2.zip`
	`sudo ./aws/install`
		Linux administrator

Set up a local destination for Db2 archive logs.	To keep the target database on Amazon EC2 in sync with the on-premises source database, the latest transacti on logs need be retrieved from the source.	DBA

Task	Description	Skills required

	In this setup, `/db2logs` is set by `LOGARCHMETH2` on the source as a staging area. The archived logs in this directory will be synced into Amazon S3 and accessed by Db2 on Amazon EC2. The pattern uses `LOGARCHMETH2` because `LOGARCHMETH1` might have been configured to use a third-party vendor tool that AWS CLI command cannot access. To retrieve the logs, run the following command:
	`db2 connect to sample`
	`db2 update db cfg for SAMPLE using LOGARCHMETH2 disk:/db2logs`	

Run an online database backup.	Run an online database backup, and save it to the local backup file system: `db2 backup db sample online to /backup`	DBA

Set up the S3 bucket and IAM policy

Task	Description	Skills required
Create an S3 bucket.	Create an S3 bucket for the on-premises server to send the backup Db2 images and	AWS systems administrator

Task	Description	Skills required
	log files to on AWS. The bucket will also be accessed by Amazon EC2: `aws s3api create-bucket --bucket logshipmig-db2 --region us-east-1`	

Task	Description	Skills required

Create an IAM policy.	The `db2bucket.json` file contains the IAM policy to access the Amazon S3 bucket:
	``{
	    "Version":
	 "2012-10-17",
	    "Statement": [
	        {
	            "Effect":
	 "Allow",
	            "Action": [
	
	 "kms:GenerateDataK
	ey",
	
	 "kms:Decrypt",
	
	 "s3:PutObject",
	
	 "s3:GetObject",
	
	 "s3:AbortMultipart
	Upload",
	
	 "s3:ListBucket",
	
	 "s3:DeleteObject",
	
	 "s3:GetObjectVersi
	on",
	
	 "s3:ListMultipartU
	ploadParts"
	            ],
	            "Resource":
	 [
	
	 "arn:aws:s3:::logs
	hipmig-db2/*",```
		AWS administrator, AWS systems administrator

Task	Description	Skills required

	<pre>"arn:aws:s3:::logs
	hipmig-db2"
	            ]
	        }
	    ]
	}</pre> To create the policy, use the following AWS CLI command: <pre>aws iam create-policy \\
	      --policy-name
	 db2s3policy \\
	      --policy-document
	 file://db2bucket.j
	son</pre> The JSON output shows the Amazon Resource Name (ARN) for the policy, where `aws_account_id` represents your account ID: <pre>"Arn": "arn:aws:
	iam::aws_account_i
	d:policy/db2s3policy"</pre>	

Task	Description	Skills required
Attach the IAM policy to the IAM role used by the EC2 instance.	In most AWS environments, a running EC2 instance has an IAM Role set by your systems administrator. If the IAM role is not set, create the role and choose **Modify IAM role** on the EC2 console to associate the role with the EC2 instance that hosts the Db2 database. Attach the IAM policy to the IAM role with the policy ARN: `aws iam attach-role-policy \ --policy-arn "arn:aws:iam::aws_account_id:policy/db2s3policy" \ --role-name db2s3role` After the policy is attached, any EC2 instance associated with the IAM role can access the S3 bucket.	AWS administrator, AWS systems administrator

Send the source database backup image and log files to Amazon S3

Task	Description	Skills required
Configure the AWS CLI on the on-premises Db2 server.	Configure the AWS CLI with the `Access Key ID` and `Secret Access Key` generated in the earlier step:	AWS administrator, AWS systems administrator

Task	Description	Skills required

	<pre>$ aws configure
	AWS Access Key ID
	 [None]: *************
	AWS Secret Access Key
	 [None]: *********
	******************
	Default region name
	 [None]: us-east-1
	Default output format
	 [None]: json</pre>	

Send the backup image to Amazon S3.	Earlier, an online database backup was saved to the `/backup` local directory. To send that backup image to the S3 bucket, run the following command:
	<pre>aws s3 sync /backup
	 s3://logshipmig-db2/
	SAMPLE_backup</pre>
		AWS administrator, Migration engineer

Task	Description	Skills required
Send the Db2 archive logs to Amazon S3.	Sync the on-premises Db2 archive logs with the S3 bucket that can be accessed by the target Db2 instance on Amazon EC2: `aws s3 sync /db2logs s3://logshipmig-db2/SAMPLE_LOG` Run this command periodically by using cron or other scheduling tools. The frequency depends on how often the source database archives transaction log files.	AWS administrator, Migration engineer

Connect Db2 on Amazon EC2 to Amazon S3 and start the database sync

Task	Description	Skills required
Create a PKCS12 keystore.	Db2 uses a Public-Key Cryptography Standards (PKCS) encryption keystore to keep the AWS access key secure. Create a keystore and configure the source Db2 instance to use it: `gsk8capicmd_64 -keydb -create -db "/home/db2inst1/.keystore/db2s3.p12" -pw "<password>" -type pkcs12 -stash`	DBA

Task	Description	Skills required
	`db2 "update dbm cfg using keystore_location /home/db2inst1/.keystore/db2s3.p12 keystore_type pkcs12"`	

Create the Db2 storage access alias.	To create the [storage access alias](https://www.ibm.com/docs/en/db2/11.5?topic=commands-catalog-storage-access), use the following script syntax:
	
	`db2 "catalog storage access alias <alias_name> vendor S3 server <S3 endpoint> container '<bucket_name>'"`
	
	For example, your script might look like the following:
	
	`db2 "catalog storage access alias DB2AWSS3 vendor S3 server s3.us-east-1.amazonaws.com container 'logshipmig-db2'"`
		DBA

Task	Description	Skills required

Set the staging area.	By default, Db2 uses `DB2_OBJECT_STORAGE_LOCAL_STAGING_PATH` as the staging area to upload and download files to and from Amazon S3. The default path is `sqllib/tmp/RemoteStorage.xxxx` under the instance home directory, with `xxxx` referring to the Db2 partition number. Note that the staging area must have enough capacity to hold the backup images and log files. You can use the registry to point the staging area into a different directory.
	
	We also recommend using `DB2_ENABLE_COS_SDK=ON`, `DB2_OBJECT_STORAGE_SETTINGS=EnableStreamingRestore`, and the link to the `awssdk` library to bypass the Amazon S3 staging area for database backup and restore:
	
	`#By root:`
	`cp -rp /home/db2inst1/sqllib/lib64/awssdk/RHEL/7.6/* /home/db2inst1/sqllib/lib64/`
	
	`#By db2 instance owner:`
		DBA

Task	Description	Skills required
	``` db2set DB2_OBJEC T_STORAGE_LOCAL_ST AGING_PATH=/db2stage db2set DB2_ENABL E_COS_SDK=ON Db2set DB2_OBJEC T_STORAGE_SETTINGS =EnableStreamingRe store db2stop db2start ```	
Restore the database from the backup image.	Restore the target database on Amazon EC2 from the backup image in the S3 bucket: ``` db2 restore db sample from DB2REMOTE:// DB2AWSS3/logshipmig- db2/SAMPLE_backup replace existing ```	DBA

Task	Description	Skills required

Roll forward the database.	After the restore is complete, the target database will be put into rollforward pending state. Configure `LOGARCHMETH1` and `LOGARCHMETH2` so that Db2 knows where to get the transaction log files:
	`db2 update db cfg for SAMPLE using LOGARCHMETH1 'DB2REMOTE://DB2AWSS3//SAMPLE_LOGS/'`
	`db2 update db cfg for SAMPLE using LOGARCHMETH2 OFF`
	Start database rollforward:
	`db2 ROLLFORWARD DATABASE sample to END OF LOGS`
	This command processes all log files that have been transferred to the S3 bucket. Run it periodically based on the frequency of the `s3 sync` command on the on-premises Db2 servers. For example, if `s3 sync` runs at each hour, and it takes 10 minutes to sync all the log files, set the command to run at 10 minutes after each hour.
		DBA

Bring Db2 on Amazon EC2 online during the cutover window

Task	Description	Skills required

Bring the target database online.	During the cutover window, do one of the following:
	
	• Put the on-premises database in `ADMIN MODE`, and run the `s3 sync` command to force the last transaction log to be archived.
	• Shut down the database.
	
	After the last transaction log is synced into Amazon S3, run the `ROLLFORWARD` command for the final time:
	
	`db2 rollforward DB sample to END OF LOGS`
	`db2 rollforward DB sample complete`
	
	`Rollforward Status`
	`....`
	`Rollforward status = not pending`
	`....`
	`DB20000I The ROLLFORWARD command completed successfully.`
	
	`db2 activate db sample`
	`DB20000I The ACTIVATE DATABASE command`
		DBA

Task	Description	Skills required
	`completed successfully.` Bring the target database online, and point the application connections to Db2 on Amazon EC2.	

Troubleshooting

Issue	Solution
If multiple databases have the same instance name and database name on different hosts (DEV, QA, PROD), backups and logs might go to the same subdirectory.	Use different S3 buckets for DEV, QA, and PROD, and add the hostname as subdirectory prefix to avoid confusion.
If there are multiple backup images in the same location, you will get the following error when you restore: `SQL2522N More than one backup file matches the time stamp value provided for the backed up database image.`	In the `restore` command, add the timestamp of the backup: `db2 restore db sample from DB2REMOTE://DB2AWSS3/logshipmig-db2/SAMPLE_backup taken at 20230628164042 replace existing`

Related resources

- Db2 backup and restore operations between different operating systems and hardware platforms
- Set up Db2 STORAGE ACCESS ALIAS and DB2REMOTE
- Db2 ROLLFORWARD command
- Db2 secondary log archive method

Migrate Db2 for LUW to Amazon EC2 with high availability disaster recovery

Feng Cai, Aruna Gangireddy, and Venkatesan Govindan, Amazon Web Services

Summary

When customers migrate their IBM Db2 LUW (Linux, UNIX, and Windows) workload to Amazon Web Services (AWS), using Amazon Elastic Compute Cloud (Amazon EC2) with the Bring Your Own License (BYOL) model is the fastest way. However, migrating large amounts of data from on-premises Db2 into AWS can be a challenge, especially when the outage window is short. Many customers try to set the outage window to less than 30 minutes, which leaves little time for the database itself.

This pattern covers how to accomplish a Db2 migration with a short outage window by using Db2 high availability disaster recovery (HADR). This approach applies to Db2 databases that are on the little-endian Linux platform and are not using Data Partitioning Feature (DPF).

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Db2 instance running on an Amazon EC2 instance that matches the on-premises file system layouts
- An Amazon Simple Storage Service (Amazon S3) bucket accessible to the EC2 instance
- An AWS Identity and Access Management (IAM) policy and role to make programmatic calls to Amazon S3
- Synchronized time zone and system clocks on Amazon EC2 and the on-premises server
- The on-premises network connected to AWS through AWS Site-to-Site VPN or AWS Direct Connect
- Communication between the on-premises server and Amazon EC2 on HADR ports

Limitations

- The Db2 on-premises instance and Amazon EC2 must be on the same platform family.
- HADR is not supported in a partitioned database environment.

- HADR doesn’t support the use of raw I/O (direct disk access) for database log files.
- HADR doesn’t support infinite logging.
- `LOGINDEXBUILD` must be set to `YES`, which will increase the log usage for rebuilding the index.
- The Db2 on-premises workload must be logged. Set `blocknonlogged=yes` in the database configuration to block any unlogged transactions.

Product versions

- Db2 for LUW version 11.5.9 and later

Architecture

Source technology stack

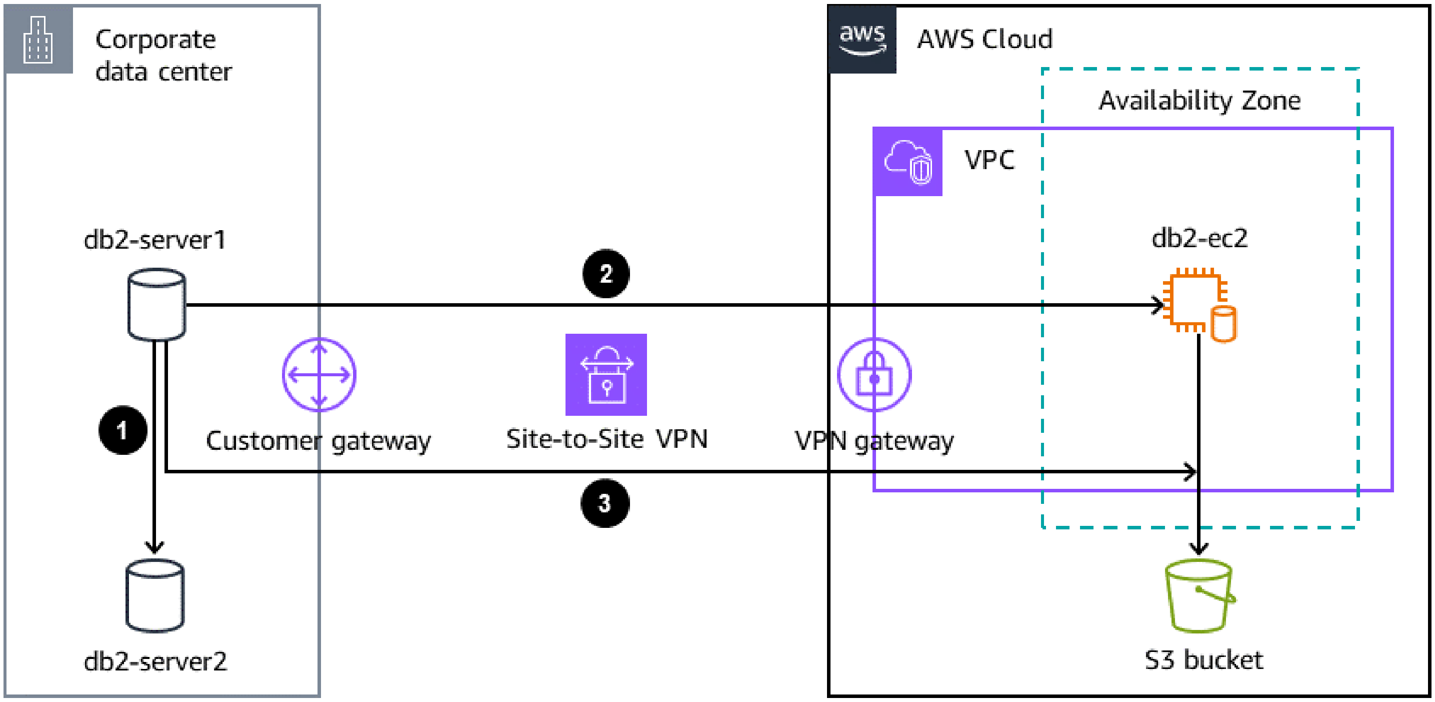
- Db2 on Linux x86_64

Target technology stack

- Amazon EC2
- AWS Identity and Access Management (IAM)
- Amazon S3
- AWS Site-to-Site VPN

Target architecture

In the following diagram, Db2 on premises is running on `db2-server1` as the primary. It has two HADR standby targets. One standby target is on premises and is optional. The other standby target, `db2-ec2`, is on Amazon EC2. After the database is cut over to AWS, `db2-ec2` becomes the primary.



1. Logs are streamed from the primary on-premises database to the standby on-premises database.
2. Using Db2 HADR, logs are streamed from the primary on-premises database through Site-to-Site VPN to Db2 on Amazon EC2.
3. Db2 backup and archive logs are sent from the primary on-premises database to the S3 bucket on AWS.

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Direct Connect links your internal network to a Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Site-to-Site VPN helps you pass traffic between instances that you launch on AWS and your own remote network.

Other tools

- db2cli is the Db2 interactive CLI command.

Best practices

- On the target database, use gateway endpoints for Amazon S3 to access the database backup image and log files in Amazon S3.
- On the source database, use AWS PrivateLink for Amazon S3 to send the database backup image and log files to Amazon S3.

Epics

Set environment variables

Task	Description	Skills required
Set environment variables.	This pattern uses the following names and ports: 1. Db2 on-premises hostname: `db2-server1` 2. HADR standby hostname: `db2-server2` (if HADR is currently running on premises) 3. Amazon EC2 hostname: `db2-ec2` 4. Instance name: `db2inst1` 5. Database name: `SAMPLE` 6. HADR ports:	DBA

Task	Description	Skills required
	• `db2-server1: 50010` • `db2-server2: 50011` • `db2-ec2: 50012` You can change them to fit your environment.	

Configure the on-premises Db2 server

Task	Description	Skills required

Set up AWS CLI.	To download and install the latest version of AWS CLI, run the following commands:
	`$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"`
	`unzip awscliv2.zip`
	`sudo ./aws/install`
		Linux administrator

Set up a local destination for Db2 archive logs.	Conditions such as heavy update batch jobs and network slowdowns can cause the HADR standby server to have a lag. To catch up, the standby server needs the transaction logs from the primary server. The sequence of places to request logs is the following:	DBA

Task	Description	Skills required

	• The active log directory on the primary server
	• The `LOGARCHMETH1` or `LOGARCHMETH2` location on the standby server
	• The `LOGARCHMETH1` or `LOGARCHMETH2` location on the primary server
	
	In this setup, `/db2logs` is set by `LOGARCHMETH2` on the source as a staging area. The archived logs in this directory will be synced into Amazon S3 and accessed by Db2 on Amazon EC2. The pattern uses `LOGARCHMETH2` because `LOGARCHMETH1` might have been configured to use a third-party vendor tool that the AWS CLI command cannot access:
	
	`db2 connect to sample`
	`db2 update db cfg for SAMPLE using LOGARCHMETH2 disk:/db2logs`	

Run an online database backup.	Run an online database backup, and save it to the local backup file system: `db2 backup db sample online to /backup`	DBA

Set up the S3 bucket and IAM policy

Task	Description	Skills required
Create an S3 bucket.	Create an S3 bucket for the on-premises server to send the backup Db2 images and log files to on AWS. The bucket will be accessed by Amazon EC2: ```aws s3api create-bucket --bucket hadrmig-db2 --region us-east-1```	AWS administrator
Create an IAM policy.	The `db2bucket.json` file contains the IAM policy for accessing the S3 bucket: ```{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["kms:GenerateDataKey", "kms:Decrypt", "s3:PutObject", "s3:GetObject", "s3:AbortMultipartUpload", "s3:ListBucket",```	AWS administrator, AWS systems administrator

Task	Description	Skills required

	``
	 "s3:DeleteObject",
	
	 "s3:GetObjectVersi
	on",
	
	 "s3:ListMultipartU
	ploadParts"
	            ],
	            "Resource":
	 [
	
	 "arn:aws:s3:::hadr
	mig-db2/*",
	
	 "arn:aws:s3:::hadr
	mig-db2"
	            ]
	        }
	    ]
	}
	``
	To create the policy, use the following AWS CLI command:
	``
	aws iam create-policy \
	      --policy-name
	 db2s3hapolicy \
	      --policy-document
	 file://db2bucket.j
	son
	``
	The JSON output shows the Amazon Resource Name (ARN) for the policy, where `aws_account_id` represents your account ID:	

Task	Description	Skills required
	```"Arn": "arn:aws:iam::aws_account_id:policy/db2s3hapolicy"```	
Attach the IAM policy to the IAM role.	Usually, the EC2 instance with Db2 running would have an IAM role assigned by the systems administrator. If no IAM role is assigned, you can choose **Modify IAM role** on the Amazon EC2 console. Attach the IAM policy to the IAM role associated with the EC2 instance. After the policy is attached, the EC2 instance can access the S3 bucket: ```aws iam attach-role-policy --policy-arn "arn:aws:iam::aws_account_id:policy/db2s3hapolicy" --role-name db2s3harole```	

Send the source database backup image and log files to Amazon S3

Task	Description	Skills required
Configure AWS CLI on the on-premises Db2 server.	Configure AWS CLI with the `Access Key ID` and `Secret Access Key` that you generated earlier:	AWS administrator, AWS systems administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>$ aws configure
AWS Access Key ID
 [None]: *************
AWS Secret Access Key
 [None]: *********
****
Default region name
 [None]: us-east-1
Default output format
 [None]: json</code></pre></td><td></td></tr>
<tr><td>Send the backup image to Amazon S3.</td><td>Earlier, an online database backup was saved to the <code>/backup</code> local directory. To send that backup image to the S3 bucket, run the following command:<pre><code>aws s3 sync /backup
 s3://hadrmig-db2/S
AMPLE_backup</code></pre></td><td>AWS administrator, AWS systems administrator</td></tr>
</table>

Task	Description	Skills required
Send the Db2 archive logs to Amazon S3.	Sync the on-premises Db2 archive logs with the Amazon S3 bucket that can be accessed by the target Db2 instance on Amazon EC2: `aws s3 sync /db2logs s3://hadrmig-db2/SAMPLE_LOGS` Run this command periodically by using cron or other scheduling tools. The frequency depends on how often the source database archives transaction log files.	

Connect Db2 on Amazon EC2 to Amazon S3 and start the initial database sync

Task	Description	Skills required
Create a PKCS12 keystore.	Db2 uses a Public-Key Cryptography Standards (PKCS) encryption keystore to keep the AWS access key secure. Create a keystore, and configure the source Db2 to use it: `gsk8capicmd_64 -keydb -create -db "/home/db2inst1/.keystore/db2s3.p12" -pw "<password>" -type pkcs12 -stash`	DBA

Task	Description	Skills required
	```db2 "update dbm cfg using keystore_location /home/db2inst1/.keystore/db2s3.p12 keystore_type pkcs12"```	

Create the Db2 storage access alias.	Db2 uses a storage access alias to access Amazon S3 directly with the `INGEST`, `LOAD`, `BACKUP DATABASE`, or `RESTORE DATABASE` commands.
	
	Because you assigned an IAM role to the EC2 instance, `USER` and `PASSWORD` are not required:
	
	`db2 "catalog storage access alias <alias_name> vendor S3 server <S3 endpoint> container '<bucket_name>'"`
	
	For example, your script might look like the following:
	
	`db2 "catalog storage access alias DB2AWSS3 vendor S3 server s3.us-east-1.amazonaws.com container 'hadrmig-db2'"`
		DBA

Task	Description	Skills required

Set the staging area.	We recommend using `DB2_ENABLE_COS_SDK =ON` , `DB2_OBJECT_STORAGE_SETTINGS =EnableStreamingRestore` , and the link to the `awssdk` library to bypass the Amazon S3 staging area for database backup and restore:
	
	`#By root:`
	`cp -rp /home/db2inst1/sqllib/lib64/awssdk/RHEL/7.6/* /home/db2inst1/sqllib/lib64/`
	
	`#By db2 instance owner:`
	`db2set DB2_OBJECT_STORAGE_LOCAL_STAGING_PATH=/db2stage`
	`db2set DB2_ENABLE_COS_SDK=ON`
	`db2set DB2_OBJECT_STORAGE_LOCAL_STAGING_PATH=/db2stage`
	`db2stop`
	`db2start`
		DBA

Task	Description	Skills required

Restore the database from the backup image.	Restore the target database on Amazon EC2 from the backup image in the S3 bucket:
	`db2 create db sample on /data1`
	`db2 restore db sample from DB2REMOTE:// DB2AWSS3/hadrmig-db2/ SAMPLE_backup replace existing`
		DBA

Set up HADR with no HADR on premises

Task	Description	Skills required
Configure the on-premises Db2 server as the primary.	Update the database configuration settings for HADR on `db2-server1` (the on-premises source) as the primary. Set `HADR_SYNCMODE` to `SUPERASYNC` mode, which has the shortest transaction response time: `db2 update db cfg for sample using HADR_LOCAL_HOST db2-server1 HADR_LOCAL_SVC 50010 HADR_REMOTE_HOST db2-ec2 HADR_REMOTE_SVC 50012 HADR_REMOTE_INST db2inst1 HADR_SYNCMODE`	DBA

Task	Description	Skills required
	SUPERASYNC DB20000 I The UPDATE DATABASE CONFIGURATION command completed successfu lly Some network delays between the on-premises data center and AWS are expected. (You can set a different HADR_SYNCMODE value based on network reliability. For more informati on, see the Related resources section).	
Change the target database log archive destination.	Change the target database log archive destination to match the Amazon EC2 environment: `db2 update db cfg for SAMPLE using LOGARCHMETH1 'DB2REMOTE://DB2AWSS3//SAMPLE_LOGS/' LOGARCHMETH2 OFF DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully`	DBA

Task	Description	Skills required
Configure HADR for Db2 on the Amazon EC2 server.	Update database configura tion for HADR on `db2-ec2` as standby: `db2 update db cfg for sample using HADR_LOCAL_HOST db2-ec2 HADR_LOCA L_SVC 50012 HADR_REMO TE_HOST db2-server1 HADR_REMOTE_SVC 50010 HADR_REMOTE_INST db2inst1 HADR_SYNC MODE SUPERASYN C DB20000I The UPDATE DATABASE CONFIGURATION command completed successfu lly`	DBA

Task	Description	Skills required
Verify HADR setup.	Verify the HADR parameters on the source and target Db2 servers. To verify the setup on `db2-server1` , run the following command: `db2 get db cfg for sample|grep HADR HADR database role = PRIMARY HADR local host name (HADR_LOCAL_HOST) = db2-server1 HADR local service name (HADR_LOCAL_SVC) = 50010 HADR remote host name (HADR_REMOTE_HOST) = db2-ec2 HADR remote service name (HADR_REMOTE_SVC) = 50012 HADR instance name of remote server (HADR_REMOTE_INST) = db2inst1 HADR timeout value (HADR_TIMEOUT) = 120 HADR target list (HADR_TARGET_LIST) = HADR log write synchronization mode`	DBA

Task	Description	Skills required

	``
	    (HADR_SYNCMODE) =
	 NEARSYNC
	 HADR spool log data
	 limit (4KB)
	 (HADR_SPOOL_LIMIT) =
	 AUTOMATIC(52000)
	 HADR log replay delay
	 (seconds) (HADR_REP
	LAY_DELAY) = 0
	 HADR peer window
	 duration (seconds)
	 (HADR_PEER_WINDOW) = 0
	 HADR SSL certifica
	te label
	 (HADR_SSL_LABEL) =
	 HADR SSL Hostname
	 Validation
	 (HADR_SSL_HOST_VAL) =
	 OFF
	``
	To verify the setup on db2-ec2, run the following command:
	``
	db2 get db cfg for
	 sample|grep HADR
	 HADR database role
	
	        = STANDBY
	 HADR local host name
	              (HADR_LOC
	AL_HOST) = db2-ec2
	 HADR local service
	 name
	 (HADR_LOCAL_SVC) =
	 50012
	 HADR remote host name
	             (HADR_REM
	OTE_HOST) = db2-serve
	r1
	``	

Task	Description	Skills required

	``
	 HADR remote service
	 name
	 (HADR_REMOTE_SVC) =
	 50010
	 HADR instance name
	 of remote server
	 (HADR_REMOTE_INST) =
	 db2inst1
	 HADR timeout value
	
	 (HADR_TIMEOUT) = 120
	 HADR target list
	               (HADR_TAR
	GET_LIST) =
	 HADR log write
	 synchronization mode
	     (HADR_SYNCMODE) =
	 SUPERASYNC
	 HADR spool log data
	 limit (4KB)
	 (HADR_SPOOL_LIMIT) =
	 AUTOMATIC(52000)
	 HADR log replay delay
	 (seconds) (HADR_REP
	LAY_DELAY) = 0
	 HADR peer window
	 duration (seconds)
	 (HADR_PEER_WINDOW) = 0
	 HADR SSL certifica
	te label
	 (HADR_SSL_LABEL) =
	 HADR SSL Hostname
	 Validation
	 (HADR_SSL_HOST_VAL) =
	 OFF
	``
	
	The `HADR_LOCAL_HOST` , `HADR_LOCAL_SVC` , `HADR_REMOTE_HOST` , and `HADR_REMOTE_SVC`	

Task	Description	Skills required
	parameters indicate the one primary and one standby HADR setup.	

Start up the Db2 HADR instance.	Start the Db2 HADR instance on the standby server `db2-ec2` first:
	`db2 start hadr on db sample as standby`
	`DB20000I  The START HADR ON DATABASE command completed successfully.`
	Start Db2 HADR on the primary (source) server `db2-server1` :
	`db2 start hadr on db sample as primary`
	`DB20000I  The START HADR ON DATABASE command completed successfully.`
	The HADR connection between Db2 on premises and on Amazon EC2 has now been successfully established. The Db2 primary server `db2-server1` starts streaming transaction log records to `db2-ec2` in real time.
		DBA

Set up HADR when HADR exists on premises

Task	Description	Skills required
Add Db2 on Amazon EC2 as an auxiliary standby.	If HADR is running on the on-premises Db2 instance, you can add Db2 on Amazon EC2 as an auxiliary standby using `HADR_TARGET_LIST` by running the following commands on `db2-ec2`: `db2 update db cfg for sample using HADR_LOCAL_HOST db2-ec2 HADR_LOCAL_SVC 50012 HADR_REMOTE_HOST db2-server1 HADR_REMOTE_SVC 50010 HADR_REMOTE_INST db2inst1 HADR_SYNCMODE SUPERASYNC DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully. db2 update db cfg for sample using HADR_TARGET_LIST "db2-server1:50010 |db2-server2:50011 " DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.`	DBA

Task	Description	Skills required
Add the auxiliary standby information to the on-premises servers.	Update `HADR_TARGET_LIST` on the two on-premises servers (primary and standby). On `db2-server1`, run the following code: `db2 update db cfg for sample using HADR_TARGET_LIST "db2-server2:50011|db2-ec2:50012" DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully. SQL1363W One or more of the parameters submitted for immediate modification were not changed dynamically. For these configuration parameters, the database must be shutdown and reactivated before the configuration parameter changes become effective.`  On `db2-server2`, run the following code:	DBA

Task	Description	Skills required
	`db2 update db cfg for sample using HADR_TARGET_LIST "db2-server1:50010|db2-ec2:50012" DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully. SQL1363W One or more of the parameters submitted for immediate modification were not changed dynamically. For these configuration parameters, the database must be shutdown and reactivated before the configuration parameter changes become effective.`	

Task	Description	Skills required
Verify HADR setup.	Verify the HADR parameters on the source and target Db2 servers. On `db2-server1` , run the following code: `db2 get db cfg for sample|grep HADR HADR database role = PRIMARY HADR local host name (HADR_LOCAL_HOST) = db2-server1 HADR local service name (HADR_LOCAL_SVC) = 50010 HADR remote host name (HADR_REMOTE_HOST) = db2-server2 HADR remote service name (HADR_REMOTE_SVC) = 50011 HADR instance name of remote server (HADR_REMOTE_INST) = db2inst1 HADR timeout value (HADR_TIMEOUT) = 120 HADR target list (HADR_TARGET_LIST) = db2-server2:50011|db2-ec2:50012`	

Task	Description	Skills required

	``
	 HADR log write
	 synchronization mode
	     (HADR_SYNCMODE) =
	 NEARSYNC
	 HADR spool log data
	 limit (4KB)
	 (HADR_SPOOL_LIMIT) =
	 AUTOMATIC(52000)
	 HADR log replay delay
	 (seconds) (HADR_REP
	LAY_DELAY) = 0
	 HADR peer window
	 duration (seconds)
	 (HADR_PEER_WINDOW) = 0
	 HADR SSL certifica
	te label
	 (HADR_SSL_LABEL) =
	 HADR SSL Hostname
	 Validation
	 (HADR_SSL_HOST_VAL) =
	 OFF
	``
	
	On `db2-server2` , run the following code:
	
	``
	db2 get db cfg for
	 sample|grep HADR
	 HADR database role
	
	         = STANDBY
	 HADR local host name
	               (HADR_LOC
	AL_HOST) = db2-server2
	 HADR local service
	 name
	 (HADR_LOCAL_SVC) =
	 50011
	 HADR remote host name
	              (HADR_REM
	OTE_HOST) = db2-serve
	r1
	``	

Task	Description	Skills required

	``
	 HADR remote service
	 name
	 (HADR_REMOTE_SVC) =
	 50010
	 HADR instance name
	 of remote server
	 (HADR_REMOTE_INST) =
	 db2inst1
	 HADR timeout value
	
	 (HADR_TIMEOUT) = 120
	 HADR target list
	              (HADR_TAR
	GET_LIST) = db2-serve
	r1:50010|db2-ec2:5
	0012
	 HADR log write
	 synchronization mode
	     (HADR_SYNCMODE) =
	 NEARSYNC
	 HADR spool log data
	 limit (4KB)
	 (HADR_SPOOL_LIMIT) =
	 AUTOMATIC(52000)
	 HADR log replay delay
	 (seconds) (HADR_REP
	LAY_DELAY) = 0
	 HADR peer window
	 duration (seconds)
	 (HADR_PEER_WINDOW) = 0
	 HADR SSL certifica
	te label
	 (HADR_SSL_LABEL) =
	 HADR SSL Hostname
	 Validation
	 (HADR_SSL_HOST_VAL) =
	 OFF
	``
	
	On db2-ec2, run the following code:	

Task	Description	Skills required

	``
	db2 get db cfg for
	 sample|grep HADR
	 HADR database role
	
	         = STANDBY
	 HADR local host name
	               (HADR_LOC
	AL_HOST) = db2-ec2
	 HADR local service
	 name
	 (HADR_LOCAL_SVC) =
	 50012
	 HADR remote host name
	              (HADR_REM
	OTE_HOST) = db2-serve
	r1
	 HADR remote service
	 name
	 (HADR_REMOTE_SVC) =
	 50010
	 HADR instance name
	 of remote server
	 (HADR_REMOTE_INST) =
	 db2inst1
	 HADR timeout value
	
	 (HADR_TIMEOUT) = 120
	 HADR target list
	              (HADR_TAR
	GET_LIST) = db2-serve
	r1:50010|db2-serve
	r2:50011
	 HADR log write
	 synchronization mode
	     (HADR_SYNCMODE) =
	 SUPERASYNC
	 HADR spool log data
	 limit (4KB)
	 (HADR_SPOOL_LIMIT) =
	 AUTOMATIC(52000)
	``	

Task	Description	Skills required

	<pre>HADR log replay delay (seconds) (HADR_REPLAY_DELAY) = 0
	HADR peer window duration (seconds) (HADR_PEER_WINDOW) = 0
	HADR SSL certificate label (HADR_SSL_LABEL) =
	HADR SSL Hostname Validation (HADR_SSL_HOST_VAL) = OFF</pre> The `HADR_LOCAL_HOST` , `HADR_LOCAL_SVC` , `HADR_REMOTE_HOST` , `HADR_REMOTE_SVC` , and `HADR_TARGET_LIST` parameters indicate the one primary and two standby HADR setup.	

Task	Description	Skills required

Stop and start Db2 HADR.	`HADR_TARGET_LIST` is now set up on all three servers. Each Db2 server is aware of the other two. Stop and restart HADR (brief outage) to take advantage of the new configuration.
	
	On `db2-server1`, run the following commands:
	
	`db2 stop hadr on db sample`
	`db2 deactivate db sample`
	`db2 activate db sample`
	
	On `db2-server2`, run the following commands:
	
	`db2 deactivate db sample`
	`db2 start hadr on db sample as standby`
	`SQL1766W  The command completed successfully`
	
	On `db2-ec2`, run the following commands:
	
	`db2 start hadr on db sample as standby`
	`SQL1766W  The command completed successfully`
	
	On `db2-server1`, run the following commands:
		DBA

Task	Description	Skills required

	<pre>db2 start hadr on db
	 sample as primary
	SQL1766W The command
	 completed successfully</pre> The HADR connection between Db2 on premises and on Amazon EC2 is now successfully established. The Db2 primary server `db2-server1` starts streaming transaction log records to both `db2-server2` and `db2-ec2` in real time.	

Make Db2 on Amazon EC2 as primary during the cutover window

Task	Description	Skills required

Make sure that there is no HADR lag on the standby server.	Check HADR status from the primary server `db2-server1`. Don't be alarmed when `HADR_STATE` is in `REMOTE_CATCHUP` status, which is normal when `HADR_SYNCMODE` is set to `SUPERASYNC`. The `PRIMARY_LOG_TIME` and `STANDBY_REPLAY_LOG_TIME` show that they are in sync: <pre>db2pd -hadr -db sample
	
	    HADR_ROLE = PRIMARY</pre>
		DBA

Task	Description	Skills required

	``
	     REPLAY_TYPE =
	 PHYSICAL
	
	   HADR_SYNCMODE =
	 SUPERASYNC
	
	  STANDBY_ID = 2
	
	 LOG_STREAM_ID = 0
	
	      HADR_STATE =
	 REMOTE_CATCHUP
	.....
	
	 PRIMARY_LOG_TIME =
	 10/26/2022 02:11:32.
	000000 (1666750292)
	
	 STANDBY_LOG_TIME =
	 10/26/2022 02:11:32.
	000000 (1666750292)
	              STANDBY_R
	EPLAY_LOG_TIME =
	 10/26/2022 02:11:32.
	000000 (1666750292)
	``	

Task	Description	Skills required

Run HADR takeover.	To complete the migration, make `db2-ec2` the primary database by running the HADR takeover command. Use the command `db2pd` to verify the `HADR_ROLE` value:
	
	`db2 TAKEOVER HADR ON DATABASE sample`
	`DB20000I  The TAKEOVER HADR ON DATABASE command completed successfully.`
	
	`db2pd -hadr -db sample`
	`Database Member 0 -- Database SAMPLE -- Active -- Up 0 days 00:03:25 -- Date 2022-10-26-02.46.45.048988`
	
	`HADR_ROLE = PRIMARY`
	`REPLAY_TYPE = PHYSICAL`
	
	To complete the migration to AWS, point the application connections to Db2 on Amazon EC2.	

Troubleshooting

Issue	Solution
If you use NAT for firewall and security reasons, the host can have two IP addresses (one internal and one external), which can cause an HADR IP address check failure. The `START HADR ON DATABASE` command will return the following message: `HADR_LOCAL_HOST:HADR_LOCAL_SVC (-xx-xx-xx-xx.:50011 (xx.xx.xx .xx:50011)) on remote database is different from HADR_REMOTE_HOST:H ADR_REMOTE_SVC (xx-xx-xx- xx.:50011 (x.x.x.x:50011)) on local database.`	To support HADR in a NAT environment, you can configure the `HADR_LOCAL_HOST` with both the internal and external address. For example, if the Db2 server has the internal name `host1` and the external name `host1E`, `HADR_LOCAL_HOST` can be `HADR_LOCA L_HOST: "host1 | host1E"` .

Related resources

- Db2 backup and restore operations between different operating systems and hardware platforms
- Set up Db2 STORAGE ACCESS ALIAS and DB2REMOTE
- Db2 high availability disaster recovery
- hadr_syncmode - HADR synchronization mode for log writes in peer state configuration parameter

Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using Application Migration Service

Senthil Ramasamy, Amazon Web Services

Summary

This pattern describes the steps for migrating a Microsoft SQL Server database from an on-premises data center to an Amazon Elastic Compute Cloud (Amazon EC2) instance. It uses the AWS Application Migration Service (AWS MGN) to rehost your database using an automated lift-and-shift migration. AWS MGN performs block-level replication of your source database server.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Microsoft SQL Server database in an on-premises data center

Limitations

- Your network bandwidth may be limited between the on-premises data center and AWS.
- AWS MGN is limited to databases that are hosted on standalone servers with dedicated storage. It doesn't support migrating clustered database systems and database systems where the rate of change exceeds a network's throughput.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Product versions

- All versions of Microsoft SQL Server database
- Windows and Linux operating systems that support AWS MGN

Architecture

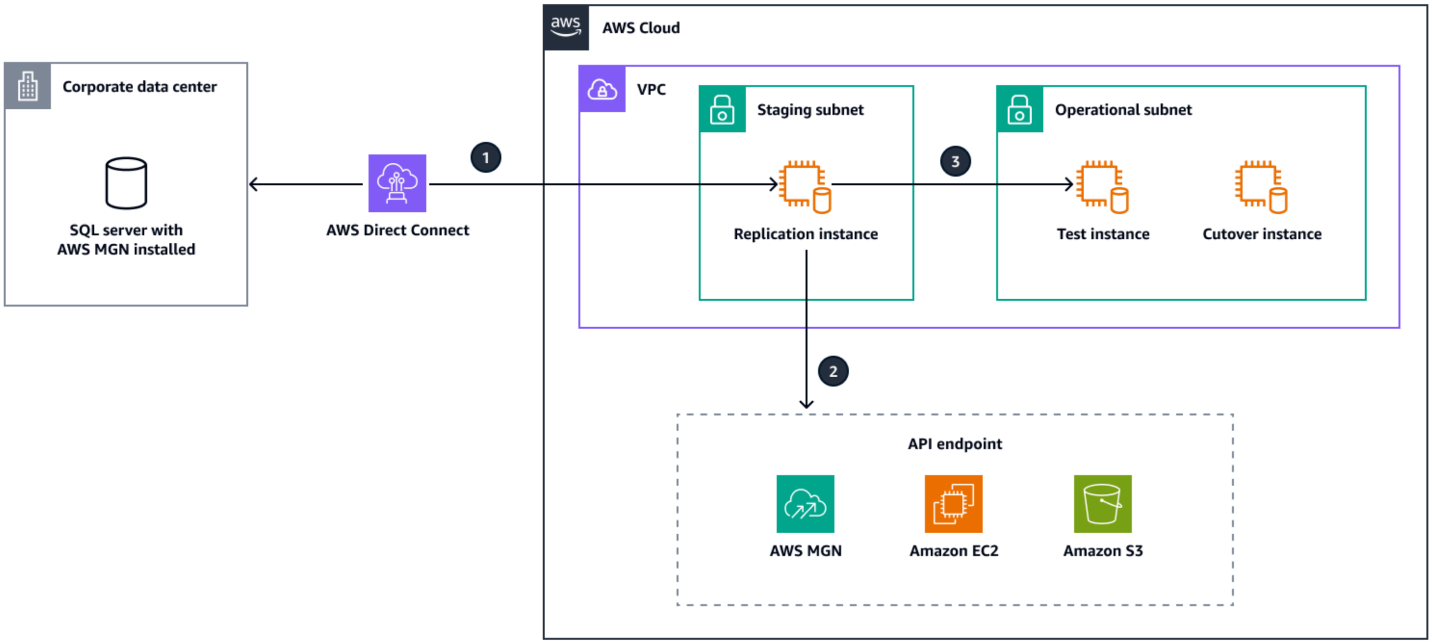
Source technology stack

An on-premises Microsoft SQL Server database

Target technology stack

A Microsoft SQL Server database on an Amazon EC2 instance

Target architecture



This architecture uses AWS MGN to replicate data from an on-premises corporate data center to AWS. The diagram shows the data replication process, API communications, and the test and cutover phases.

1. Data replication:
   - AWS MGN replicates data from the on-premises corporate data center to AWS and initiates ongoing replication of changes.
   - Replication servers in the staging subnet receive and process the data.
2. API communication:
   - Replication servers connect to AWS MGN, Amazon EC2, and Amazon Simple Storage Service (Amazon S3) API endpoints through TCP port 443.
   - AWS MGN manages the migration.
   - Amazon EC2 manages instance operations.
3. Test and cutover:
   - Test instances launch in the operational subnet using replicated data.
   - After successful testing, AWS MGN creates cutover instances for the final migration.

Tools

- AWS Application Migration Service (AWS MGN) helps you rehost (*lift and shift*) applications to the AWS Cloud without change and with minimal downtime.
- AWS Direct Connect links your internal network to a Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Best practices

- Set up API regional endpoints for AWS MGN, Amazon EC2, and Amazon S3 in the virtual private cloud (VPC) to prohibit public access from the internet.
- Set up AWS MGN launch settings to launch target database servers in a private subnet.
- Allow only required ports in database security groups.
- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Set up AWS MGN

Task	Description	Skills required
Configure AWS MGN.	Search for the AWS Applicati on Migration Service in the AWS Management Console, and initiate the setup process. This will create a replicati on template and redirect you to the MGN console	DBA, Migration engineer

Task	Description	Skills required
	Source servers page. As you configure the MGN service, choose a service role from the generated list.	
Add source server.	Add details of your on-premises source database server, and then add the server.	DBA, Migration engineer
Install the AWS MGN agent on the source server.	Download the AWS MGN agent installer to your local system, and transfer the installer to your source database server. To validate the installer hash, see Validating the downloaded AWS Replication Agent installer for Windows 2012.	DBA, Migration engineer

Install AWS MGN agent on source machines

Task	Description	Skills required
Generate client IAM credentials.	Before you install the AWS MGN agent, generate AWS credentials by creating a new IAM user with the appropriate permissions. For more information, see AWS managed policies for AWS Application Migration Service and Generating the required AWS credentials.	DBA, Migration engineer

Task	Description	Skills required
Install the agent on the source server.	Install the agent on the source machine that hosts the Microsoft SQL Server database. For more informati on, see Installing the AWS Replication Agent on Windows servers. Provide the following AWS credentials: • AWS Region • AWS access key ID • AWS secret access key Your unique AWS credentials enable the AWS MGN agent to authenticate and perform migration tasks.	App owner, DBA, Migration engineer

Task	Description	Skills required
Choose disks to replicate.	After entering your AWS credentials, the installer verifies that your server meets the minimum requirements for agent installation (for example, whether the server has enough disk space to install the AWS MGN agent). The installer displays the volume labels and storage details. To replicate your database using AWS MGN service, select the applicable disks on your source server. Enter the path of each disk, separated by commas. If you want to replicate all of the disks, leave the path blank. After you confirm the selected disks, the installation proceeds.	DBA, Migration engineer

Task	Description	Skills required
Monitor synchronization progress.	AWS Replication Agent initiates the synchronization process by first taking a snapshot of the selected disks and then replicating the data. You can monitor the synchronization progress from the **Source server** page in the AWS MGN console. For more information, see [Monitor the server in the migration lifecycle]().	DBA, Migration engineer

Replication using AWS MGN

Task	Description	Skills required
Manage replication progress.	After you start the initial synchronization, your source server appears in the AWS MGN console, where you can manage and monitor the migration. The console displays an estimated time for complete replication, which is based on the total size of selected disks and available network bandwidth.	DBA, Migration engineer
Verify the synchronization.	After the disks on the source server are fully synchronized, verify that all selected disks are listed as fully synced and	App owner, DBA, Migration engineer

Task	Description	Skills required
	no errors are reported in the console. The AWS MGN console will then automatically transitio n the migration lifecycle status to **Ready for testing**, indicating that the replicate d environment in AWS is prepared for performance and functionality testing.	

Test and cut over

Task	Description	Skills required
Configure launch settings.	Choose the source server in the AWS MGN console, and update the launch settings for the target test instance. From the source **Server details** page, navigate to the **Launch settings** tab to configure the test instance. Choose a cost-effective instance type and Amazon Elastic Block Store (Amazon EBS) volume type, and then configure the security groups and network requirements. For more information, see [Launch settings]().	DBA, Migration engineer

Task	Description	Skills required
Launch the target test instance.	Navigate to the AWS MGN console of your synchroni zed source machine, and launch a target test instance by choosing **Test and cut over** and then **Launch test instances**. This creates a launch job that deploys the test instance using your configured settings. The instance launches in the AWS Cloud and replicates your source database server's environme nt. Monitor the launch progress from the **Launch history** page, where you can track the instance creation and address any issues.	DBA, Migration engineer

Task	Description	Skills required
Validate the target test instance.	Validate the Amazon EC2 database server: 1. Ensure the AWS MGN console shows that the test instance is running successfully. 2. Use an RDP client to log in. 3. From the **Start** menu, open **SQL Server Configuration Manager**. 4. Verify the status of SQL Server services, including SQL Server (`MSSQLSERVER`) and SQL Server Agent, ensuring they are intact and set to the correct startup type. 5. Compare disk settings and configurations between the test instance and the source database server, confirming that drive letters, volumes, disk allocations, and necessary directories are mapped correctly. 6. Connect to the SQL Server on the test Amazon EC2 instance and verify that all source databases have migrated and appear in the database list.	DBA, Migration engineer

Task	Description	Skills required
	Conduct validation tests to ensure the database functions as expected.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Rename the server.</td><td>AWS MGN migration involves a storage-level copy of your on-premises source server. Your SQL Server EC2 instance contains only the original source server's details in its binaries, so update the binary information to reflect the new server's name.

Use SQL Server Management Studio (SSMS) to connect to your SQL Server EC2 instance.
Check the server’s name:
<pre><code>SELECT @@SERVERNAME
 AS 'Current Server
 Name';</code></pre>
Rename the SQL Server instance by replacing <code>NEW_SERVER_NAME</code> with the name of your server:
<pre><code>EXEC sp_dropserver
 'OLD_SERVER_NAME'
EXEC sp_addserver
 'NEW_SERVER_NAME',
 'local';</code></pre>
Verify that the server name is correct:
<pre><code>SELECT @@SERVERNAME
 AS 'Updated Server
 Name';</code></pre>
</td><td>DBA, Migration engineer</td></tr>
</table>

Task	Description	Skills required
	5. Restart the SQL Server instance.	

Task	Description	Skills required
Launch the cutover instance.	In the AWS MGN console, on the **Source servers** page, confirm that the migration lifecycle status of the server is **Ready for cutover**. Configure the launch settings for the cutover instance, ensuring that the settings mirror your on-premises environment. Before initiating the cutover, shut down your on-premises database, which ensures the following: • All ongoing transactions are complete. • No new transactions will occur during the cutover process. • Data synchronization between the source and target disks is complete. Initiate the cutover instance in the AWS MGN console. When the cutover instance is operational, log in to the instance and perform the following tests: 1. Ensure that SQL Server starts correctly and that the databases are accessibl e.	App owner, DBA, Migration engineer, Migration lead

Task	Description	Skills required
	2. Validate that your data is intact and consistent with the source server. 3. Run any application tests to confirm that they perform as expected. 4. In the AWS MGN console, set the migration state to **Cutover complete**. 5. Begin routing traffic to the EC2 instance.	

Troubleshooting

Issue	Solution
The initial synchronization fails at the authentication step.	This is a network connectivity issue. The replication server can't connect to AWS MGN.

Related resources

AWS documentation

- Getting started with AWS Application Migration Service
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2
- What is Microsoft SQL Server on Amazon EC2?

Videos

- Performing a Lift and Shift Migration with AWS Application Migration Service (video)

Migrate VMware VMs with HCX Automation by using PowerCLI

Giri Nadiminty and Hassan Adekoya, Amazon Web Services

Naveen Deshwal, None

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

This pattern describes how to migrate VMware on-premises virtual machines (VMs) to VMware Cloud on AWS by using VMware Hybrid Cloud Extension (HCX) Automation powered by VMware PowerCLI scripts. PowerCLI is a command-line tool that's built on Windows PowerShell. It helps you manage VMware software, and automates infrastructure and migration tasks.

You can adapt this pattern for migration between any combination of vCenters, software-defined data centers (SDDCs), and cloud environments. The PowerCLI scripts included with this pattern use automation instead of mouse clicks for all VM configuration and scheduling tasks, so they provide time savings in migration activities and help reduce the risk of human error.

Prerequisites and limitations

Prerequisites

- A VMware Cloud on AWS account with SDDC
- An existing on-premises or cloud-based vCenter or SDDC
- A user account with the necessary permissions for source and destination vCenters or SDDCs
- HCX Site Pairing with HCX Network Extension (HCX-NE) configured between source and destination vCenters or SDDCs
- VMware PowerCLI installed on your server of choice

Limitations

- If the source vCenter uses cross-vCenter NSX, the PowerCLI module will not work. Use a scripting method (such as Python) with the HCX API instead of PowerCLI.

- If the migrated VMs need new names or IP addresses, use a scripting method (such as Python) with the HCX API.
- This pattern doesn't populate the .csv file, which is required. You can populate the file by using VMware vRealize Network Insight (vRNI) or some other method.

Product versions

- VMware vSphere version 5 or later
- VMware HCX version 4.4 or later
- VMware PowerCLI version 12.7 or later

Architecture

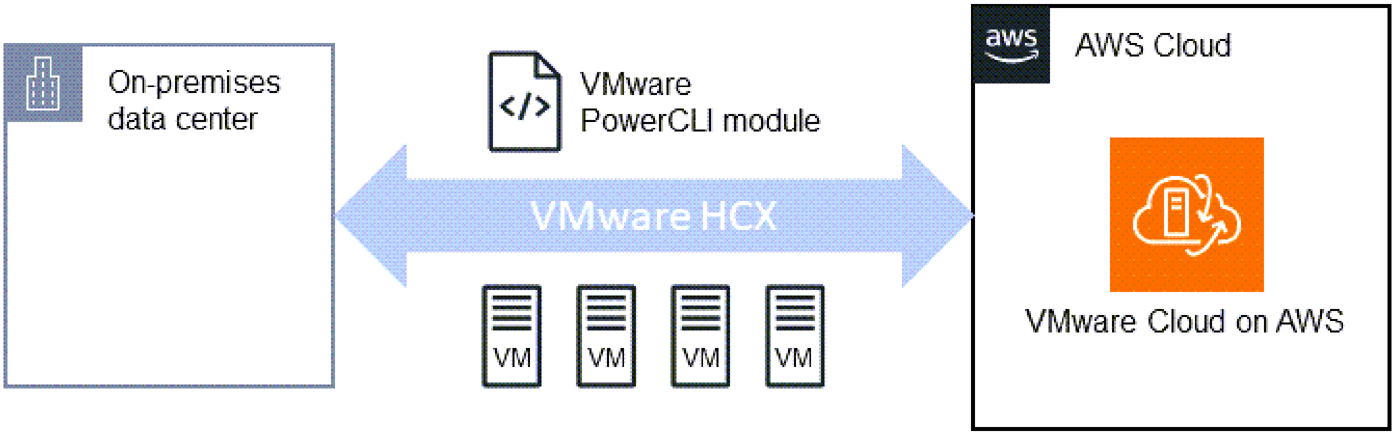
Source technology stack

- On-premises or cloud-based VMware

Target technology stack

- VMware Cloud on AWS

Target architecture



Tools

AWS services

- VMware Cloud on AWS is a service jointly designed by AWS and VMware to help you migrate and extend your on-premises VMware vSphere-based environments to the AWS Cloud.

Other tools

- VMware Hybrid Cloud Extension (HCX) is a utility for migrating workloads from your on-premises VMware environment to VMware Cloud on AWS without changing the underlying platform. Note: This product was formerly known as Hybrid Cloud Extension and NSX Hybrid Connect. This pattern uses HCX for VM migration.
- VMware PowerCLI is a command-line tool for automating VMware vSphere and vCloud management. You run PowerCLI commands in Windows PowerShell by using PowerShell cmdlets. This pattern uses PowerCLI to run migration commands.

Code

Simple, self-contained script

We recommend that you use this single-machine script for initial testing, to verify that configuration options are accepted and behave as expected. For instructions, see the Epics section.

```
<# Manual Variables #>
$HcxServer = "[enterValue]"
$SrcNetworkName = "[enterValue]"
$DstNetworkName = "[enterValue]"
$DstComputeName = "[enterValue]"
$DstDSName = "[enterValue]"
$DstFolderName = "[enterValue]"
$vmName = "[enterValue]"

<# Environment Setup #>
Connect-HCXServer -Server $HcxServer
$HcxDstSite = Get-HCXSite -Destination
$HcxSrcSite = Get-HCXSite -Source
$SrcNetwork = Get-HCXNetwork -Name $SrcNetworkName -Type VirtualWire -Site $HcxSrcSite
$DstNetwork = Get-HCXNetwork -Name $DstNetworkName -Type NsxtSegment -Site $HcxDstSite
$DstCompute = Get-HCXContainer -Name $DstComputeName -Site $HcxDstSite
$DstDS = Get-HCXDatastore -Name $DstDSName -Site $HcxDstSite
$DstFolder = Get-HCXContainer -name $DstFolderName -Site $HcxDstSite
$vm = Get-HCXVM -Name $vmName
```

```
<# Migration #>
$NetworkMapping = New-HCXNetworkMapping -SourceNetwork $SrcNetwork -DestinationNetwork
 $DstNetwork
$NewMigration = New-HCXMigration -VM $vm -MigrationType vMotion -SourceSite $HcxSrcSite
 -DestinationSite $HcxDstSite -Folder $DstFolder -TargetComputeContainer $DstCompute
 -TargetDatastore $DstDS -NetworkMapping $NetworkMapping -DiskProvisionType Thin
 -UpgradeVMTools $True -RemoveISOs $True -ForcePowerOffVm $True -RetainMac $True -
UpgradeHardware $True -RemoveSnapshots $True
```

Full-featured, .csv-based script

After testing is complete, you can use the following script in your production environments. For instructions, see the Epics section.

```
<# Schedule #>
write-host("Getting Time for Scheduling")
$startTime = [DateTime]::Now.AddDays(12)
$endTime = [DateTime]::Now.AddDays(15)

<# Migration #>
Connect-HCXServer -Server [enterValue]
write-host("Getting Source Site")
$HcxSrcSite = Get-HCXSite
write-host("Getting Target Site")
$HcxDstSite = Get-HCXSite -Destination
$HCXVMS = Import-CSV .\Import_VM_list.csv
ForEach ($HCXVM in $HCXVMS) {
    $DstFolder = Get-HCXContainer $HCXVM.DESTINATION_VM_FOLDER -Site $HcxDstSite
    $DstCompute = Get-HCXContainer $HCXVM.DESTINATION_COMPUTE  -Site $HcxDstSite
    $DstDatastore = Get-HCXDatastore $HCXVM.DESTINATION_DATASTORE -Site $HcxDstSite
    $SrcNetwork = Get-HCXNetwork $HCXVM.SOURCE_NETWORK -Type VirtualWire -Site
 $HcxSrcSite
    $DstNetwork = Get-HCXNetwork $HCXVM.DESTINATION_NETWORK -Type NsxtSegment -Site
 $HcxDstSite
    $NetworkMapping = New-HCXNetworkMapping -SourceNetwork $SrcNetwork -
DestinationNetwork $DstNetwork
    $NewMigration = New-HCXMigration -VM (Get-HCXVM $HCXVM.VM_NAME) -MigrationType
 Bulk -SourceSite $HcxSrcSite -DestinationSite $HcxDstSite -Folder $DstFolder -
TargetComputeContainer $DstCompute -TargetDatastore $DstDatastore -NetworkMapping
 $NetworkMapping -DiskProvisionType Thin -UpgradeVMTools $True -RemoveISOs $True -
ForcePowerOffVm $True -RetainMac $True -UpgradeHardware $True -RemoveSnapshots $True -
ScheduleStartTime $startTime -ScheduleEndTime $endTime
    Start-HCXMigration -Migration $NewMigration -Confirm:$false
```

```
}
```

Epics

Collect information for manual variables

Task	Description	Skills required
Find the source and destinati on vCenter and SDDC server names.	PowerCLI scripts require the variables described in this epic. You can gather this information in advance for ease of script use. In the HCX section of the vSphere console, choose **Infrastructure**, **Site Pairing**. Make a note of the source and destination server names that are displayed.	Cloud architect
Find the source and destinati on HCX names.	In the HCX section of the vSphere console, choose **System**, **Administration**. Make a note of the source and destination HCX names that are displayed.	Cloud architect
Find the source and destinati on network names.	In the HCX section of the vSphere console, choose **System**, **Network Extension**. Make a note of the source and destination network names. **Note** As an alternative, you can get the source and destination	Cloud architect

Task	Description	Skills required
	network names by using the PowerCLI **Get-HCXNetwork** and **Get-HCXNetwork-Destination** commands after you connect to the HCX server.	
Gather additional information from the vSphere console.	On the vSphere console, collect the following information: • Names of VMs that you want to migrate • Destination compute environment (cluster/host) • Destination datastore • Destination VM folder name	Cloud architect

Make migration decisions

Task	Description	Skills required
Determine migration options.	Determine the following: • `MigrationType` – The HCX-assisted migration types are vMotion, bulk, cold, and RAV. Your choice depends on your downtime requirements, network bandwidth, migration time frame, and workload	Cloud architect

Task	Description	Skills required
	type. For more information, see the AWS blog post [Migrating Workloads to VMware Cloud on AWS with Hybrid Cloud Extension (HCX)](). • `DiskProvisionType (Thin, Thick)` • `UpgradeVMTools ($True, $False)` • `RemoveISOs ($True, $False)` • `ForcePowerOffVm ($True, $False)` • `RetainMac ($True, $False)` • `UpgradeHardware ($True, $False)` • `RemoveSnapshots ($True, $False)` For more Information about each option, see the [VMware developer documentation]().	

Run the simple script for initial testing

Task	Description	Skills required
Copy the script.	The simple version of the script is self-contained in a single file. You can use it to	Cloud architect

Task	Description	Skills required
	test the migration of a single machine. Copy the first script from the *Code* section of this pattern and store it on the computer that has the VMware PowerCLI module installed. (To install PowerCLI, follow the instructions in the VMware documentation.)	
Set script variables.	Set all the variables in the `Manual Variables` section of the script.	Cloud architect
Set migration variables.	Set all the `New-HCXMigration` settings in the `Migration` section of the script.	Cloud architect
Specify sites.	(Optional) If the source or destination has multiple sites, specify the sites manually in the `Environment Setup` section of the script. If the source and destination have single sites, the script will automatically look the information up.	Cloud architect

Task	Description	Skills required
Run the script.	On the server where PowerCLI is installed, from an elevated PowerShell window, run the script, and enter your credentials when prompted.	Cloud architect
Validate the script.	Confirm that VM migration has been initiated.	Cloud architect

Run the full-featured script to migrate multiple VMs

Task	Description	Skills required
Create and populate the .csv file.	Create a .csv file called `Import_VM_list.csv` on your computer and populate it with the following sample content: `VM_NAME,DESTINATION_VM_FOLDER,DESTINATION_COMPUTE,DESTINATION_DATASTORE,SOURCE_NETWORK,DESTINATION_NETWORK [enterValue],[enterValue],[enterValue],[enterValue],[enterValue],[enterValue]` Replace each `[enterValue]` in the .csv file with the information you collected earlier.	Cloud architect

Task	Description	Skills required
	Note You can populate the .csv file by using VMware vRealize Network Insight (vRNI) or some other method.	
Copy the script.	The full-featured version of the script uses informati on from an external .csv file to automatically migrate multiple VMs. Copy the second script from the *Code* section of this pattern and store it on the computer that has the VMware PowerCLI module installed, in the same folder as the .csv file.	Cloud architect

Task	Description	Skills required
Modify the script.	Edit the script to make the following changes: • Line 7: Set the HCX server variable (`Connect-HCXServer`). • Line 12: (Optional) If you set the .csv filename differently, update it. • Lines 3-4: (Optional) Set the schedule. • Line 20: (Optional) Specify the `New-HCXMigration` settings in the `Migration` section. • Lines 9 and 11: (Optional) If the source or destination includes multiple sites, specify the desired sites manually.	Cloud architect
Run the script.	On the server where PowerCLI is installed, from an elevated PowerShell window, run the script, and enter your credentials when prompted.	Cloud architect
Validate the script.	Confirm that VM migration has been initiated.	Cloud architect

Troubleshooting

Issue	Solution
The script fails with the error message: "All source networks are not mapped to target!"	If the source vCenter uses cross-vCenter NSX, the PowerCLI module will not work. Use a scripting method (such as Python) with the HCX API instead of PowerCLI. This is a known limitation of the PowerCLI script.
The script fails with the error message: "Connect-HCXServer Error: Unauthorized"	The credentials you entered don't provide the necessary permissions.

Related resources

- Migrating Workloads to VMware Cloud on AWS with Hybrid Cloud Extension (HCX) (AWS blog post)
- Choosing a migration approach for relocating your VMware applications and workloads to the AWS Cloud (AWS Prescriptive Guidance)
- Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX (AWS Prescriptive Guidance)
- Getting Started with the HCX Module (VMware blog post)

Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud

Suresh Veeragoni and Deepak Kumar, Amazon Web Services

Summary

Organizations are looking to migrate to the AWS Cloud to increase their agility and resilience. After you migrate your F5 BIG-IP security and traffic management solutions to the AWS Cloud, you can focus on agility and adoption of high-value operational models across your enterprise architecture.

This pattern describes how to migrate an F5 BIG-IP workload to an F5 BIG-IP Virtual Edition (VE) workload on the AWS Cloud. The workload will be migrated by rehosting the existing environment and deploying aspects of replatforming, such as service discovery and API integrations. AWS CloudFormation templates accelerate your workload's migration to the AWS Cloud.

This pattern is intended for technical engineering and architectural teams that are migrating F5 security and traffic management solutions, and accompanies the guide Migrating from F5 BIG-IP to F5 BIG-IP VE on the AWS Cloud on the AWS Prescriptive Guidance website.

Prerequisites and limitations

Prerequisites

- An existing on-premises F5 BIG-IP workload.
- Existing F5 licenses for BIG-IP VE versions.
- An active AWS account.
- An existing virtual private cloud (VPC) configured with an egress through a NAT gateway or Elastic IP address, and configured with access to the following endpoints: Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), AWS Security Token Service (AWS STS), and Amazon CloudWatch. You can also modify the Modular and scalable VPC architecture Quick Start as a building block for your deployments.
- One or two existing Availability Zones, depending on your requirements.
- Three existing private subnets in each Availability Zone.
- AWS CloudFormation templates, available in the F5 GitHub repository.

During the migration, you might also use the following, depending on your requirements:

- An F5 Cloud Failover Extension to manage Elastic IP address mapping, secondary IP mapping, and route table changes.

- If you use multiple Availability Zones, you will need to use the F5 Cloud Failover Extensions to handle the Elastic IP mapping to virtual servers.
- You should consider using F5 Application Services 3 (AS3), F5 Application Services Templates (FAST), or another infrastructure as code (IaC) model to manage the configurations. Preparing the configurations in an IaC model and using code repositories will help with the migration and your ongoing management efforts.

Expertise

- This pattern requires familiarity with how one or more VPCs can be connected to existing data centers. For more information about this, see Network-to-Amazon VPC connectivity options in the Amazon VPC documentation.
- Familiarity is also required with F5 products and modules, including Traffic Management Operating System (TMOS), Local Traffic Manager (LTM), Global Traffic Manager (GTM), Access Policy Manager (APM), Application Security Manager (ASM), Advanced Firewall Manager (AFM), and BIG-IQ.

Product versions

- We recommend that you use F5 BIG-IP version 13.1 or later, although the pattern supports F5 BIG-IP version 12.1 or later.

Architecture

Source technology stack

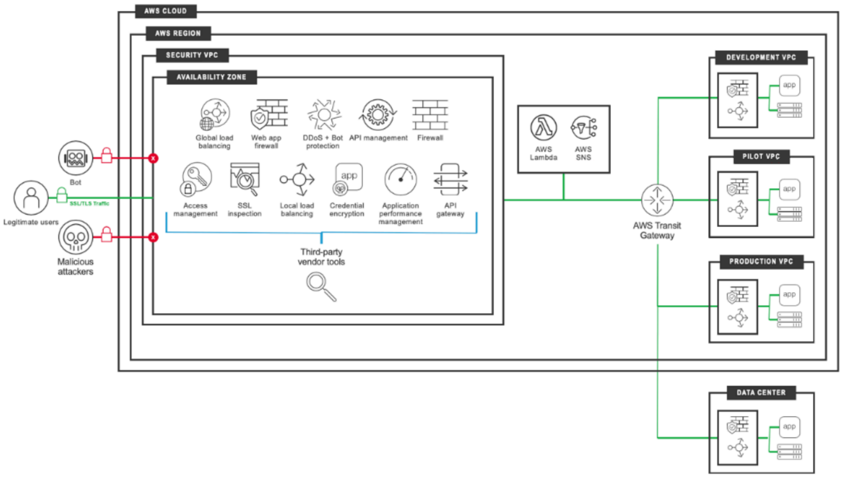
- F5 BIG-IP workload

Target technology stack

- Amazon CloudFront
- CloudWatch
- Amazon EC2
- Amazon S3
- Amazon VPC
- AWS Global Accelerator

- AWS STS
- AWS Transit Gateway
- F5 BIG-IP VE

Target architecture



Tools

- [AWS CloudFormation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html) helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- [Amazon CloudFront](https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html) speeds up distribution of your web content by delivering it through a worldwide network of data centers, which lowers latency and improves performance.
- [Amazon CloudWatch](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html) helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- [Amazon Elastic Compute Cloud (Amazon EC2)](https://docs.aws.amazon.com/ec2/) provides scalable computing capacity in the AWS Cloud You can launch as many virtual servers as you need and quickly scale them up or down.
- [AWS Identity and Access Management (IAM)](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [Amazon Simple Storage Service (Amazon S3)](https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- [AWS Security Token Service (AWS STS)](https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html) helps you request temporary, limited-privilege credentials for users.
- [AWS Transit Gateway](https://docs.aws.amazon.com/vpc/latest/tgw/what-is-transit-gateway.html) is a central hub that connects virtual private clouds (VPCs) and on-premises networks.

- [Amazon Virtual Private Cloud (Amazon VPC)](#) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Discovery and assessment

Task	Description	Skills required
Assess the performance of F5 BIG-IP.	Collect and record the performance metrics of the applications on the virtual server, and metrics of systems that will be migrated. This will help to correctly size the target AWS infrastructure for better cost optimization.	F5 Architect, Engineer and Network Architect, Engineer
Evaluate the F5 BIG-IP operating system and configuration.	Evaluate which objects will be migrated and if a network structure needs to be maintained, such as VLANs.	F5 Architect, Engineer
Evaluate F5 license options.	Evaluate which license and consumption model you will require. This assessmen t should be based on your evaluation of the F5 BIG-IP operating system and configuration.	F5 Architect, Engineer
Evaluate the public applicati ons.	Determine which applicati ons will require public IP addresses. Align those applications to the required instances and clusters to meet performance and service-level	F5 Architect, Cloud Architect , Network Architect, Engineer, App Teams

Task	Description	Skills required
	agreement (SLA) requireme nts.	
Evaluate internal applications.	Evaluate which applications will be used by internal users. Make sure you know where those internal users sit in the organization and how those environments connect to the AWS Cloud. You should also make sure those applications can use domain name system (DNS) as part of the default domain.	F5 Architect, Cloud Architect , Network Architect, Engineer, App Teams
Finalize the AMI.	Not all F5 BIG-IP versions are created as Amazon Machine Images (AMIs). You can use the F5 BIG-IP Image Generator Tool if you have specific required quick-fix engineering (QFE) versions. For more information about this tool, see the "Related resources" section.	F5 Architect, Cloud Architect, Engineer
Finalize the instance types and architecture.	Decide on the instance types, VPC architecture, and interconnected architecture.	F5 Architect, Cloud Architect, Network Architect, Engineer

Complete security and compliance-related activities

Task	Description	Skills required
Document the existing F5 security policies.	Collect and document existing F5 security policies. Make sure you create a copy of them in a secure code repository.	F5 Architect, Engineer
Encrypt the AMI.	(Optional) Your organizat ion might require encryptio n of data at rest. For more information about creating a custom Bring Your Own License (BYOL) image, see the "Related resources" section.	F5 Architect, Engineer Cloud Architect, Engineer
Harden the devices.	This will help protect against potential vulnerabilities.	F5 Architect, Engineer

Configure your new AWS environment

Task	Description	Skills required
Create edge and security accounts.	Sign in to the AWS Management Console and create the AWS accounts that will provide and operate the edge and security services. These accounts might be different from the accounts that operate VPCs for shared services and applications. This step can be completed as part of a landing zone.	Cloud Architect, Engineer

Task	Description	Skills required
Deploy edge and security VPCs.	Set up and configure the VPCs required to deliver edge and security services.	Cloud Architect, Engineer
Connect to the source data center.	Connect to the source data center that hosts your F5 BIG-IP workload.	Cloud Architect, Network Architect, Engineer
Deploy the VPC connections.	Connect the edge and security service VPCs to the application VPCs.	Network Architect, Engineer
Deploy the instances.	Deploy the instances by using the CloudFormation templates from the "Related resources" section.	F5 Architect, Engineer
Test and configure instance failover.	Make sure that the AWS Advanced HA iAPP template or F5 Cloud Failover Extension is configured and operating correctly.	F5 Architect, Engineer

Configure networking

Task	Description	Skills required
Prepare the VPC topology.	Open the Amazon VPC console and make sure that your VPC has all the required subnets and protections for the F5 BIG-IP VE deployment.	Network Architect, F5 Architect, Cloud Architect, Engineer
Prepare your VPC endpoints.	Prepare the VPC endpoints for Amazon EC2, Amazon S3,	Cloud Architect, Engineer

Task	Description	Skills required
	and AWS STS if an F5 BIG-IP workload does not have access to a NAT Gateway or Elastic IP address on a TMM interface.	

Migrate data

Task	Description	Skills required
Migrate the configuration.	Migrate the F5 BIG-IP configuration to F5 BIG-IP VE on the AWS Cloud.	F5 Architect, Engineer
Associate the secondary IPs.	Virtual server IP addresses have a relationship with the secondary IP addresses assigned to the instances. Assign secondary IP addresses and make sure "Allow remap/reassignment" is selected.	F5 Architect, Engineer

Test configurations

Task	Description	Skills required
Validate the virtual server configurations.	Test the virtual servers.	F5 Architect, App Teams

Finalize operations

Task	Description	Skills required
Create the backup strategy.	Systems must be shut down to create a full snapshot. For more information, see "Updating an F5 BIG-IP virtual machine" in the "Related resources" section.	F5 Architect, Cloud Architect, Engineer
Create the cluster failover runbook.	Make sure that the failover runbook process is complete.	F5 Architect, Engineer
Set up and validate logging.	Configure F5 Telemetry Streaming to send logs to the required destinations.	F5 Architect, Engineer

Complete the cutover

Task	Description	Skills required
Cut over to the new deployment.		F5 Architect, Cloud Architect , Network Architect, Engineer, AppTeams

Related resources

Migration guide

- [Migrating from F5 BIG-IP to F5 BIG-IP VE on the AWS Cloud]

F5 resources

- [AWS CloudFormation templates in the F5 GitHub repository]
- [F5 in AWS Marketplace]

- F5 BIG-IP VE overview
- Example Quickstart - BIG-IP Virtual Edition with WAF (LTM + ASM)
- F5 Application services on AWS: an overview (video)
- F5 Application Services 3 Extension User Guide
- F5 cloud documentation
- F5 iControl REST wiki
- F5 Overview of single configuration files (11.x - 15.x)
- F5 whitepapers
- F5 BIG-IP Image Generator Tool
- Updating an F5 BIG-IP VE virtual machine
- Overview of the UCS archive "platform-migrate" option

Migrate an on-premises Go web application to AWS Elastic Beanstalk by using the binary method

Suhas Basavaraj and Shumaz Mukhtar Kazi, Amazon Web Services

Summary

This pattern describes how to migrate an on-premises Go web application to AWS Elastic Beanstalk. After the application is migrated, Elastic Beanstalk builds the binary for the source bundle and deploys it to an Amazon Elastic Compute Cloud (Amazon EC2) instance.

As a rehost migration strategy, this pattern's approach is fast and requires no code changes, which means less testing and migration time.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An on-premises Go web application.
- A GitHub repository that contains your Go application's source code. If you do not use GitHub, there are other ways to [create an application source bundle for Elastic Beanstalk](#).

Product versions

- The most recent Go version supported by Elastic Beanstalk. For more information, see the [Elastic Beanstalk documentation](#).

Architecture

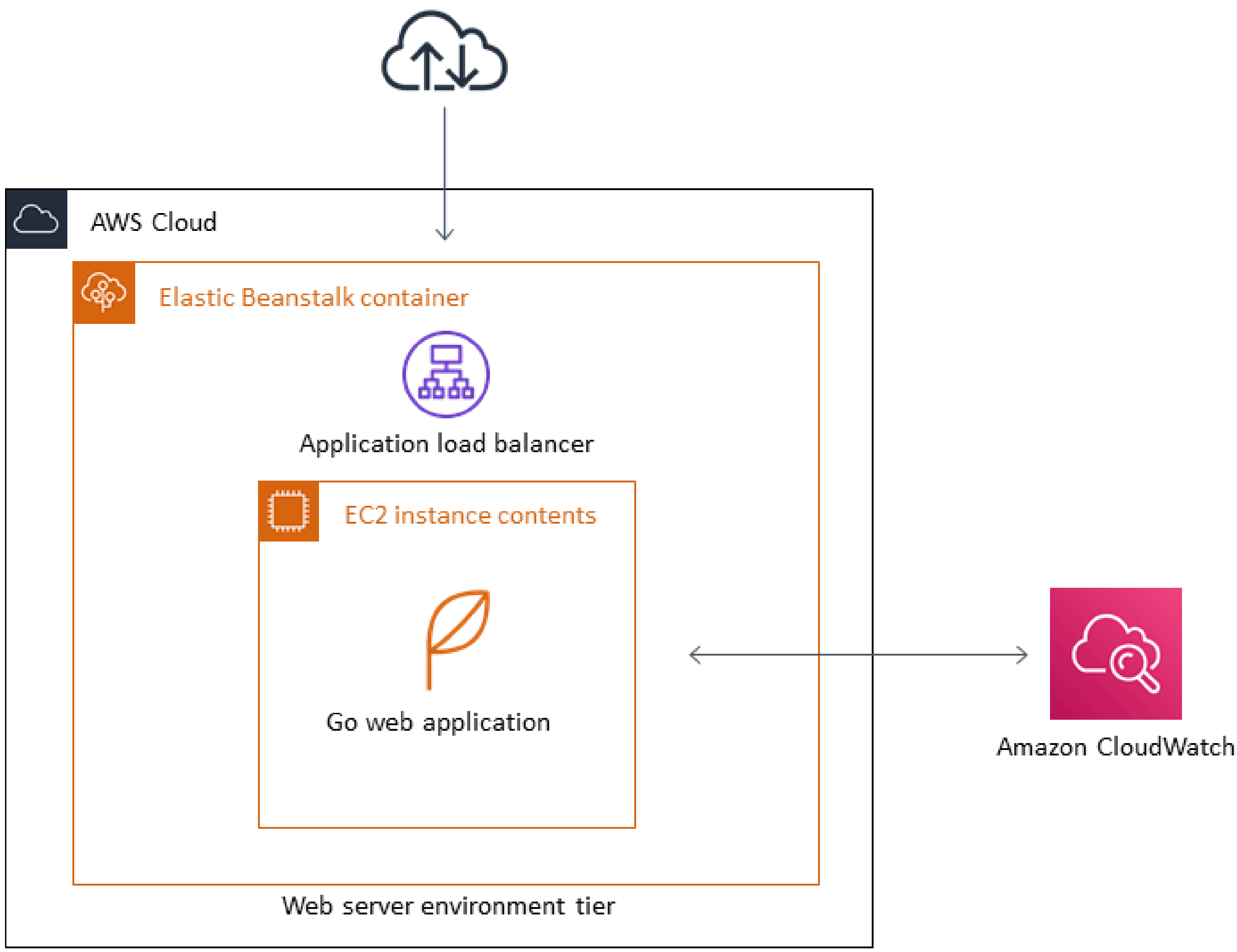
Source technology stack

- An on-premises Go web application

Target technology stack

- AWS Elastic Beanstalk
- Amazon CloudWatch

Target architecture



Tools

- AWS Elastic Beanstalk quickly deploys and manages applications in the AWS Cloud without users having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control.
- GitHub is an open-source distributed version control system.

Epics

Create the Go web application source bundle .zip file

Task	Description	Skills required
Create the source bundle for the Go application.	Open the GitHub repository that contains your Go application's source code and prepare the source bundle. The source bundle contains an `application.go` source file in the root directory, which hosts the main package for your Go application. If you do not use GitHub, see the *Prerequisites* section earlier in this pattern for other ways to create your application source bundle.	System Admin, Application Developer
Create a configuration file.	Create an `.ebextensions` folder in your source bundle, and then create an `options.config` file inside this folder. For more information, see the [Elastic Beanstalk documentation]().	System Admin, Application Developer
Create the source bundle .zip file.	Run the following command. `git archive -o ../godemoapp.zip HEAD` This creates the source bundle .zip file. Download and save the .zip file as a local file.	System Admin, Application Developer

Task	Description	Skills required
	Important The .zip file cannot exceed 512 MB and cannot include a parent folder or top-level directory.	

Migrate the Go web application to Elastic Beanstalk

Task	Description	Skills required
Choose the Elastic Beanstalk application.	1. Sign in to the AWS Management Console and open the [Elastic Beanstalk console](). 2. From the **Regions** list, choose your AWS Region. 3. In the navigation pane, choose **Applications**, and then choose an existing Elastic Beanstalk applicati on or create one. For instructions on how to create an Elastic Beanstalk application, see the [Elastic Beanstalk documentation]().	System Admin, Application Developer
Initiate the Elastic Beanstalk web server environment.	1. On the application overview page, choose **Create a new environme**	System Admin, Application Developer

Task	Description	Skills required
	nt, and then choose **Web server environment**. 2. Complete the **Environment name** and **Domain name** fields. 3. Choose **Platform version**, and select **Go** as your platform.	
Upload the source bundle .zip file to Elastic Beanstalk.	1. In **Application code**, choose **Upload your code**, and then choose **Local file**. 2. Choose the .zip file that contains your source bundle. 3. In **Version label**, give the file a unique name, and then choose **Create environment**.	System Admin, Application Developer
Test the deployed Go web application.	You will be redirected to the Elastic Beanstalk application's overview page. At the top of the overview, next to **Environment ID**, choose the URL that ends in `elasticbeanstalk.com` to navigate to your application. Your application must use this name in its configuration file as an environment variable and display it on the web page.	System Admin, Application Developer

Troubleshooting

Issue	Solution
Unable to access the application through an Application Load Balancer.	Check the target group that contains your Elastic Beanstalk application. If it's unhealthy , log in to your Elastic Beanstalk instance and check the `nginx.conf` file configuration to verify that it routes to the correct health status URL. You might need to change the target group health check URL.

Related resources

- Go platform versions supported by Elastic Beanstalk
- Using configuration files with Elastic Beanstalk
- Creating an example application in Elastic Beanstalk

Migrate an on-premises SFTP server to AWS using AWS Transfer for SFTP

Akash Kumar, Amazon Web Services

Summary

This pattern describes how to migrate an on-premises file transfer solution that uses the Secure Shell (SSH) File Transfer Protocol (SFTP) to the AWS Cloud by using the AWS Transfer for SFTP service. Users generally connect to an SFTP server either through its domain name or by fixed IP. This pattern covers both cases.

AWS Transfer for SFTP is a member of the AWS Transfer Family. It is a secure transfer service that you can use to transfer files into and out of AWS storage services over SFTP. You can use AWS Transfer for SFTP with Amazon Simple Storage Service (Amazon S3) or Amazon Elastic File System (Amazon EFS). This pattern uses Amazon S3 for storage.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing SFTP domain name or fixed SFTP IP.

Limitations

- The largest object that you can transfer in one request is currently 5 GiB. For files that are larger than 100 MiB, consider using Amazon S3 multipart upload.

Architecture

Source technology stack

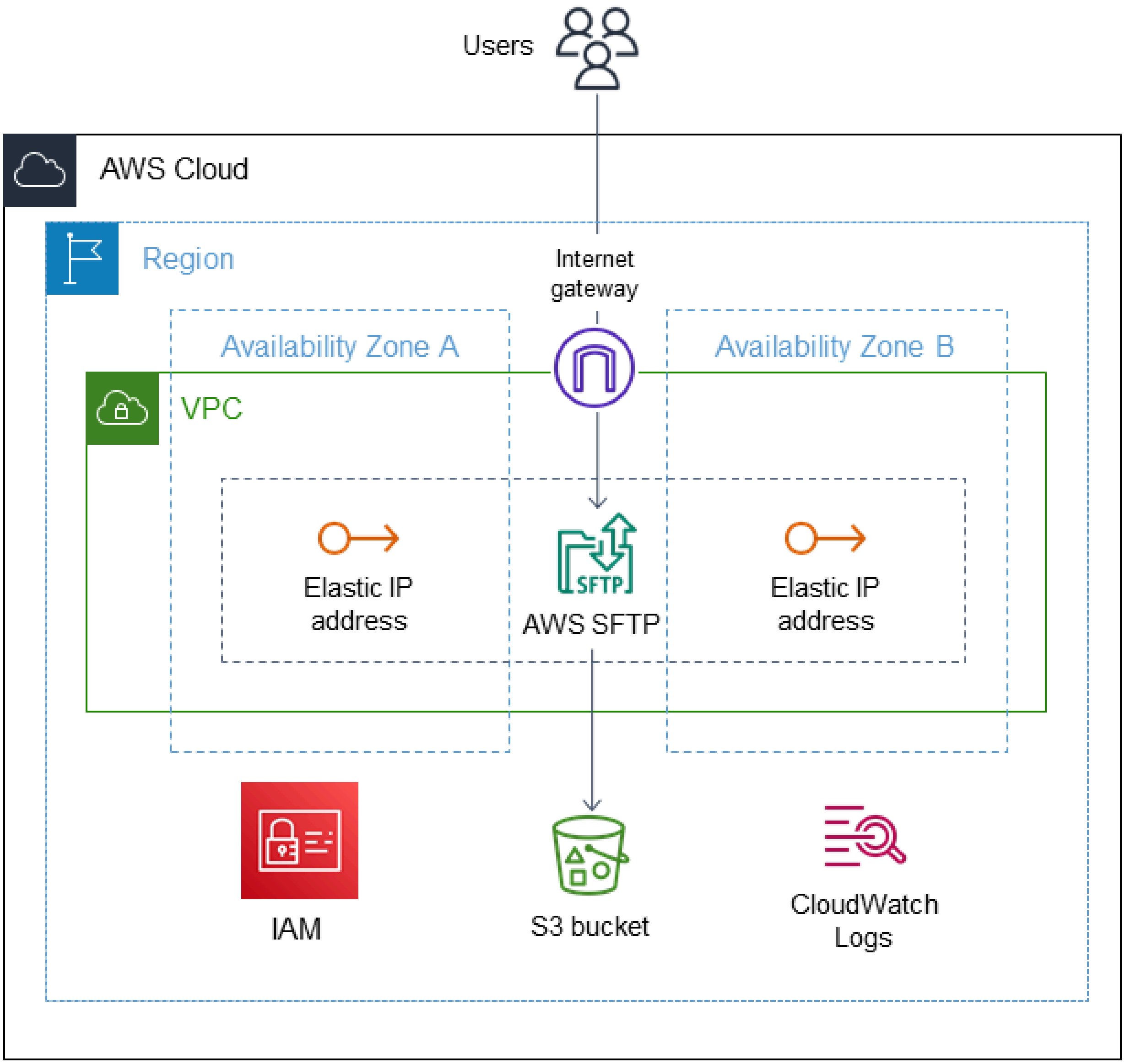
- On-premises flat files or database dump files.

Target technology stack

- AWS Transfer for SFTP
- Amazon S3
- Amazon Virtual Private Cloud (Amazon VPC)

- AWS Identity and Access Management (IAM) roles and policies
- Elastic IP addresses
- Security groups
- Amazon CloudWatch Logs (optional)

Target architecture



Automation and scale

To automate the target architecture for this pattern, use the attached AWS CloudFormation templates:

- `amazon-vpc-subnets.yml` provisions a virtual private cloud (VPC) with two public and two private subnets.
- `amazon-sftp-server.yml` provisions the SFTP server.
- `amazon-sftp-customer.yml` adds users.

Tools

AWS services

- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data. This pattern uses Amazon S3 as the storage system for file transfers.
- AWS Transfer for SFTP helps you transfer files into and out of AWS storage services over the SFTP protocol.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Create a VPC

Task	Description	Skills required
Create a VPC with subnets.	Open the Amazon VPC console. Create a virtual private cloud (VPC) with two public subnets. (The second subnet provides high availabil ity.) —or—	Developer, Systems administr ator

Task	Description	Skills required
	You can deploy the attached CloudFormation template, `amazon-vpc-subnets .yml` , in the [CloudFormation console](#) to automate the tasks in this epic.	
Add an internet gateway.	Provision an internet gateway and attach it to the VPC.	Developer, Systems administr ator
Migrate an existing IP.	Attach an existing IP to the Elastic IP address. You can create an Elastic IP address from your address pool and use it.	Developer, Systems administr ator

Provision an SFTP server

Task	Description	Skills required
Create an SFTP server.	Open the [AWS Transfer Family console](#). Follow the instructions in [Create an internet-facing endpoint for your server](#) in the AWS Transfer Family documenta tion to create an SFTP server with an internet-facing endpoint. For **Endpoint type**, choose **VPC hosted**. For **Access**, choose **Internet Facing**. For **VPC**, choose the VPC you created in the previous epic.	Developer, Systems administr ator

Task	Description	Skills required
	—or— You can deploy the attached CloudFormation template, `amazon-sftp-server .yml` , in the [CloudFormation console](#) to automate the tasks in this epic.	
Migrate the domain name.	Attach the existing domain name to the custom hostname. If you're using a new domain name, use the **Amazon Route 53 DNS** alias. For an existing domain name, choose **Other DNS**. For more information, see [Working with custom hostnames](#) in the AWS Transfer Family documentation.	Developer, Systems administr ator
Add a CloudWatch logging role.	(Optional) if you want to enable CloudWatch logging, create a `Transfer` role with the CloudWatch Logs API operations `logs:CreateLogGrou p` , `logs:CreateLogStre am` , `logs:Desc ribeLogStreams` , and `logs:PutLogEvents` . For more information, see [Log activity with CloudWatch](#) in the AWS Transfer Family documentation.	Developer, system administr ator

Task	Description	Skills required
Save and submit.	Choose **Save**. For **Actions**, choose **Start** and wait for the SFTP server to be created with the status **Online**.	Developer, Systems administr ator

Map Elastic IP addresses to the SFTP server

Task	Description	Skills required
Stop the server so you can modify settings.	On the [AWS Transfer Family console](https://console.aws.amazon.com/transfer/), choose **Servers**, and then select the SFTP server you created. For **Actions**, choose **Stop**. When the server is offline, choose **Edit** to modify its settings.	Developer, system administr ator
Choose Availability Zones and subnets.	In the **Availability Zones** section, choose the Availabil ity Zones and subnets for your VPC.	Developer, Systems administr ator
Add Elastic IP addresses.	For **IPv4 Addresses**, choose an Elastic IP address for each subnet, and then choose **Save**.	Developer, Systems administr ator

Add users

Task	Description	Skills required
Create an IAM role for users to access the S3 bucket.	Create a IAM role for `Transfer` and add `s3:ListBu`	Developer, Systems administr ator

Task	Description	Skills required
	`cket` , `s3:GetBucketLocation` , and `s3:PutObject` with the S3 bucket name as a resource. For more information, see [Create an IAM role and policy](#) in the AWS Transfer Family documentation. —or— You can deploy the attached CloudFormation template, `amazon-sftp-customer.yml` , in the [CloudFormation console](#) to automate the tasks in this epic.	
Create an S3 bucket.	Create a S3 bucket for the application.	Developer, Systems administrator
Create optional folders.	(Optional) If you want to store files for users separately, in specific Amazon S3 folders, add folders as appropriate.	Developer, Systems administrator
Create an SSH public key.	To create an SSH key pair, see [Generate SSH keys](#) in the AWS Transfer Family documentation.	Developer, Systems administrator

Task	Description	Skills required
Add users.	On the [AWS Transfer Family console](), choose **Servers**, select the SFTP server you created, and then choose **Add user**. For **Home directory**, choose the S3 bucket you created. For **SSH public key**, specify the public key portion of the SSH key pair. Add users for the SFTP server, and then choose **Add**.	Developer, Systems administrator

Test the SFTP server

Task	Description	Skills required
Update the security group.	In the **Security Groups** section of your SFTP server, add your test machine's IP to gain SFTP access.	Developer
Use an SFTP client utility to test the server.	Test file transfers by using any SFTP client utility. For a list of clients and instructions, see [Transferring files using a client]() in the AWS Transfer Family documentation.	Developer

Related resources

- [AWS Transfer Family User Guide]()
- [Amazon S3 User Guide]()
- [Elastic IP addresses]() in the Amazon EC2 documentation

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Migrate an on-premises VM to Amazon EC2 by using AWS Application Migration Service

Thanh Nguyen, Amazon Web Services

Summary

When it comes to application migration, organizations can take different approaches to rehost (lift and shift) the application's servers from the on-premises environment to the Amazon Web Services (AWS) Cloud. One way is to provision new Amazon Elastic Compute Cloud (Amazon EC2) instances and then install and configure the application from scratch. Another approach is to use third-party or AWS native migration services to migrate multiple servers at the same time.

This pattern outlines the steps for migrating a supported virtual machine (VM) to an Amazon EC2 instance on the AWS Cloud by using AWS Application Migration Service. You can use the approach in this pattern to migrate one or multiple virtual machines manually, one by one, or automatically by creating appropriate automation scripts based on the outlined steps.

Prerequisites and limitations

Prerequisites

- An active AWS account in one of the AWS Regions that support Application Migration Service
- Network connectivity between the source server and target EC2 server through a private network by using AWS Direct Connect or a virtual private network (VPN), or through the internet

Limitations

- For the latest list of supported Regions, see the [Supported AWS Regions].
- For a list of supported operating systems, see the [Supported operating systems] and the *General* section of [Amazon EC2 FAQs].

Architecture

Source technology stack

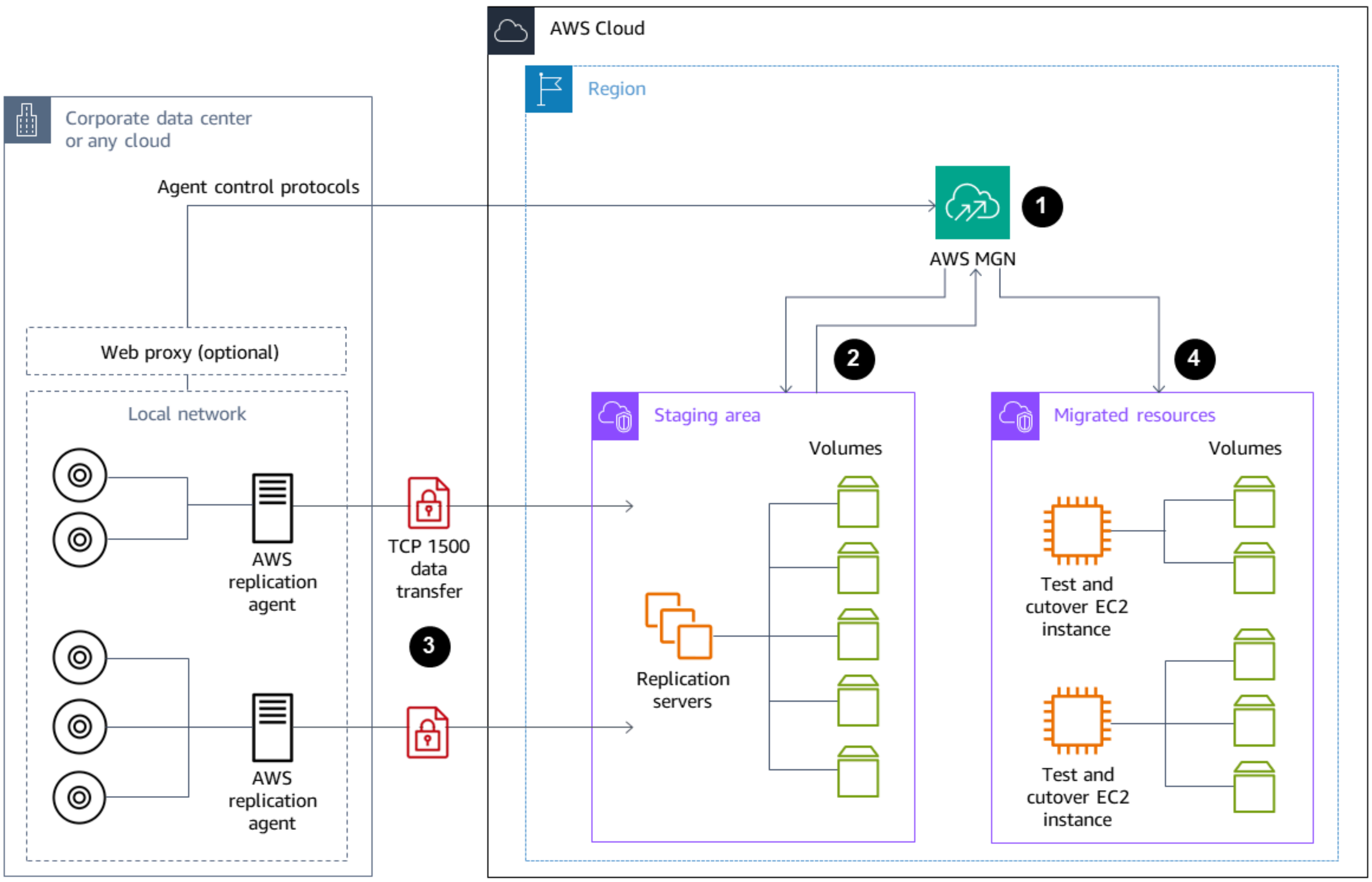
- A physical, virtual, or cloud-hosted server running an operating system supported by Amazon EC2

Target technology stack

- An Amazon EC2 instance running the same operating system as the source VM
- Amazon Elastic Block Store (Amazon EBS)

Source and target architecture

The following diagram shows the high-level architecture and main components of the solution. In the on-premises data center, there are virtual machines with local disks. On AWS, there is a staging area with replication servers and a migrated resources area with EC2 instances for test and cutover. Both subnets contain EBS volumes.

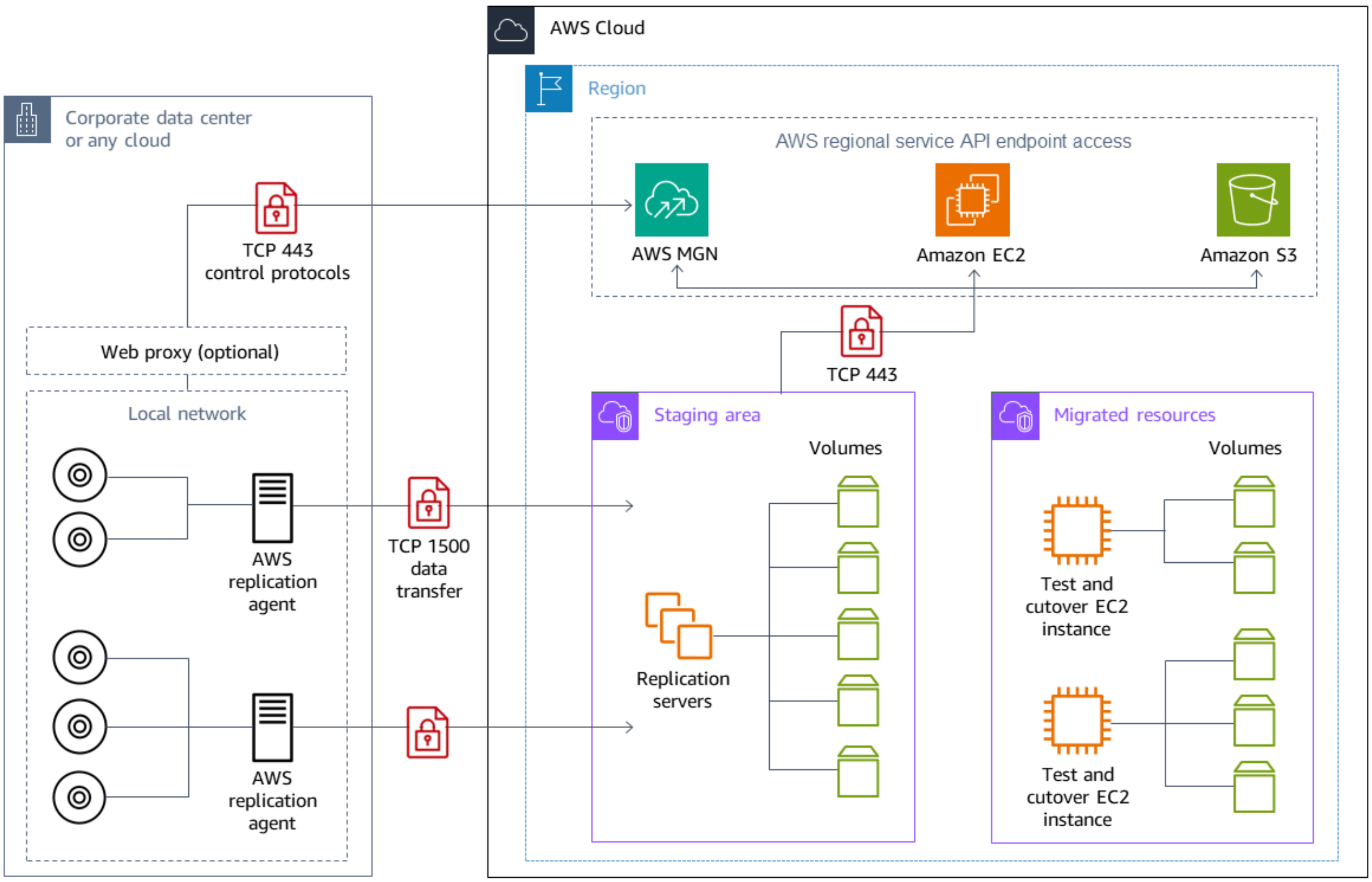


1. Initialize AWS Application Migration Service.
2. Set up the staging area server configuration and reporting, including staging area resources.
3. Install agents on source servers, and use continuous block-level data replication (compressed and encrypted).

4. Automate orchestration and system conversion to shorten the cutover window.

Network architecture

The following diagram shows the high-level architecture and main components of the solution from the networking perspective, including required protocols and ports for communication between primary components in the on-premises data center and on AWS.



Tools

- [AWS Application Migration Service](https://docs.aws.amazon.com/mgn/latest/ug/what-is-application-migration-service.html) helps you rehost (*lift and shift*) applications to the AWS Cloud without change and with minimal downtime.

Best practices

- Do not take the source server offline or perform a reboot until the cutover to the target EC2 instance is complete.

- Provide ample opportunity for the users to perform user acceptance testing (UAT) on the target server to identify and resolve any issues. Ideally, this testing should start at least two weeks before cutover.
- Frequently monitor the server replication status on the Application Migration Service console to identify issues early on.
- Use temporary AWS Identity and Access Management (IAM) credentials for agent installation instead of permanent IAM user credentials.

Epics

Generate AWS credentials

Task	Description	Skills required
Create the AWS Replication Agent IAM role.	Sign in with administrative permissions to the AWS account. On the AWS Identity and Access Management (IAM) [console](https://console.aws.amazon.com/iam/), create an IAM role: 1. On the IAM console, choose **Roles**. 2. Choose **Create role**. 3. On the **Select trusted entity** page, in **Trusted entity type** section, select **AWS Account**. 4. In the **An AWS account** section, select **This account (<*account-id*>)**. 5. Choose **Next**. 6. On the **Add permissions** page, search for the `AWSApplicationMigr`	AWS administrator, Migration engineer

Task	Description	Skills required
	`ationAgentInstallationPolicy` policy, select the check box next to the policy name. 7. Choose **Next**. 8. On the **Role details** page, enter **MGN_Agent_Installation_Role** as the role name. 9. Verify that the fields are correct, and then choose **Create role**.	

Task	Description	Skills required
Generate temporary security credentials.	On a machine with AWS Command Line Interface (AWS CLI) installed, sign in with administrative permissions. Or alternatively (within a supported AWS Region), on the AWS Management Console, sign in with administrative permissions to the AWS account, and open AWS CloudShell. Generate temporary credentials with the following command, replacing `<account-id>` with the AWS account ID. `aws sts assume-role --role-arn arn:aws:iam::<account-id>:role/MGN_Agent_Installation_Role --role-session-name mgn_installation_session_role` From the output of the command, copy the values for `AccessKeyId` , `SecretAccessKey` , and `SessionToken` . Store them in a safe location for later use.	AWS administrator, Migration engineer

Task	Description	Skills required
	Important These temporary credentials will expire after one hour. If you need credentials after one hour, repeat the previous steps.	

Initialize Application Migration Service and create the Replication Settings template

Task	Description	Skills required
Initialize the service.	On the console, sign in with administrative permissions to the AWS account. Choose **Application Migration Service**, and then choose **Get started**.	AWS administrator, Migration engineer
Create and configure the Replication Settings template.	1. Provide the following configuration details: a. Select the staging area subnet. b. Select the replication server instance type (`t3.small` by default). c. Select the EBS volume type (gp3 by default). d. Select the EBS encryptio n option.	AWS administrator, Migration engineer

Task	Description	Skills required
	e. Ensure that the **Always use Application Migration Service security group** check box is selected. f. Select the **Use private IP for data replication (VPN, DirectConnect, VPC peering)** check box if you are using private network connectivity between the on-premises environment and AWS. g. Select the **Throttle network bandwidth (per server - in Mbps)** check box if you want to limit the network bandwidth for Application Migration Service. 2. Choose **Create template**. Application Migration Service will automatically create all the IAM roles required to facilitate data replication and the launching of migrated servers.	

Install AWS Replication Agents on source machines

Task	Description	Skills required
Have the required AWS credentials ready.	When you run the installer file on a source server, you will need to enter the temporary credentials that you generated earlier, including `AccessKeyId` , `SecretAccessKey` , and `SessionToken` .	Migration engineer, AWS administrator
For Linux servers, install the agent.	Copy the installer command, log in to your source servers, and run the installer. For detailed instructions, see the [AWS documentation]().	AWS administrator, Migration engineer
For Windows servers, install the agent.	Download the installer file to each server, and then run the installer command. For detailed instructions, see the [AWS documentation]().	AWS administrator, Migration engineer
Wait for initial data replicati on to be completed.	When the agent has been installed, the source server will appear on the Applicati on Migration Service console, in the **Source servers** section. Wait while the server undergoes initial data replication.	AWS administrator, Migration engineer

Configure launch settings

Task	Description	Skills required
Specify the server details.	On the Application Migration Service console, choose the **Source servers** section, and then choose a server name from the list to access the server details.	AWS administrator, Migration engineer
Configure the launch settings.	Choose the **Launch settings** tab. You can configure a variety of settings, including general launch settings and EC2 launch template settings. For detailed instructions, see the [AWS documentation]().	AWS administrator, Migration engineer

Perform a test

Task	Description	Skills required
Test the source servers.	1. On the Application Migration Service console, in the **Source servers** section, ensure that the source servers' **Migration lifecycle** is **Ready for testing** and that **Data replication status** is **Healthy**. 2. Select the check box to the left of each source server.	AWS administrator, Migration engineer

Task	Description	Skills required
	3. Choose **Test and Cutover**, and then choose **Launch Test Instance**. 4. When prompted, choose **Launch**. The servers will be launched.	
Verify that the test completed successfully.	After the test server is completely launched, the **Alerts** status on the page will show **Launched** for each server.	AWS administrator, Migration engineer
Test the server.	Perform testing against the test server to ensure that it functions as expected.	AWS administrator, Migration engineer

Schedule and perform a cutover

Task	Description	Skills required
Schedule a cutover window.	Schedule an appropriate cutover timeframe with relevant teams.	AWS administrator, Migration engineer
Perform the cutover.	1. On the Application Migration console, on the **Source Servers** page, select the check box to the left of each source server. 2. Choose **Test and Cutover**, and select **Mark as 'Ready for cutover'**.	AWS administrator, Migration engineer

Task	Description	Skills required
	3. Verify that each source server's **Migration lifecycle** is **Ready for cutover**. 4. Choose **Test and Cutover**, and then select **Launch cutover instances**. 5. When prompted, choose **Launch**. The servers will be launched. The source server's **Migration lifecycle** will change to **Cutover in progress**.	
Verify that the cutover completed successfully.	After the cutover servers are completely launched, the **Alerts** status on the **Source Servers** page will show **Launched** for each server.	AWS administrator, Migration engineer
Test the server.	Perform testing against the cutover server to ensure that it functions as expected.	AWS administrator, Migration engineer
Finalize the cutover.	Choose **Test and Cutover**, and then select **Finalize cutover** to finalize the migration process.	AWS administrator, Migration engineer

Related resources

- [AWS Application Migration Service]
- [AWS Application Migration Service User Guide]

Migrate small sets of data from on premises to Amazon S3 using AWS SFTP

Charles Gibson and Sergiy Shevchenko, Amazon Web Services

Summary

This pattern describes how to migrate small sets of data (5 TB or less) from on-premises data centers to Amazon Simple Storage Service (Amazon S3) by using AWS Transfer for SFTP (AWS SFTP). The data can be either database dumps or flat files.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS Direct Connect link established between your data center and AWS

Limitations

- The data files must be less than 5 TB. For files over 5 TB, you can perform a multipart upload to Amazon S3 or choose another data transfer method.

Architecture

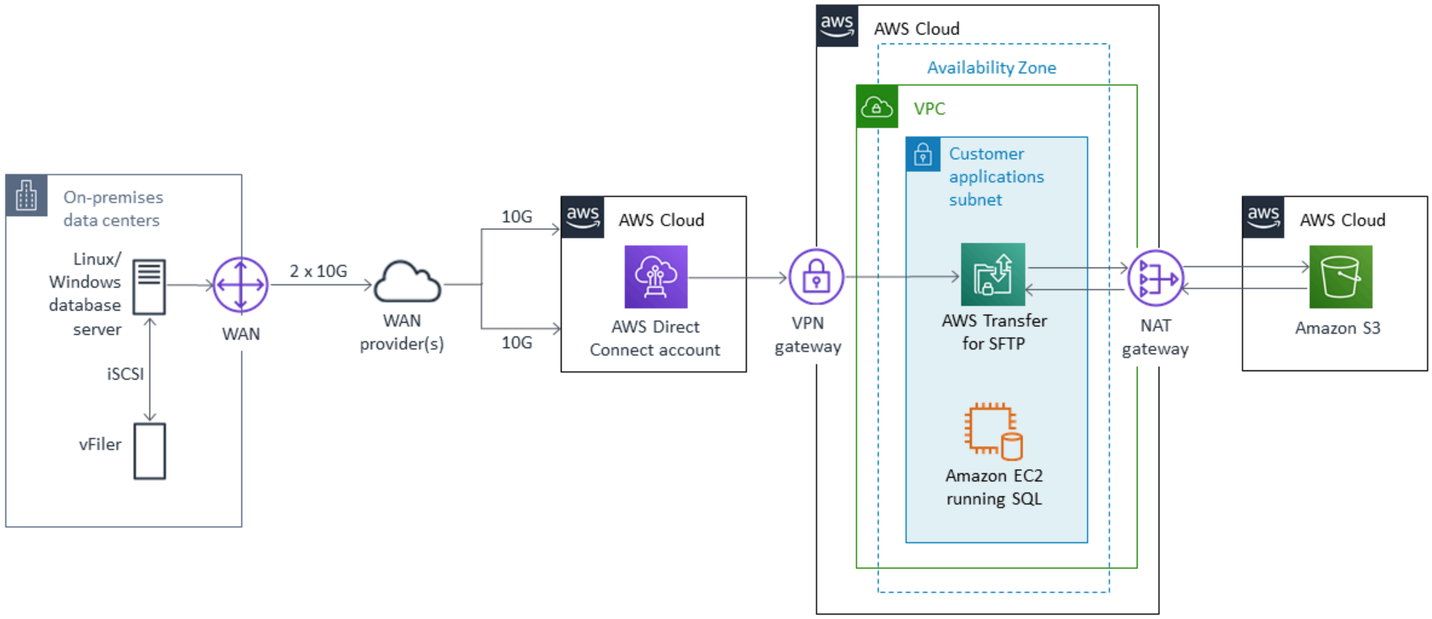
Source technology stack

- On-premises flat files or database dumps

Target technology stack

- Amazon S3

Source and target architecture



Tools

- [AWS SFTP](#) – Enables the transfer of files directly into and out of Amazon S3 using Secure File Transfer Protocol (SFTP).
- [AWS Direct Connect](#) – Establishes a dedicated network connection from your on-premises data centers to AWS.
- [VPC endpoints](#) – Enable you to privately connect a VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without an internet gateway, network address translation (NAT) device, VPN connection, or AWS Direct Connect connection. Instances in a VPC don't require public IP addresses to communicate with resources in the service.

Epics

Prepare for the migration

Task	Description	Skills required
Document the current SFTP requirements.		Application owner, SA
Identify the authentication requirements.	Requirements may include key-based authentication,	Application owner, SA

Task	Description	Skills required
	user name or password, or identity provider (IdP).	
Identify the application integration requirements.		Application owner
Identify the users who require the service.		Application owner
Determine the DNS name for the SFTP server endpoint.		Networking
Determine the backup strategy.		SA, DBA (if data is transferr ed)
Identify the application migration or cutover strategy.		Application owner, SA, DBA

Configure the infrastructure

Task	Description	Skills required
Create one or more virtual private clouds (VPCs) and subnets in your AWS account.		Application owner, AMS
Create the security groups and network access control list (ACL).		Security, Networking, AMS
Create the S3 bucket.		Application owner, AMS
Create the identity and access management (IAM) role.	Create an IAM policy that includes the permissions to enable AWS SFTP to access your S3 bucket. This IAM policy determines what level	Security, AMS

Task	Description	Skills required
	of access you provide SFTP users. Create another IAM policy to establish a trust relationship with AWS SFTP.	
Associate a registered domain (optional).	If you have your own registere d domain, you can associate it with the SFTP server. You can route SFTP traffic to your SFTP server endpoint from a domain or from a subdomain.	Networking, AMS
Create an SFTP server.	Specify the identity provider type used by the service to authenticate your users.	Application owner, AMS
Open an SFTP client.	Open an SFTP client and configure the connection to use the SFTP endpoint host. AWS SFTP supports any standard SFTP client. Commonly used SFTP clients include OpenSSH, WinSCP, Cyberduck, and FileZilla. You can get the SFTP server host name from the AWS SFTP console.	Application owner, AMS

Plan and test

Task	Description	Skills required
Plan the application migration.	Plan for any application configuration changes required, set the migration	Application owner, AMS

Task	Description	Skills required
	date, and determine the test schedule.	
Test the infrastructure.	Test in a non-production environment.	Application owner, AMS

Related resources

References

- AWS Transfer for SFTP User Guide
- AWS Direct Connect resources
- VPC Endpoints

Tutorials and videos

- AWS Transfer for SFTP (video)
- AWS Transfer for SFTP user guide
- AWS SA Whiteboarding - Direct Connect (video)

Migrate from Oracle GlassFish to AWS Elastic Beanstalk

Sandeep Bondugula, Amazon Web Services

Summary

This pattern describes how to migrate a Java application running on an on-premises Oracle GlassFish server to AWS Elastic Beanstalk in the AWS Cloud.

On AWS, the Java application is deployed on a Docker GlassFish server with AWS Elastic Beanstalk, which runs in an Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling group.

Additional features:

- Amazon Elastic Beanstalk acts as a wrapper for several underlying resources. It sets up Elastic Load Balancing (which handles incoming traffic from Amazon Route 53), disperses the traffic to one or more EC2 instances, and also serves as a deployment tool.
- To migrate an on-premises database to Amazon Relational Database Service (Amazon RDS), update the database connection details. In the backend database, you can configure Amazon RDS Multi-AZ deployments and choose the database engine type.
- You can use Multi-AZ deployment for high availability along with the Auto Scaling group and scaling policy to improve resiliency.
- You can set up a scaling policy based on Amazon CloudWatch metrics.
- In AWS Elastic Beanstalk, you can configure the underlying Elastic Load Balancing settings and Amazon EC2 Auto Scaling.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An on-premises Java application running on GlassFish
- A Java Web Application Resource (WAR) file

Product versions

- Oracle Glassfish 4.1.2 and 5.0

- Java 7 GlassFish 4.0
- Java 8 GlassFish 4.1 or later

Architecture

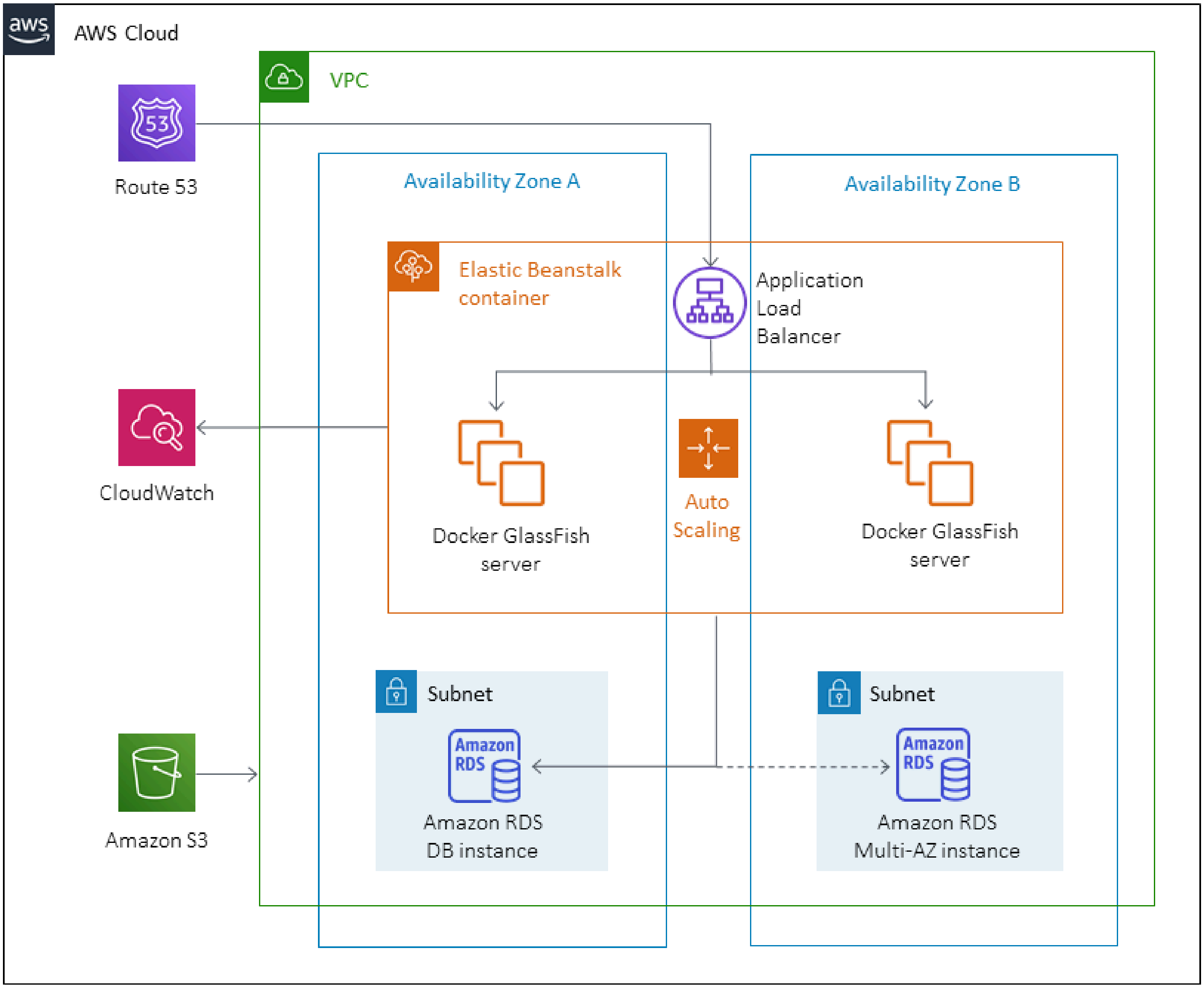
Source technology stack

- Applications developed in GlassFish

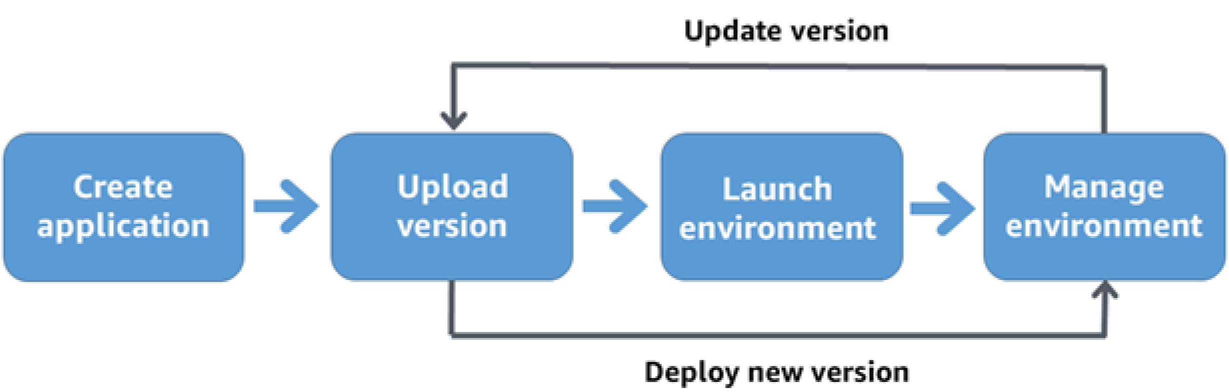
Target technology stack

- Elastic Beanstalk

Target architecture



Deployment workflow



Tools

- Amazon Elastic Beanstalk – A service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on servers including Apache, NGINX, Passenger, and IIS.
- Amazon CloudWatch – Provides data and actionable insights to monitor applications, responds to systemwide performance changes, optimizes resource utilization, and provides a unified view of operational health.
- Docker – A platform that packages software into standardized units to build, test, and deploy applications quickly.
- Java – A general-purpose programming language. Java is class-based, object-oriented, and designed to have fewer implementation dependencies.

Epics

Set up a VPC

Task	Description	Skills required
Create a virtual private cloud (VPC) instance with the required information.		SysAdmin
Create at least two subnets within the VPC.		SysAdmin
Create a route table per requirements.		SysAdmin

Set up Amazon S3

Task	Description	Skills required
Create an Amazon Simple Storage Service (Amazon S3) bucket.		SysAdmin

Task	Description	Skills required
Copy the WAR file to the S3 bucket and upload the application code.		SysAdmin

Create an IAM role

Task	Description	Skills required
Create an AWS Identity and Access Management (IAM) role.	You can use the default "aws-elasticbeanstalk-ec2-role" profile, or let Elastic Beanstalk create it automatic ally.	SysAdmin

Set up Elastic Beanstalk

Task	Description	Skills required
Open the Elastic Beanstalk dashboard.		SysAdmin
Create a new application and choose the web server environment.		SysAdmin
Choose GlassFish Docker as the preconfigured platform.		SysAdmin
Upload the code.	Provide the S3 bucket file URL or ZIP file from local system files.	SysAdmin

Task	Description	Skills required
Choose the environment type.	In Configuration Capacity settings, choose either Single Instance or Load Balancer.	SysAdmin
Configure Load Balancer.	If you chose Load Balancer in the previous step, configure the Multi-AZ deployment.	SysAdmin
In Configuration Security settings, choose the previousl y created IAM role.		SysAdmin
In Configuration Security settings, if you have an existing key pair, use it or create a new Amazon EC2 key pair.		SysAdmin
In Configuration Monitoring settings, configure Amazon CloudWatch.		SysAdmin
In Configuration Security settings, choose the previousl y created VPC.		SysAdmin
Choose Create Environment.		SysAdmin

Test the application

Task	Description	Skills required
Test the application by using the URL provided in the created environment.		

Task	Description	Skills required
Apply the Domain Name Service (DNS) changes in Amazon Route 53.		

Related resources

- Oracle GlassFish documentation
- GlassFish Open Source Java EE Reference Implementation
- AWS Elastic Beanstalk documentation
- Using Elastic Beanstalk with Amazon CloudWatch
- AWS Elastic Beanstalk pricing
- EC2 Auto Scaling Group
- Scaling the Size of Your Auto Scaling Group
- Amazon RDS Multi-AZ Deployments

Migrate an on-premises Oracle database to Oracle on Amazon EC2

Baji Shaik and Pankaj Choudhary, Amazon Web Services

Summary

This pattern walks you through the steps for migrating an on-premises Oracle database to Oracle on an Amazon Elastic Compute Cloud (Amazon EC2) instance. It describes two options for migration: using AWS Data Migration Service (AWS DMS) or using native Oracle tools such as RMAN, Data Pump import/export, transportable tablespaces, and Oracle GoldenGate.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle database in an on-premises data center

Limitations

- The target operating system (OS) must be supported by Amazon EC2. For a complete list of supported systems, see Amazon EC2 FAQs.

Product versions

- Oracle versions 10.2 and later (for versions 10.x), 11g and up to 12.2, and 18c for the Enterprise, Standard, Standard One, and Standard Two editions. For the latest list of versions supported by AWS DMS, see "On-premises and Amazon EC2 instance databases" in Sources for Data Migration in the AWS DMS documentation.

Architecture

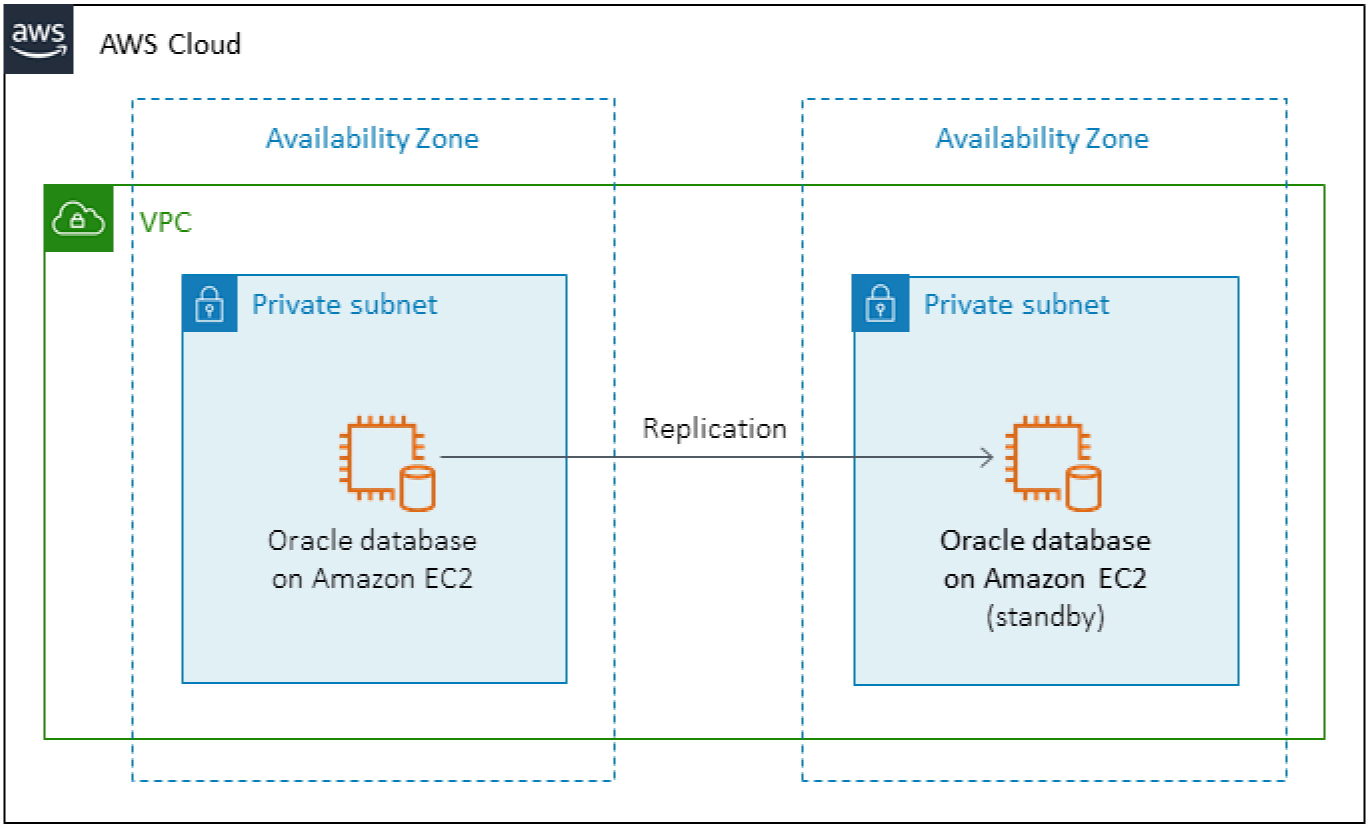
Source technology stack

- An on-premises Oracle database

Target technology stack

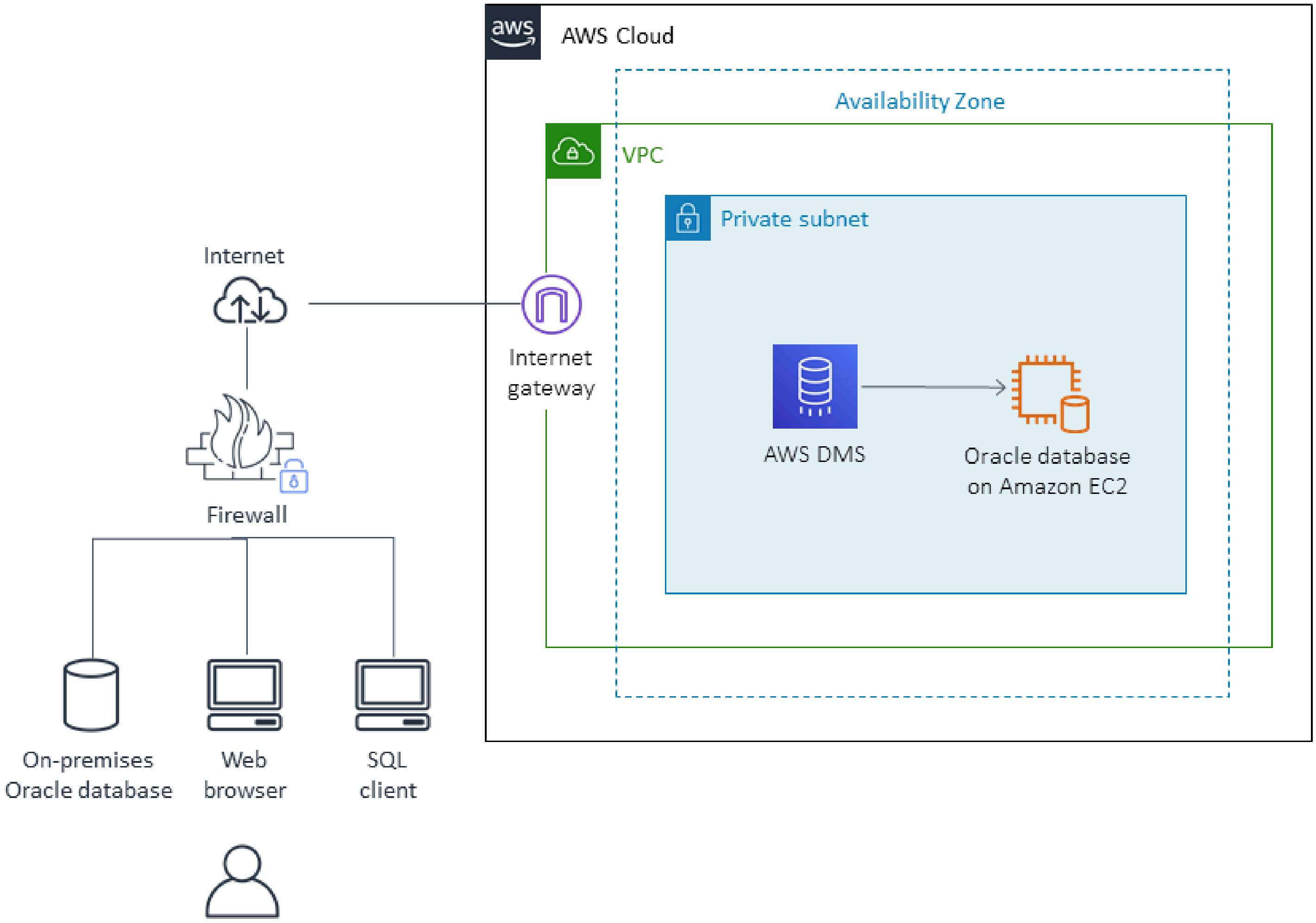
- An Oracle database instance on Amazon EC2

Target architecture

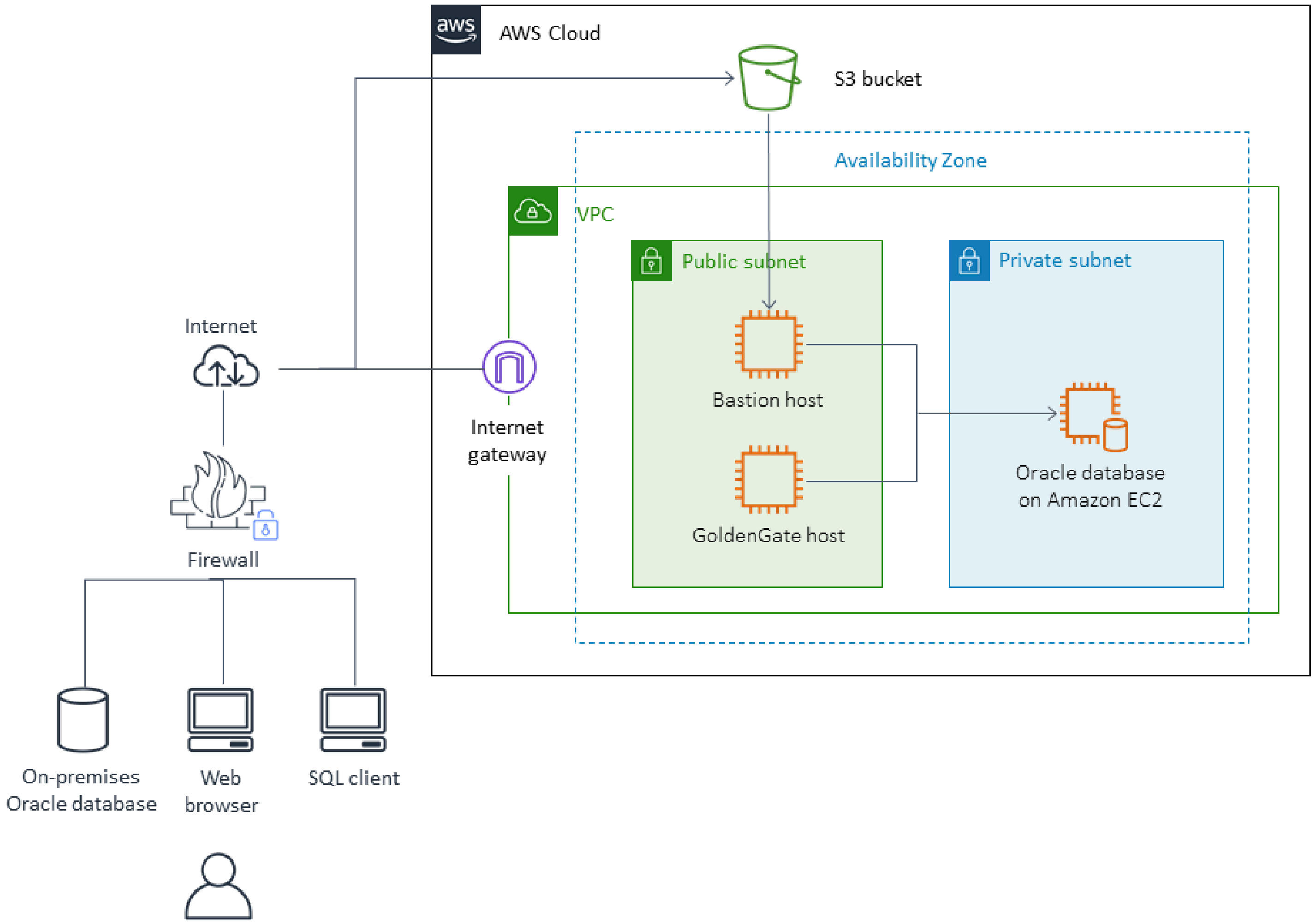


Data migration architecture

Using AWS DMS:



Using native Oracle tools:



Tools

- **AWS DMS -** [AWS Database Migration Services](https://aws.amazon.com/dms/) (AWS DMS) supports several types of source and target databases. For information about the database versions and editions that are supported, see [Using an Oracle Database as a Source for AWS DMS](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html). We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.
- **Native Oracle tools -** RMAN, Data Pump import/export, transportable tablespaces, Oracle GoldenGate

Epics

Plan the migration

Task	Description	Skills required
Validate the versions of the source and target databases.		DBA
Identify the version of the target OS.		DBA, SysAdmin
Identify hardware requireme nts for the target server instance based on the Oracle compatibility list and capacity requirements.		DBA, SysAdmin
Identify storage requirements (storage type and capacity).		DBA, SysAdmin
Identify network requireme nts (latency and bandwidth).		DBA, SysAdmin
Choose the proper instance type based on capacity, storage features, and network features.		DBA, SysAdmin
Identify network/host access security requirements for source and target databases.		DBA, SysAdmin
Identify a list of OS users required for Oracle software installation.		DBA, SysAdmin

Task	Description	Skills required
Download AWS Schema Conversion Tool (AWS SCT) and drivers.		DBA
Create an AWS SCT project for the workload, and connect to the source database.		DBA
Generate SQL files for the creation of objects (tables, indexes, sequences, etc.).		DBA
Determine a backup strategy.		DBA, SysAdmin
Determine availability requirements.		DBA
Identify the application migration/switch-over strategy.		DBA, SysAdmin, App owner

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC) and subnets in your AWS account.		SysAdmin
Create security groups and network access control lists (ACLs).		SysAdmin
Configure and start the EC2 instance.		SysAdmin

Install the Oracle software

Task	Description	Skills required
Create the OS users and groups required for the Oracle software.		DBA, SysAdmin
Download the required version of Oracle software.		
Install the Oracle software on the EC2 instance.		DBA, SysAdmin
Create objects like tables, primary keys, views, and sequences by using the scripts generated by AWS SCT.		DBA

Migrate data - option 1

Task	Description	Skills required
Use native Oracle tools or third-party tools to migrate database objects and data.	Oracle tools include Data Pump import/export, RMAN, transportable tablespaces, and GoldenGate.	DBA

Migrate data - option 2

Task	Description	Skills required
Determine the migration method.		DBA

Task	Description	Skills required
Create a replication instance in the AWS DMS console.		DBA
Create source and target endpoints.		DBA
Create a replication task.		DBA
Enable change data capture (CDC) to capture changes for a continuous replication.		DBA
Run the replication task and monitor logs.		DBA
Create secondary objects like indexes and foreign keys when the full load is done.		DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.		DBA, SysAdmin, App owner

Cut over

Task	Description	Skills required
Follow the application cutover/switch-over strategy.		DBA, SysAdmin, App owner

Close the project

Task	Description	Skills required
Shut down temporary AWS Secrets Manager resources.		DBA, SysAdmin
Review and validate the project documents.		DBA, SysAdmin, App owner
Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.		DBA, SysAdmin, App owner
Close out the project and provide feedback.		

Related resources

References

- Strategies for Migrating Oracle Databases to AWS
- Migrating Oracle databases to the AWS Cloud
- Amazon EC2 website
- AWS DMS website
- AWS DMS blog posts
- Amazon EC2 Pricing
- Licensing Oracle Software in the Cloud Computing Environment

Tutorials and videos

- Getting Started with Amazon EC2
- Getting Started with AWS DMS
- Introduction to Amazon EC2 - Elastic Cloud Server & Hosting with AWS (video)

Migrate an on-premises Oracle database to Amazon EC2 by using Oracle Data Pump

Navakanth Talluri, Amazon Web Services

Summary

When migrating databases, you must consider factors such as the source and target database engines and versions, migration tools and services, and acceptable downtime periods. If you're migrating an on-premises Oracle database to Amazon Elastic Compute Cloud (Amazon EC2), you can use Oracle tools, such as Oracle Data Pump and Oracle Recovery Manager (RMAN). For more information about strategies, see Migrating Oracle databases to the AWS Cloud.

Oracle Data Pump helps you extract the logical, consistent backup of the database and restore it to the target EC2 instance. This pattern describes how to migrate an on-premises Oracle database to an EC2 instance by using Oracle Data Pump and the `NETWORK_LINK` parameter, with minimal downtime. The `NETWORK_LINK` parameter starts an import through a database link. The Oracle Data Pump Import (impdp) client on the target EC2 instance connects to the source database, retrieves data from it, and writes the data directly to the database on the target instance. There are no backup, or *dump*, files used in this solution.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An on-premises Oracle database that:
  - Isn't an Oracle Real Application Clusters (RAC) database
  - Isn't an Oracle Automatic Storage Management (Oracle ASM) database
  - Is in read-write mode.
- You have created an AWS Direct Connect link between your on-premises data center and AWS. For more information, see Create a connection (Direct Connect documentation).

Product versions

- Oracle Database 10g release 1 (10.1) and later

Architecture

Source technology stack

- A standalone (non-RAC and non-ASM) Oracle database server in an on-premises data center

Target technology stack

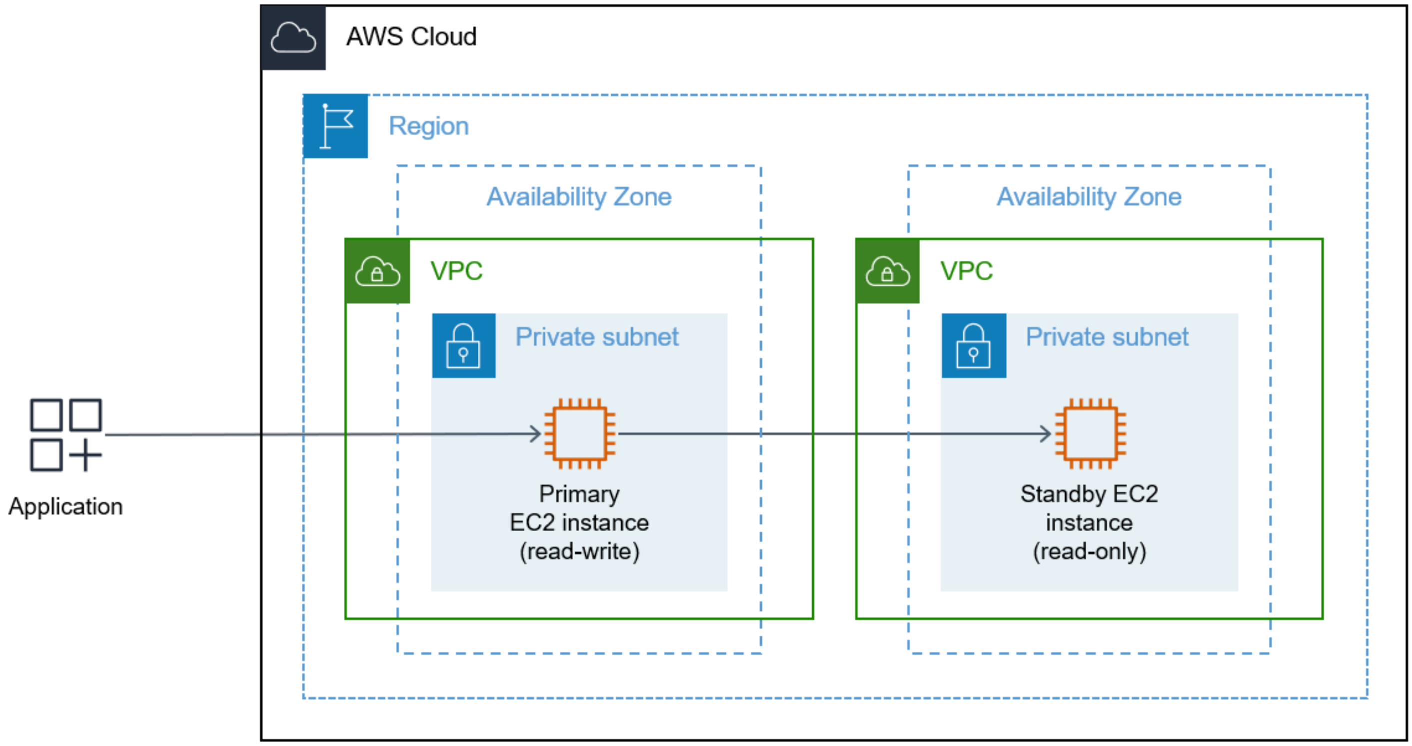
- An Oracle database running on Amazon EC2

Target architecture

The reliability pillar of the AWS Well-Architected Framework recommends creating data backups to help provide high availability and resiliency. For more information, see Architecting for high availability in *Best Practices for Running Oracle Database on AWS*. This pattern sets up primary and standby databases on EC2 instances by using Oracle Active Data Guard. For high availability, the EC2 instances should be in different Availability Zones. However, the Availability Zones can be in the same AWS Region or in different AWS Regions.

Active Data Guard provides read-only access to a physical standby database and applies redo changes continuously from the primary database. Based on your recovery point objective (RPO) and recovery time objective (RTO), you can choose between synchronous and asynchronous redo transport options.

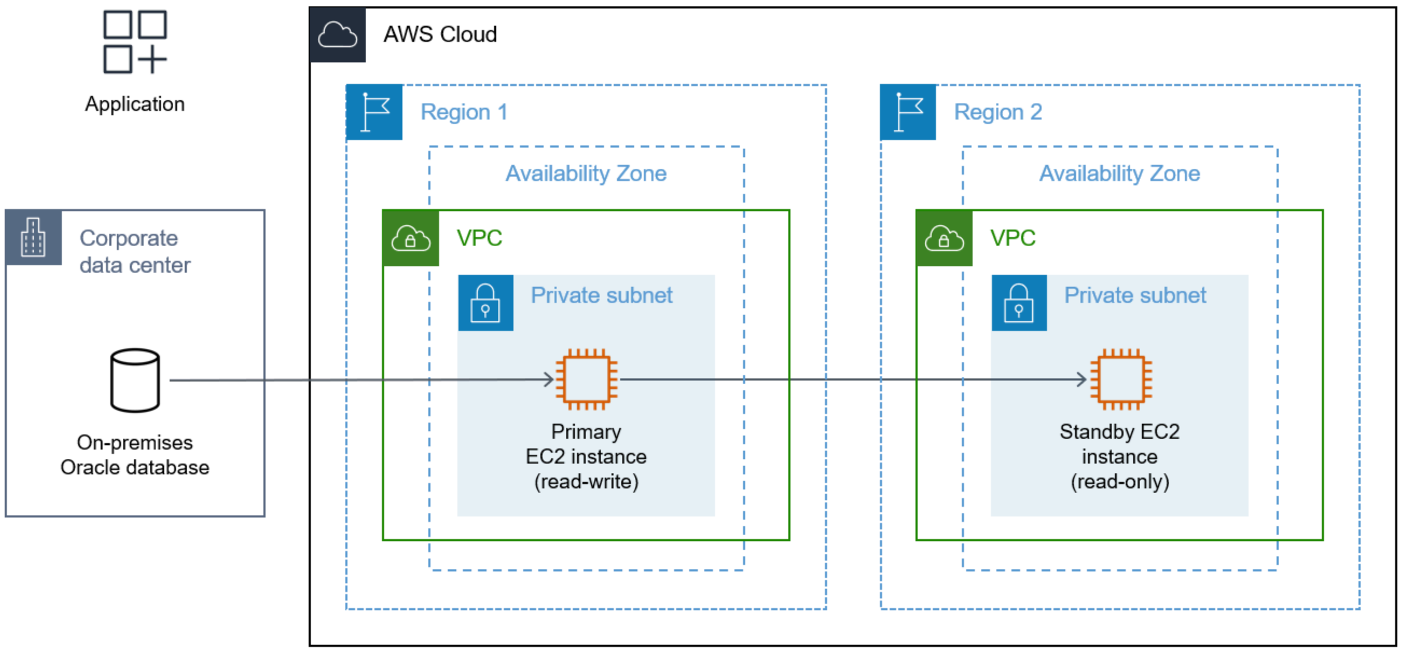
The following image shows the target architecture if the primary and standby EC2 instances are in different AWS Regions.



Data migration architecture

After you have finished setting up the target architecture, you use Oracle Data Pump to migrate the on-premises data and schemas to the primary EC2 instance. During cutover, applications can't access the on-premises database or the target database. You shut down these applications until they can be connected to the new target database on the primary EC2 instance.

The following image shows the architecture during the data migration. In this sample architecture, the primary and standby EC2 instances are in different AWS Regions.



Tools

AWS services

- AWS Direct Connect links your internal network to a Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.

Other tools and services

- Oracle Active Data Guard helps you create, maintain, manage, and monitor standby databases.
- Oracle Data Pump helps you move data and metadata from one database to another at high speeds.

Best practices

- Best Practices for Running Oracle Database on AWS
- Importing data using NETWORK_LINK

Epics

Set up the EC2 instances on AWS

Task	Description	Skills required
Identify the source hardware configuration for the on-premises host and the kernel parameters.	Validate the on-premises configuration, including storage size, input/output operations per second (IOPS), and CPU. This is important for Oracle licensing, which is based on CPU cores.	DBA, SysAdmin
Create the infrastructure on AWS.	Create the virtual private clouds (VPCs), private subnets, security groups, network access control lists (ACLs), route tables, and internet gateway. For more information, see the following : • VPCs and subnets • Tutorial: Create a VPC for use with a database instance	DBA, AWS systems administr ator
Set up the EC2 instances by using Active Data Guard.	Configure AWS EC2 instances by using an Active Data Guard configuration, as described in the AWS Well-Architected Framework. The version of Oracle Database on the EC2 instance can be different from the on-premises version because this pattern uses	DBA, AWS systems administr ator

Task	Description	Skills required
	logical backups. Note the following: • Put the target database in read-write mode. • On the target database, provide the Transpare nt Network Substrate (TNS) detail for the source database. For more information, see: • Starting Up a Database (Oracle documentation) • Creating and Configuring an Oracle Database (Oracle documentation)	

Migrate the database to Amazon EC2

Task	Description	Skills required
Create a dblink to the on-premises database from the EC2 instance.	Create a database link (dblink) between the Oracle database on the EC2 instance and the on-premises Oracle database. For more informati on, see Using Network Link Import to Move Data (Oracle documentation).	DBA

Task	Description	Skills required
Verify the connection between the EC2 instance and the on-premises host.	Use the dblink to confirm that the connection between the EC2 instance and the on-premises database is functioning. For instructions, see CREATE DATABASE LINK (Oracle documentation).	DBA
Stop all applications connected to the on-premises database.	After the database downtime is approved, shut down any applications and dependent jobs that connent to your on-premises database. You can do this either from the application directly or from the database by using cron. For more information, see Use the Crontab Utility to Schedule Tasks on Oracle Linux.	DBA, App developer
Schedule the data migration job.	On the target host, use the command `impdb` to schedule the Data Pump import. This connects the target database to the on-premises host and starts the data migration . For more information, see Data Pump Import and NETWORK_LINK (Oracle documentation).	DBA

Task	Description	Skills required
Validate the data migration.	Data validation is a crucial step. For data validation, you can use custom tools or Oracle tools, such as a combination of dblink and SQL queries.	DBA

Cut over

Task	Description	Skills required
Put the source database in read-only mode.	Confirm that the application is shut down and no changes are being made to the source database. Open the source database in read-only mode. This helps you avoid any open transactions. For more information, see `ALTER DATABASE` in [SQL Statements](#) (Oracle documentation).	DBA, DevOps engineer, App developer
Validate the object count and data.	To validate the data and object, use custom tools or Oracle tools, such as a combination of dblink and SQL queries.	DBA, App developer
Connect the applications to the database on the primary EC2 instance.	Change the application's connection attribute to point to the new database you created on the primary EC2 instance.	DBA, App developer

Task	Description	Skills required
Validate the application performance.	Start the application. Validate the functionality and performance of the applicati on by using Automated Workload Repository (Oracle documentation).	App developer, DevOps engineer, DBA

Related resources

AWS references

- Migrating Oracle databases to the AWS Cloud
- Amazon EC2 for Oracle
- Migrating bulky Oracle databases to AWS for cross-platform environments
- VPCs and subnets
- Tutorial: Create a VPC for use with a database instance

Oracle references

- Oracle Data Guard Configurations
- Data Pump Import

Migrate RHEL BYOL systems to AWS License-Included instances by using AWS MGN

Mike Kuznetsov, Amazon Web Services

Summary

When you migrate your workloads to AWS by using AWS Application Migration Service (AWS MGN), you might have to lift and shift (rehost) your Red Hat Enterprise Linux (RHEL) instances and change the license from the default Bring Your Own License (BYOL) model to an AWS License Included (LI) model during migration. AWS MGN supports a scalable approach that uses Amazon Machine Image (AMI) IDs. This pattern describes how to accomplish the license change on RHEL servers during the rehost migration at scale. It also explains how to change the license for a RHEL system that's already running on Amazon Elastic Compute Cloud (Amazon EC2).

Prerequisites and limitations

Prerequisites

- Access to the target AWS account
- AWS MGN initialized in the target AWS account and Region for the migration (not required if you have already migrated from your on-premises system to AWS)
- A source RHEL server with a valid RHEL license

Architecture

This pattern covers two scenarios:

- Migrating a system from on premises directly into an AWS LI instance by using AWS MGN. For this scenario, follow the instructions in the first epic (*Migrate to LI instance - option 1*) and third epic.
- Changing the licensing model from BYOL to LI for a previously migrated RHEL system that's already running on Amazon EC2. For this scenario, follow the instructions in the second epic (*Migrate to LI instance - option 2*) and third epic.

> **Note**
>
> The third epic involves reconfiguring the new RHEL instance to use the Red Hat Update Infrastructure (RHUI) servers provided by AWS. This process is the same for both scenarios.

Tools

AWS services

- AWS Application Migration Service (AWS MGN) helps you rehost (lift and shift) applications to the AWS Cloud without change and with minimal downtime.

Epics

Migrate to LI instance - option 1 (for an on-premises RHEL system)

Task	Description	Skills required
Find the AMI ID of the RHEL AWS LI instance in the target Region.	Visit AWS Marketplace or use the Amazon EC2 console to find the RHEL AMI ID that matches the version of the RHEL source system (for example, RHEL-7.7), and write down the AMI ID. On the Amazon EC2 console, you can filter the AMIs by using one of the following search terms: • Description = Provided by Red Hat, Inc. • AMI name = RHEL-7.7	Cloud administrator
Configure AWS MGN launch settings.	1. On the AWS MGN console, add the source RHEL system: Install the AWS Replication Agent and	Cloud administrator

Task	Description	Skills required
	add the source server by following the instructi ons in the [AWS MGN documentation](). 2. On the **Source servers** page, choose the source RHEL system, and then choose the **Launch settings** tab. 3. In the **General launch settings** section, choose **Edit**. To disable automatic selection and manually specify the target instance type, change **Instance type right sizing** to **None**, and then choose **Save settings**. This lets you use the instance type that you configure in your Amazon EC2 launch template. For more information, see the [AWS MGN documentation](). 4. In the **EC2 Launch Template** section, choose **Modify**. In the **About modifying EC2 launch templates** dialog box, choose **Modify** again. This opens the Amazon EC2 console so you can change the template for this instance.	

Task	Description	Skills required
	5. Review the key considerations in the [AWS MGN documentation](). **Note** You can disregard the warning against choosing your own AMI. 6. On the [Amazon EC2 console](), in the new launch template, modify the following: • For **AMI**, specify the AMI ID you identified previously, or search for RHEL-*x* and specify the version you require (for example, RHEL-7.7). • For **Instance type**, set the desired target instance type. • Leave the following sections unchanged: **Key pair (login)**, **Network settings** (unless you want to specify a target subnet and security groups), **Storage**, **Resource tags** (unless you want to add or modify any tags).	

Task	Description	Skills required
	• (Optional) In the **Advanced details** section, specify the IAM instance profile role, if needed for future management by AWS Systems Manager. 7. Choose **Create template version**, and then choose the link in the success message to view the launch template. 8. Choose **Actions**, **Set default version**. For **Template version**, select the latest version (version 2 for a new system), and then choose **Set as default version**. AWS MGN will now use this version of the launch template to launch test or cutover instances. For more information, see the [AWS MGN documentation]().	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Validate settings.</td><td>

On the AWS MGN console, on the Source servers page, choose your source server, and then choose the Launch settings tab.
In the EC2 Launch Template section, verify that the Instance type, Subnet, and Security groups parameters are set correctly.

Note
This section doesn't display the AMI ID you selected. To see the ID, you can open the Amazon EC2 console, Launch Templates view, and search for the template ID that's shown in this section.
</td><td>Cloud administrator</td></tr>
</table>

Task	Description	Skills required
Launch the new LI instance.	1. When the initial sync is complete, the **Migration lifecycle** column for the server on the AWS MGN console **Source servers** page changes to **Ready for testing**. To launch the new test instance, choose your source server, open the **Test and cutover** menu, and then choose **Launch test instances**. Choose **View job details** to monitor the status of the launch job. For more information, see the [AWS MGN documentation](). 2. Wait for the launch job to complete, and then open the launched EC2 instance details page. Choose the **Details** tab and verify that the **Instance details** section contains the following: • Platform details: "Red Hat Enterprise Linux" • AMI name: The name of AMI you specified in the EC2 launch template 3. Cut over to the new LI instance by following the instructions in the [AWS MGN documentation]().	Cloud administrator

Task	Description	Skills required
	4. Reconfigure the new instance to use the AWS-provided RHUI servers by following the steps in the last epic.	

Migrate to LI instance - option 2 (for a RHEL BYOL EC2 instance)

Task	Description	Skills required
Migrate your RHEL BYOL EC2 instance to an AWS LI instance.	You can switch RHEL systems that you previously migrated to AWS as BYOL to AWS LI instances by moving their disks (Amazon Elastic Block Store volumes) and attaching them to a new LI instance. To make this switch, follow these steps: 1. Launch a new target RHEL instance from a RHEL LI AMI. Make sure that the AMI you selected: • Uses the same RHEL version as your current RHEL instance. • Has the same boot-up process (BIOS or UEFI) as your current RHEL instance. For example, if the source server is BIOS-based, use the AWS Marketplace RHEL AMI	Cloud administrator

Task	Description	Skills required
	that is also BIOS-based; for UEFI-based systems, choose the UEFI-based AMI. 2. Stop both instances: the new LI instance and the original source instance. 3. Detach all EBS volumes (including the root disk) from the new LI instance and delete them. 4. Detach all EBS volumes (including the root disk) from the old source instance and attach them to the new LI instance. Keep the same mapping of volumes to devices. (For example, the EBS volume that was previousl y attached to the `/dev/sda` drive must be attached as `/dev/sda` to the new instance.) 5. Delete the source (now diskless) instance. 6. Start the new LI instance. Log in to the instance and reconfigure it to use the AWS-provided RHUI servers by following the steps in the next epic.	

Reconfigure RHEL OS to use AWS-provided RHUI – both options

Task	Description	Skills required
Deregister the OS from the Red Hat subscription and license.	After migration and successful cutover, the RHEL system has to be removed from the Red Hat subscription to stop consuming the Red Hat license and avoid double billing. To remove RHEL OS from the Red Hat subscription, follow the process described in the [Red Hat Subscription Management (RHSM) documentation](). Use the CLI command: `subscription-manager unregister` You can also disable the subscription manager plugin to stop checking the status of the subscription on every **yum** call. To do this, edit the configuration file `/etc/yum/pluginconf.d/subscription-manager.conf` and change the parameter `enabled=1` to `enabled=0` .	Linux or system administrator
Replace the old update configuration (RHUI, Red	You must reconfigure the migrated RHEL system	Linux or system administrator

Task	Description	Skills required
Hat Satellite network, yum repositories) with the AWS-provided RHUI.	to use the AWS-provided RHUI servers. This gives you access to the RHUI servers within AWS Regions without requiring the external update infrastructure. The change involves the following process: 1. Back up the existing **yum** configuration. 2. Remove the old RHUI (**yum** repositories) configuration and packages. 3. Add the new AWS-provided RHUI configuration and certificate packages. You have to retrieve these from another RHEL instance on AWS because these configuration packages are available only on AWS-provided RHUI servers. Here are the detailed steps and commands: 1. Back up the existing **yum** configuration and certificates by copying all `/etc/yum*` and `/etc/pki/*` folders to a backup location. For example: `mkdir yum-backup`	

Task	Description	Skills required

	`cp -ra /etc/yum* /etc/pki ./yum-backup`
	`tar czf yum-backup.tgz ./yum-backup`
	2. Remove the old RHUI configuration and packages:
	a. Find all installed RHUI packages:
	`sudo rpm -qa | grep rhui`
	b. Delete these packages:
	`sudo yum remove $(rpm -qa | grep rhui)`
	c. Remove the `/etc/yum/vars/releasever` file, if it exists.
	3. Add the new AWS-provided RHUI and certificate packages. You must retrieve these from another RHEL instance on AWS. There are several ways to do this. For example, you can follow the instructions provided in the [Red Hat Knowledgebase article](#):
	a. Launch another RHEL (RHEL-EC2) instance from [AWS Marketplace](#).	

Task	Description	Skills required

	b. Download two packages from this instance: the latest RHUI client configuration package and the certificate authority (CA) certificates. For example, run this command from your desktop:
	`ssh RHEL-EC2 "sudo yumdownloader ca-certificates rh-amazon-rhui-client"`
	c. Copy the packages from the RHEL-EC2 instance to the new migrated system. For example:
	`scp RHEL-EC2:rh-amazon-rhui-client\* RHEL-EC2:ca-certificates\* .`
	`ssh <migrated-instance> "mkdir /tmp/amazon"`
	`scp rh-amazon-rhui-client* ca-certificates* <migrated-instance>:/tmp/amazon`
	d. Install the new RHUI and CA configuration packages on the migrated instance:	

Task	Description	Skills required
	`ssh <migrated-instance> "sudo rpm -Uhv /tmp/amazon/*"`	

Validate the configuration.	On the target migrated instance, verify that the new configuration is correct:
	`sudo yum clean all`
	`sudo yum repolist`
		Linux or system administrator

Related resources

- AWS Application Migration Service (AWS MGN) User Guide
- Get an AWS RHUI client package supporting IMDSv2 (Red Hat Knowledgebase article)
- Amazon EC2 launch templates (Amazon EC2 documentation)

Migrate an on-premises SAP ASE database to Amazon EC2

Sergey Dmitriev and Gergely Cserdi, Amazon Web Services

Summary

This pattern describes how to migrate an SAP Adaptive Server Enterprise (ASE) database from an on-premises host to an Amazon Elastic Compute Cloud (Amazon EC2) instance. The pattern covers the use of AWS Database Migration Service (AWS DMS) or SAP ASE native tools such as ASE Cockpit, Sybase Central for ASE, and DBA Cockpit for migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An SAP ASE source database in an on-premises data center

Limitations

- The source database must be less than 64 TB

Product versions

- SAP ASE version 15.x and 16.x or later

Architecture

Source technology stack

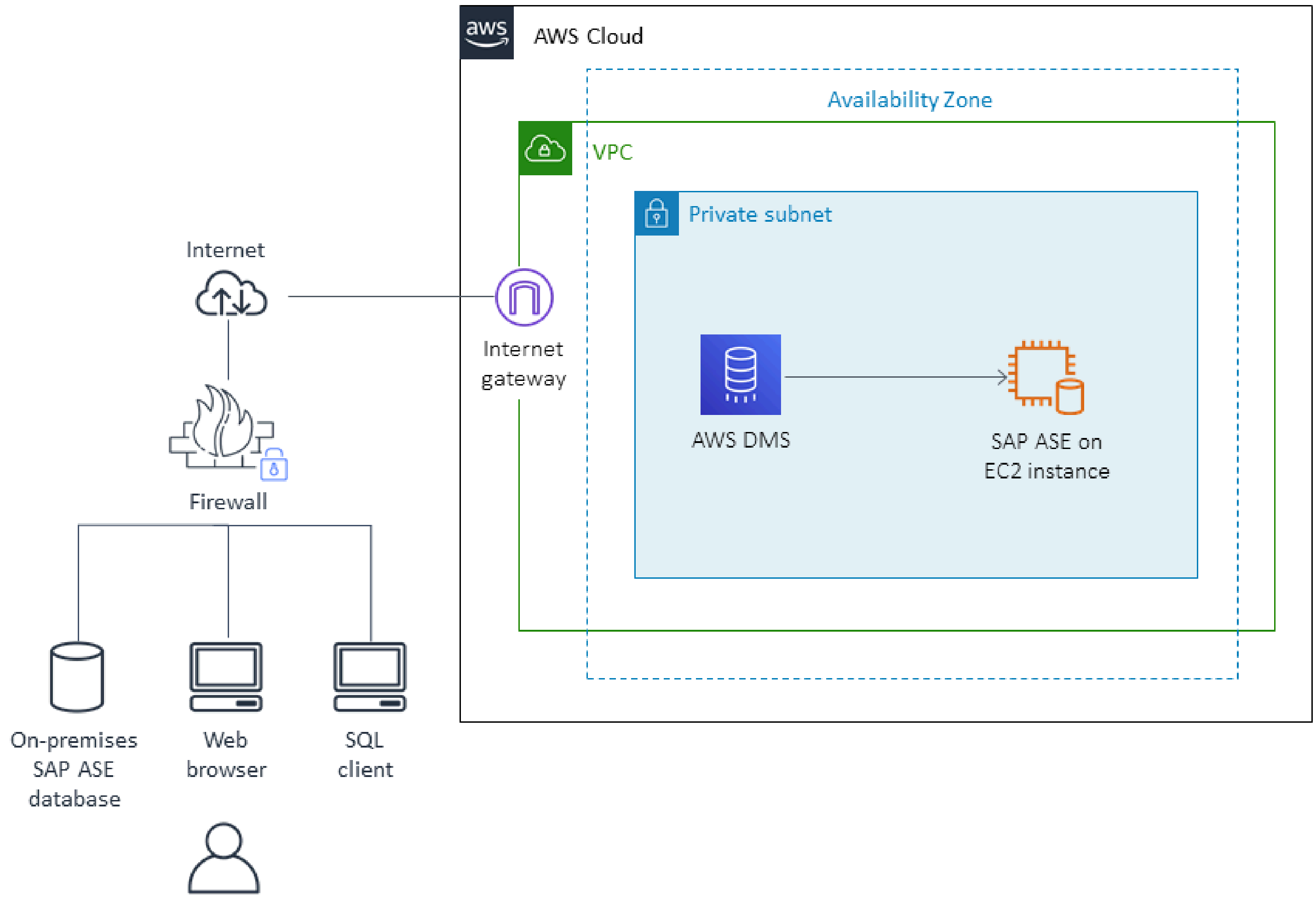
- On-premises SAP ASE database

Target technology stack

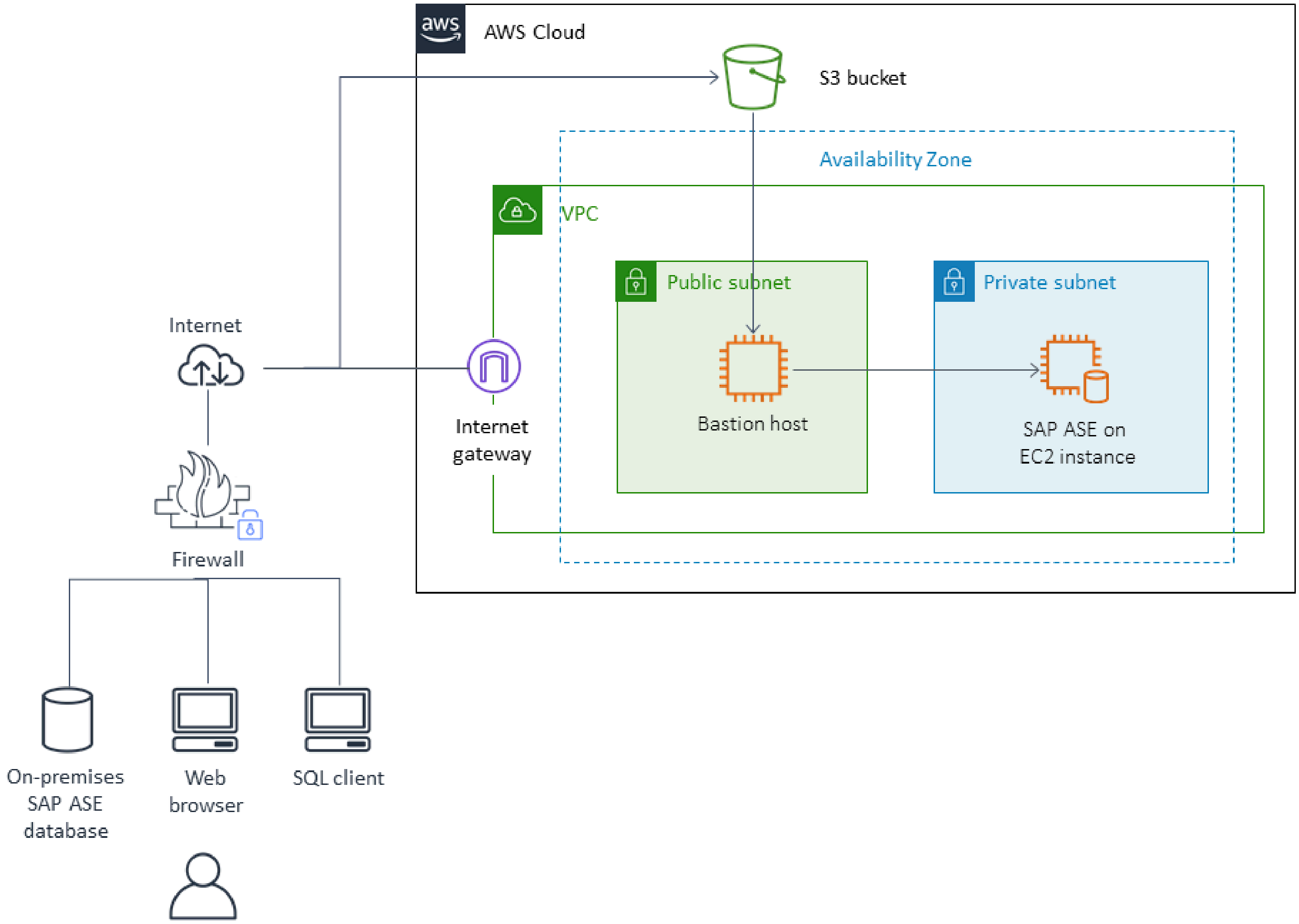
- SAP ASE database on an EC2 instance

Database migration architecture

Using AWS DMS:



Using native SAP ASE tools:



Tools

- **AWS DMS** - AWS Data Migration Service (AWS DMS) supports several different source and target databases. For more information, see Sources for Data Migration and Targets for Data Migration. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.
- **SAP ASE** - Native tools include ASE Cockpit, Sybase Central for ASE, and DBA Cockpit.

Epics

Analyze the migration

Task	Description	Skills required
Validate the source and target database versions.		DBA
Identify the target OS version.		DBA, SysAdmin
Identify the hardware requirements for the target server instance based on the SAP ASE compatibility list and capacity requirements.		DBA, SysAdmin
Identify the requirements for the storage type and capacity.		DBA, SysAdmin
Identify the network requirements including latency and bandwidth.		DBA, SysAdmin
Choose the proper instance type, capacity, storage features, and network features.		DBA, SysAdmin
Identify the network and host access security requireme nts for the source and target databases.		DBA, SysAdmin
Identify a list of operating system users required for the SAP ASE software installation.		DBA, SysAdmin

Task	Description	Skills required
Determine the backup strategy.		DBA
Determine the availability requirements.		DBA
Identify the application migration and switchover strategy.		DBA, SysAdmin, App owner

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC) and subnets.		SysAdmin
Create security groups and the network access control list (ACL).		SysAdmin
Configure and start the EC2 instance.		SysAdmin

Install the software

Task	Description	Skills required
Create the OS users and groups required for SAP ASE software to work.		DBA, SysAdmin
Download the required version of SAP ASE software.		DBA, SysAdmin

Task	Description	Skills required
Install the SAP ASE database, backup server software, and replication server software on the EC2 instance and then configure the server.		DBA, SysAdmin

Migrate the data - option 1

Task	Description	Skills required
Migrate the database objects and data by using native SAP ASE tools or third-party tools.	See the documentation for the SAP ASE or third-par ty tools. These include ASE Cockpit, Sybase Central for ASE, and DBA Cockpit.	DBA

Migrate the data - option 2

Task	Description	Skills required
Migrate the data by using AWS DMS.		DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.		DBA, SysAdmin, App owner

Cut over

Task	Description	Skills required
Follow the application cutover or switchover strategy.		DBA, SysAdmin, App owner

Close the project

Task	Description	Skills required
Shut down the temporary AWS resources.		DBA, SysAdmin
Validate and review the project documents.		DBA, SysAdmin, App owner
Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.		DBA, SysAdmin, App owner
Close the project and provide any feedback.		DBA, SysAdmin, App owner

Related resources

References

- Amazon EC2
- AWS DMS
- Amazon EC2 pricing

Tutorials and videos

- Getting Started with Amazon EC2
- Getting Started with AWS Database Migration Service
- AWS Data Migration Service (video)
- Introduction to Amazon EC2 - Elastic Cloud Server & Hosting with AWS (video)

Migrate an on-premises Microsoft SQL Server database to Amazon EC2

Senthil Ramasamy, Amazon Web Services

Summary

This pattern describes how to migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on an Amazon Elastic Compute Cloud (Amazon EC2) instance. It covers two options for migration: using AWS Database Migration Service (AWS DMS) or using native Microsoft SQL Server tools such as backup and restore, Copy Database Wizard, or copy and attach database.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An operating system supported by Amazon EC2 (for a complete list of supported operating system versions, see [Amazon EC2 FAQs])
- A Microsoft SQL Server source database in an on-premises data center

Product versions

- For on-premises and Amazon EC2 instance databases, AWS DMS supports:
  - SQL Server versions 2005, 2008, 2008R2, 2012, 2014, 2016, 2017, and 2019
  - Enterprise, Standard, Workgroup, Developer, and Web editions
- For the latest list of supported versions, see [Using a Microsoft SQL Server Database as a Target for AWS DMS].

Architecture

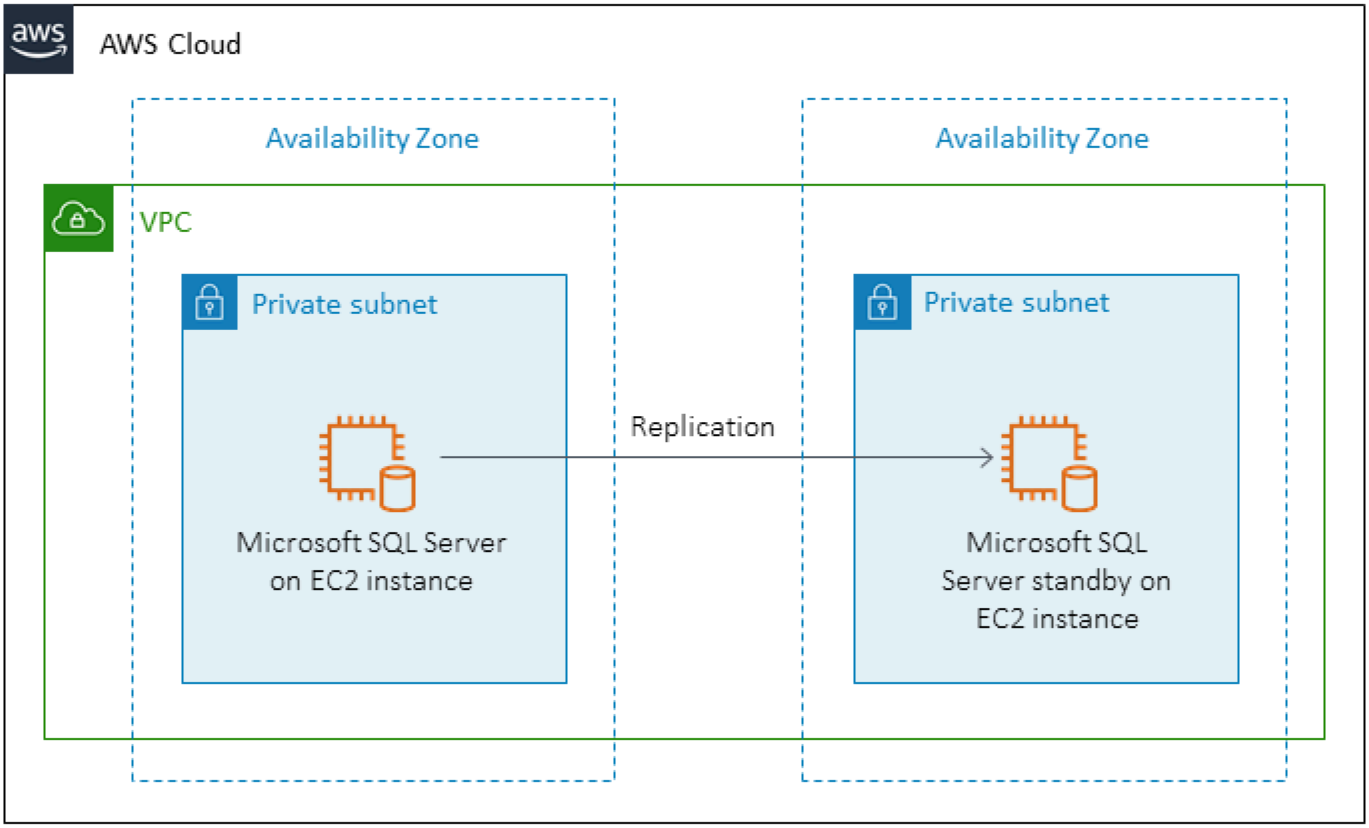
Source technology stack

- On-premises Microsoft SQL Server database

Target technology stack

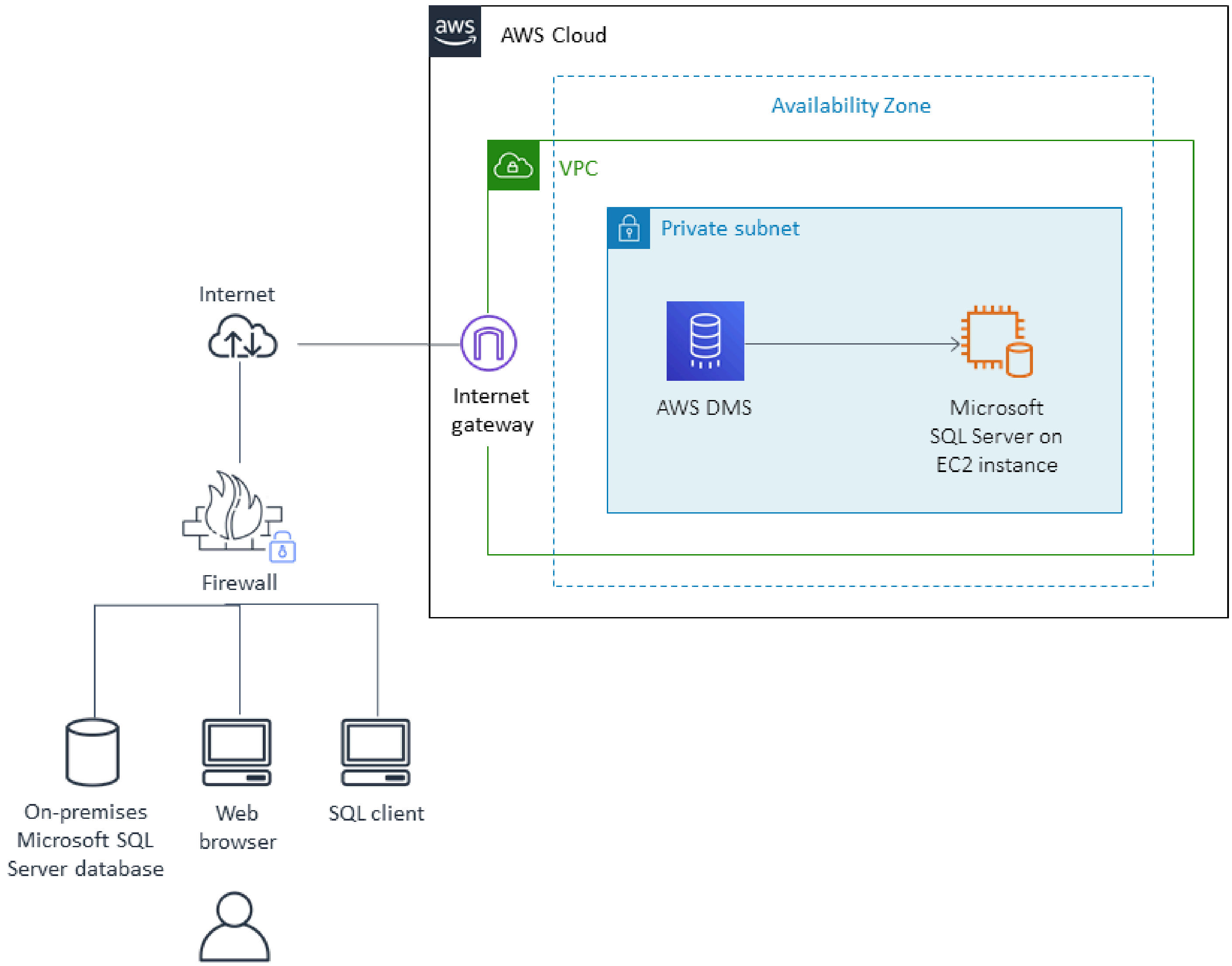
- Microsoft SQL Server database on an EC2 instance

Target architecture

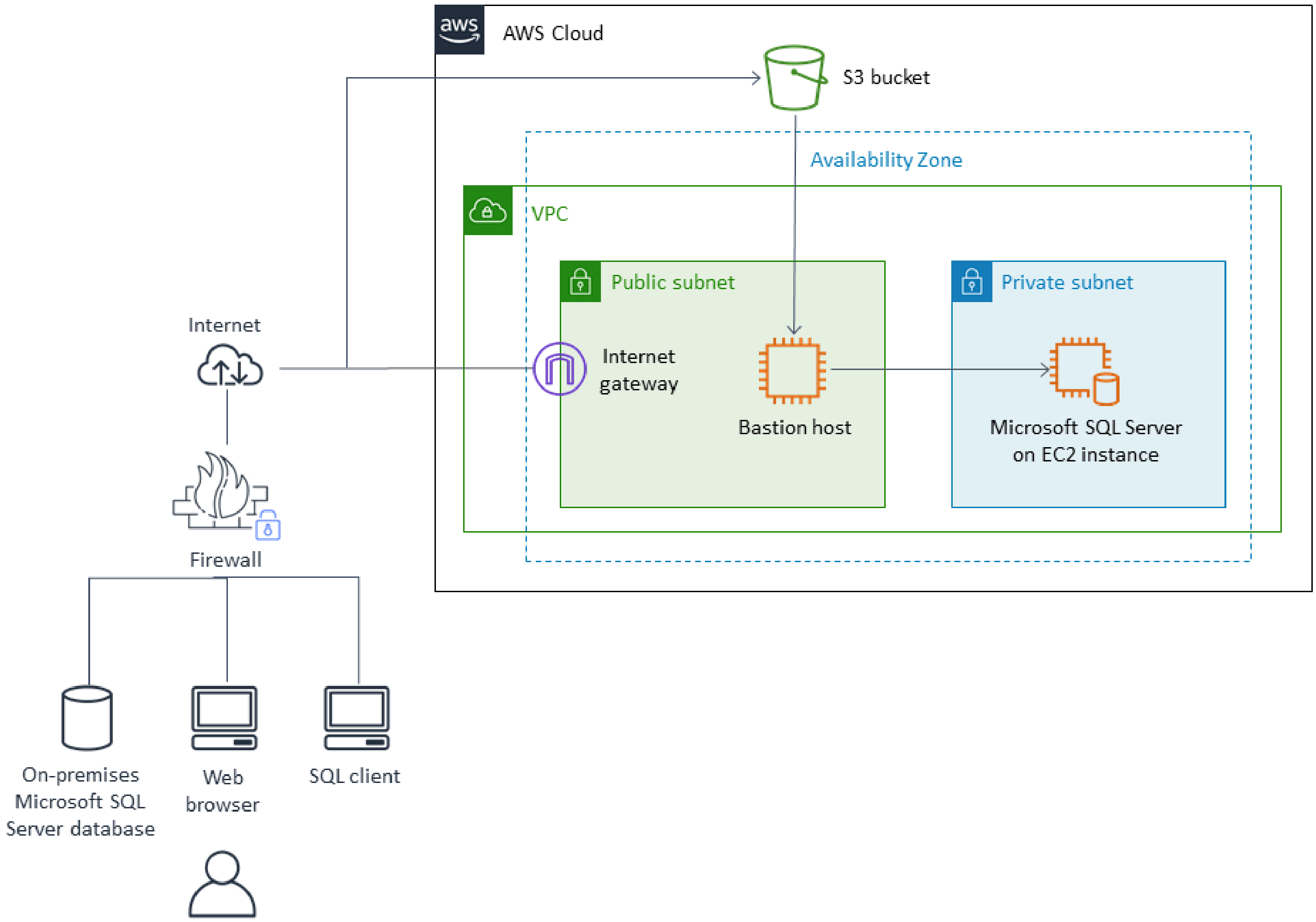


Data migration architecture

- Using AWS DMS



- Using native SQL Server tools



Tools

- [AWS Database Migration Service (AWS DMS)] helps you migrate your data to and from widely used commercial and open-source databases, including Oracle, SQL Server, MySQL, and PostgreSQL. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups.
- [AWS Schema Conversion Tool (AWS SCT)] supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.
- Native Microsoft SQL Server tools include backup and restore, Copy Database Wizard, and copy and attach database.

Epics

Plan the migration

Task	Description	Skills required
Validate the source and target database versions.		DBA
Identify the target operating system version.		DBA, Systems administrator
Identify the hardware requirements for the target server instance based on the Microsoft SQL Server compatibility list and capacity requirements.		DBA, Systems administrator
Identify the storage requirements for type and capacity.		DBA, Systems administrator
Identify the network requirements, including latency and bandwidth.		DBA, Systems administrator
Choose the EC2 instance type based on capacity, storage features, and network features.		DBA, Systems administrator
Identify the network and host access security requireme nts for the source and target databases.		DBA, Systems administrator
Identify a list of users required for the Microsoft		DBA, Systems administrator

Task	Description	Skills required
SQL Server software installat ion.		
Determine the backup strategy.		DBA
Determine the availability requirements.		DBA
Identify the application migration and cutover strategy.		DBA, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC) and subnets.		Systems administrator
Create security groups and network access control list (ACL).		Systems administrator
Configure and start an EC2 instance.		Systems administrator

Install the software

Task	Description	Skills required
Create the users and groups required for Microsoft SQL Server software.		DBA, Systems administrator

Task	Description	Skills required
Download the Microsoft SQL Server software.		DBA, Systems administrator
Install Microsoft SQL Server software on the EC2 instance and configure the server.		DBA, Systems administrator

Migrate the data - option 1

Task	Description	Skills required
Use native Microsoft SQL Server tools or third-party tools to migrate the database objects and data.	Tools include backup and restore, Copy Database Wizard, and copy and attach database. For more information, see the guide [Migrating Microsoft SQL Server databases to the AWS Cloud]().	DBA

Migrate the data - option 2

Task	Description	Skills required
Migrate the data by using AWS DMS.	For more information about using AWS DMS, see the links in the [Related resources]() section.	DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.	Use AWS Schema Conversion Tool (AWS SCT) to analyze and modify SQL code that’s embedded in application source code.	DBA, App owner

Cut over

Task	Description	Skills required
Follow the application switch-over strategy.		DBA, App owner, Systems administrator

Close the project

Task	Description	Skills required
Shut down all temporary AWS resources.	Temporary resources include the AWS DMS replication instance and the EC2 instance for AWS SCT.	DBA, Systems administrator
Review and validate the project documents.		DBA, App owner, Systems administrator
Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.		DBA, App owner, Systems administrator
Close the project and provide feedback.		DBA, App owner, Systems administrator

Related resources

References

- Migrating Microsoft SQL Server databases to the AWS Cloud
- Amazon EC2
- Amazon EC2 FAQs
- Amazon EC2 pricing
- AWS Database Migration Service
- Microsoft Products on AWS
- Microsoft Licensing on AWS
- Microsoft SQL Server on AWS

Tutorials and videos

- Getting Started with Amazon EC2
- Getting Started with AWS Database Migration Service
- Join an Amazon EC2 instance to your Simple AD Active Directory
- Join an Amazon EC2 instance to your AWS Managed Microsoft AD Active Directory
- AWS Database Migration Service (video)
- Introduction to Amazon EC2 – Elastic Cloud Server & Hosting with AWS (video)

Migrate an on-premises MySQL database to Amazon EC2

Lorenzo Mota, Amazon Web Services

Summary

This pattern provides guidance for migrating an on-premises MySQL database to a MySQL database on an Amazon Elastic Compute Cloud (Amazon EC2) instance. The pattern discusses the use of AWS Database Migration Service (AWS DMS) or native MySQL tools such as **mysqldump** for the migration. It focuses on full database migration to a MySQL DB instance.

The pattern is primarily for DBAs and solutions architects. It can be used in small or large projects, in a testing or final migration phase. We recommend that you run at least one testing cycle before you use this pattern in a production environment.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A MySQL source database in an on-premises data center

Product versions

- MySQL version 5.5 and later
- A target operating system supported by Amazon EC2; see [Amazon EC2 FAQs]

Architecture

Source technology stack

- An on-premises MySQL database

Target technology stack

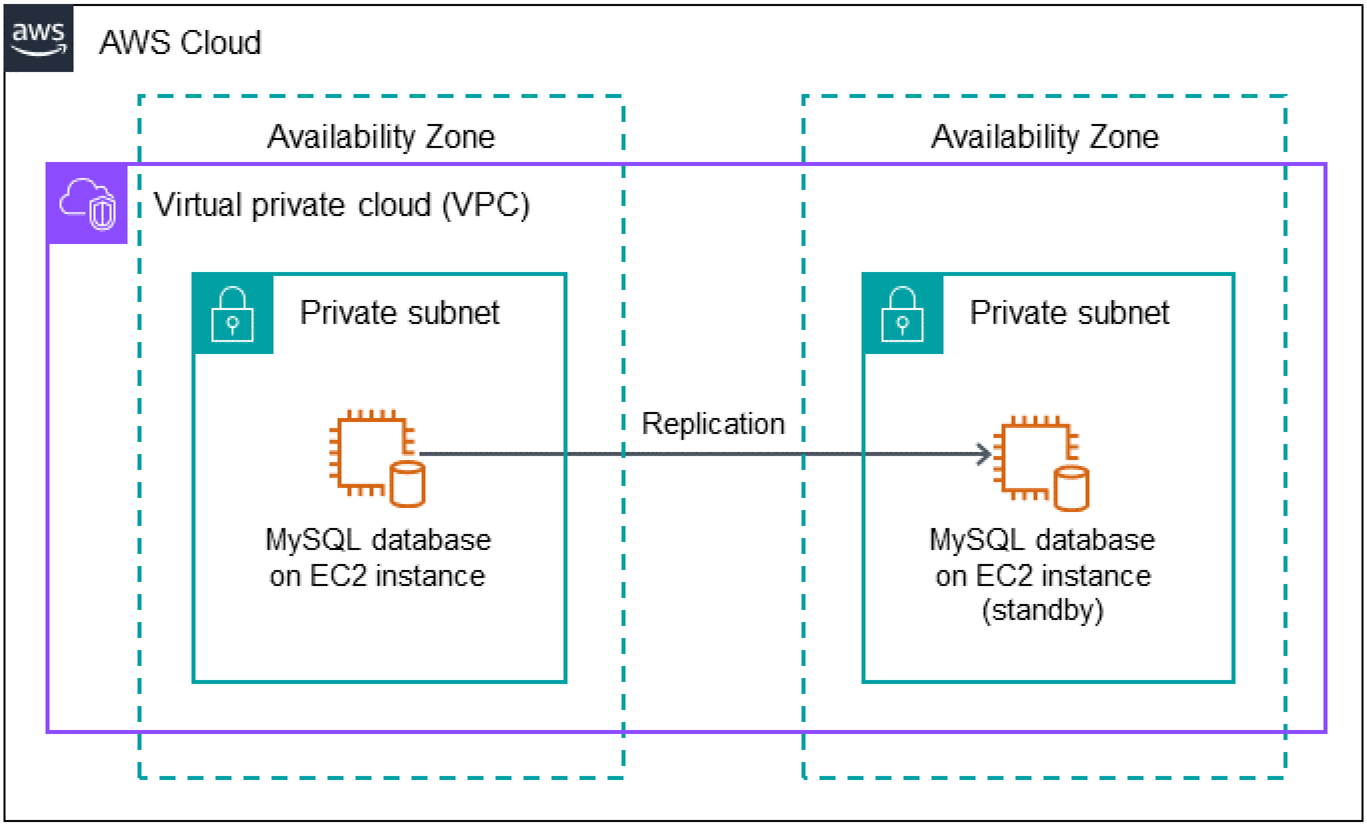
- A MySQL database instance on Amazon EC2

AWS data migration methods

- AWS DMS
- Native MySQL tools such as mysqldump, or third-party tools such as Percona XtraBackup

Target architecture

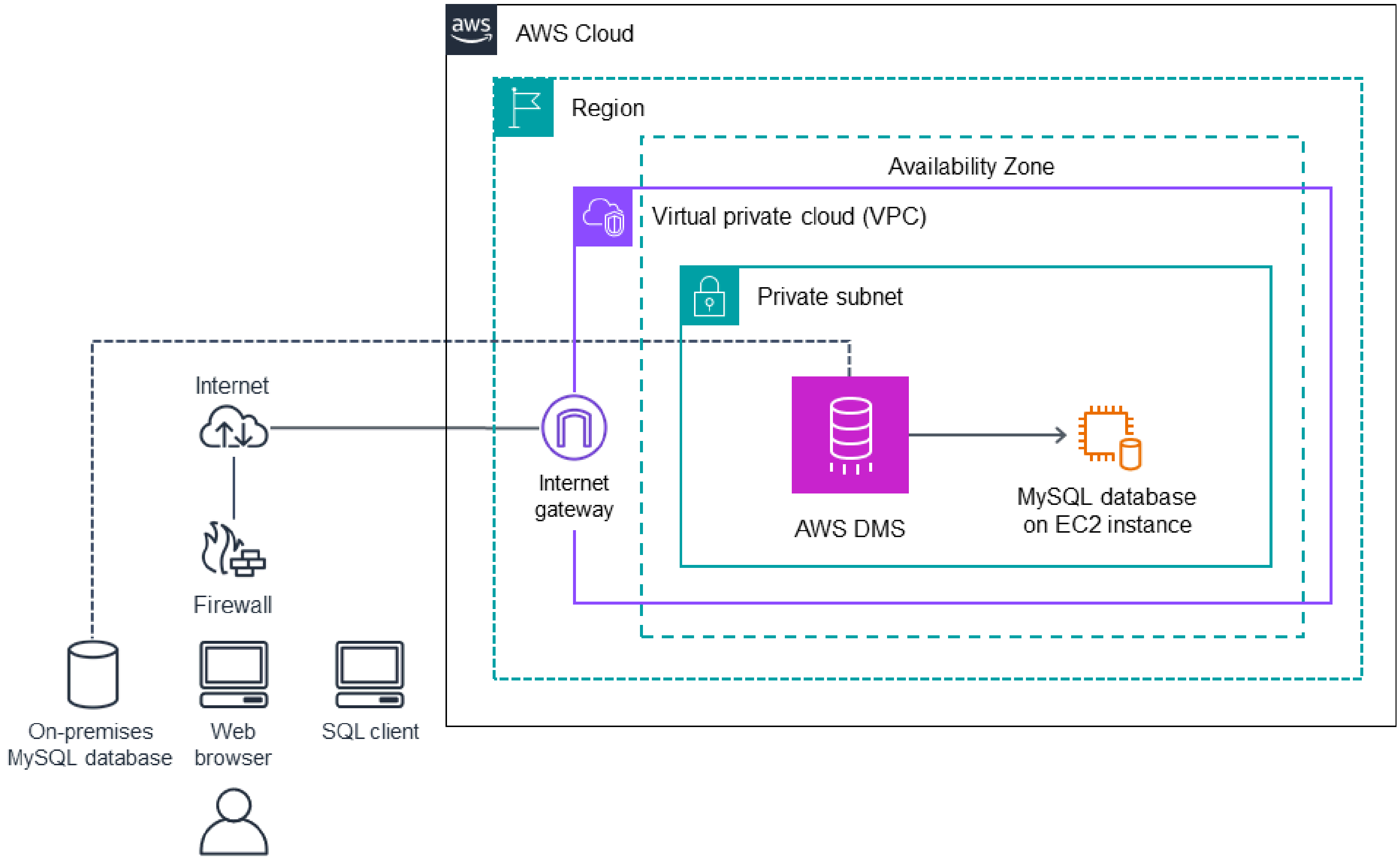
The following diagram illustrates the target Amazon EC2 implementation after cutover.



AWS data migration architecture

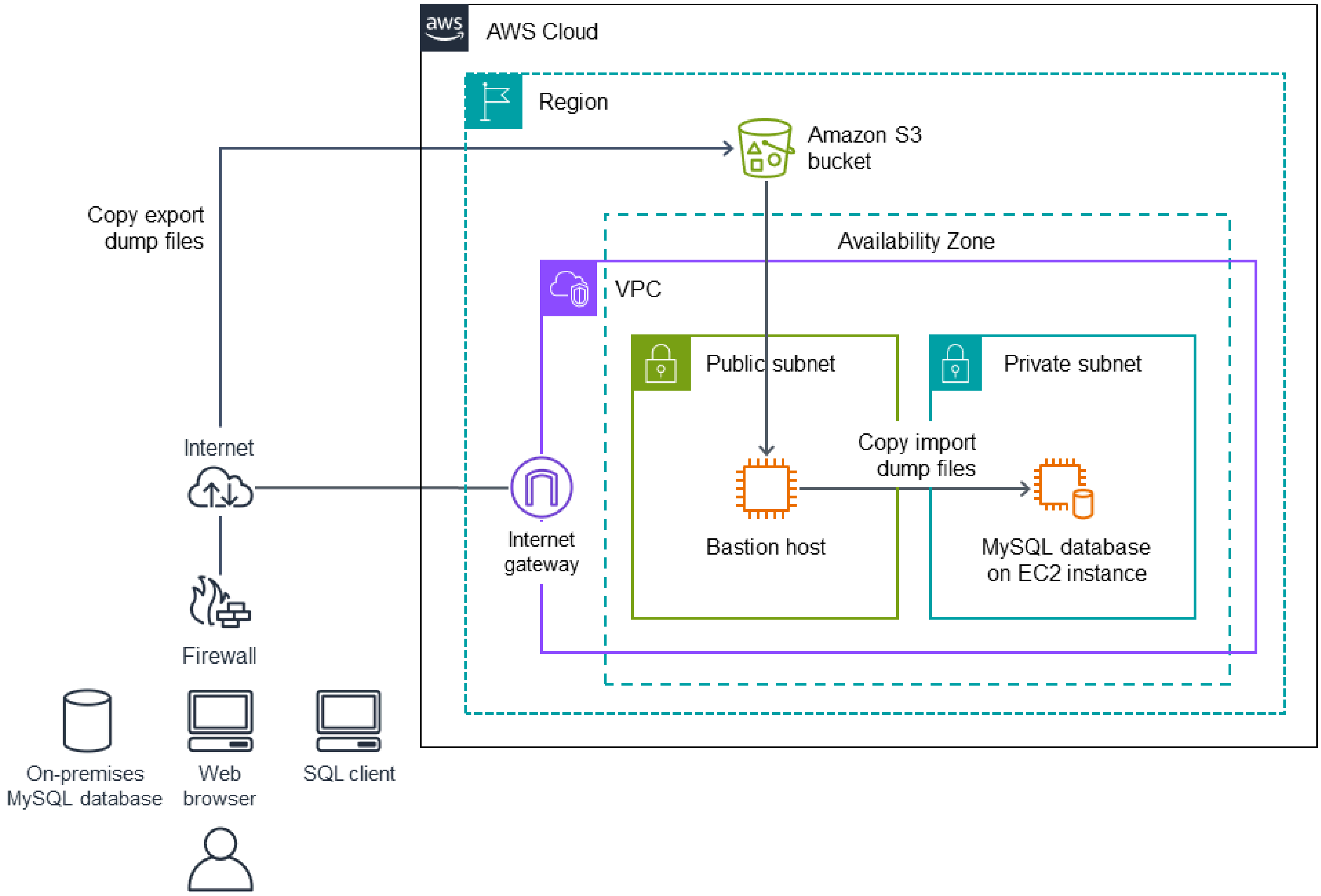
Using AWS DMS:

The following diagram illustrates the data migration workflow based on AWS DMS for sending full and incremental changes to the target MySQL database until cutover. The network connection from on premises to AWS depends on the requirements of the SQL client and is outside the scope of this pattern.



Using other MySQL tools:

The following diagram illustrates the data migration workflow based on using MySQL tools to generate export dump files from the on-premises database. These files are moved to Amazon Simple Storage Service (Amazon S3) and imported into the target MySQL database before cutover. The network connection from on premises to AWS depends on the requirements of the SQL client and is outside the scope of this pattern.



Notes:

- Depending on down time considerations and the size of the database for the final cutover, you can use AWS DMS or another change data capture (CDC) tool to minimize the cutover time. When you use a CDC tool such as AWS DMS, you can migrate to the target database in minutes.
- An offline strategy with **mysqldump** can suffice if the size of the database and network latency allow for a short cutover migration window. (We recommend that you perform testing to get an approximate time.)
- Usually a CDC strategy through AWS DMS requires more monitoring and complexity than an offline option.

Tools

AWS services

- AWS Database Migration Service (AWS DMS) supports several source and target databases. For information about MySQL source and target databases supported by AWS DMS, see Using a MySQL-compatible database as a source for AWS DMS and Using a MySQL-compatible database as a target for AWS DMS. If your source database isn't supported by AWS DMS, you must choose another method to migrate your data.

Other tools

- mysqldump is a MySQL utility that creates a dump file from a MySQL database for backup or migration purposes.
- Percona XtraBackup is an open source utility for performing non-blocking backups on MySQL databases.

Epics

Plan the migration

Task	Description	Skills required
Validate the database versions.	Validate the versions of the source and target databases . For information about MySQL versions supported by AWS DMS, see Sources for AWS DMS and Targets for AWS DMS in the AWS DMS documentation.	DBA
Identify the target operating system.	Determine the version of the target operating system. For a list of target operating systems supported by Amazon EC2, see Amazon EC2 FAQs.	DBA, Systems administrator
Identify hardware requireme nts.	Determine the hardware requirements for the target	DBA, Systems administrator

Task	Description	Skills required
	server instance based on the MySQL compatibility list and capacity requirements.	
Identify storage requireme nts.	Determine the storage type and capacity for the target database.	DBA, Systems administrator
Identify network requireme nts.	Determine networking requirements such as latency and bandwidth.	DBA, Systems administrator
Choose the target instance type.	Choose the target instance type based on capacity, storage features, and network features.	DBA, Systems administrator
Identify security requireme nts.	Determine the network or host access security requirements for the source and target databases.	DBA, Systems administrator
Identify users.	Determine the list of operating system users for the MySQL software installat ion. For more information, see the MySQL documentation.	DBA, Systems administrator
Determine a backup strategy.		DBA
Determine availability requirements.		DBA
Identify the application migration or switchover strategy.		DBA, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC) and subnets.	Configure route tables, internet gateway, NAT gateways, and subnets. For more information, see [VPC configuration options] in the Amazon VPC documentation.	Systems administrator
Create security groups and network access control lists (ACLs).	Configure ports (default for MySQL is 3306) and CIDR ranges or specific IPs depending on your requireme nts.	Systems administrator
Configure and start an EC2 instance.	For instructions, see [Launch an EC2 instance] in the Amazon EC2 documentation.	Systems administrator

Install MySQL software

Task	Description	Skills required
Create users and groups.	Create the operating system users and groups who need access to the server and database. For more informati on, see [Access Control and Account Management] in the MySQL documentation.	DBA, Systems administrator
Download MySQL.	Download the MySQL software. For instructions and binaries, see [Installing MySQL] in the MySQL documentation.	DBA, Systems administrator

Task	Description	Skills required
Install MySQL on the EC2 instance and configure the server.	Connect to your EC2 instance and install the MySQL software. For more informati on, see Connect to your EC2 instance in the Amazon EC2 documentation.	DBA, Systems administrator

Migrate data – option 1

Task	Description	Skills required
Migrate data with native MySQL or third-party tools.	This option uses native MySQL tools or third-party tools to migrate database objects and data. For instructi ons, see the documenta tion for mysqldump or Percona XtraBackup (for physical migration). For more information about using these tools, see the AWS blog post Migration options for MySQL to Amazon RDS for MySQL or Amazon Aurora MySQL.	DBA

Migrate data – option 2

Task	Description	Skills required
Migrate data with AWS DMS.	For more information, see High-level view of	DBA

Task	Description	Skills required
	[AWS DMS](#) in the AWS DMS documentation.	

Prepare for cutover

Task	Description	Skills required
Gather object counts.	Gather object counts from source database and new target databases. Fix any discrepancies in the target database.	DBA
Check dependencies.	Confirm that dependencies (links) to and from other databases are still valid and work correctly.	DBA
Test.	If this is a testing cycle, perform query testing, gather metrics, and fix any issues.	DBA

Cut over

Task	Description	Skills required
Move clients.	Switch the application clients over to the new infrastru cture.	DBA, App owner, Systems administrator
Provide support.	Provide support during functional application testing.	DBA

Close the project

Task	Description	Skills required
Shut down resources.	Shut down the AWS DMS replication instance and other temporary AWS resources.	DBA, Systems administrator
Review and project documents.	Review and validate the project documents.	DBA, App owner, Systems administrator
Collect metrics.	Collect metrics such as time to migrate, percentage of manual changes compared with tool-aided changes, and cost savings.	DBA, App owner, Systems administrator
Close out project.	Close out the migration project and provide feedback.	DBA, App owner, Systems administrator
Decommission source database.	Decommission the on-premis es MySQL database.	DBA, Systems administrator

Related resources

References

- Amazon EC2 documentation
- AWS DMS documentation
- Amazon EC2 Pricing
- AWS DMS Step-by-Step Walkthroughs
- mysqldump
- Percona XtraBackup

Tutorials and videos

- Getting Started with AWS DMS

- Introduction to Amazon EC2 – Elastic Cloud Server & Hosting with AWS (video)

Reduce homogeneous SAP migration cutover time by using Application Migration Service

Pavel Rubin, Diego Valverde, and Sunil Yadav, Amazon Web Services

Summary

This pattern outlines the steps for migrating SAP workloads by using AWS Application Migration Service. Application Migration Service facilitates cutovers by using block-level replication to maintain replication volumes that continually sync from their sources.

SAP workloads include the applications SAP Customer Relationship Management (SAP CRM), SAP Enterprise Resource Planning (ERP), and SAP Business Warehouse (SAP BW).

Prerequisites and limitations

Prerequisites

- An active AWS account with stable network connectivity between source SAP servers and the destination virtual private cloud (VPC) on AWS
- An SAP Adaptive Server Enterprise (ASE) source database for Linux or Windows in an on-premises data center

Limitations

- The target operating system must be supported by Amazon Elastic Compute Cloud (Amazon EC2). For more information, see Amazon EC2 FAQs.

Architecture

Source technology stack

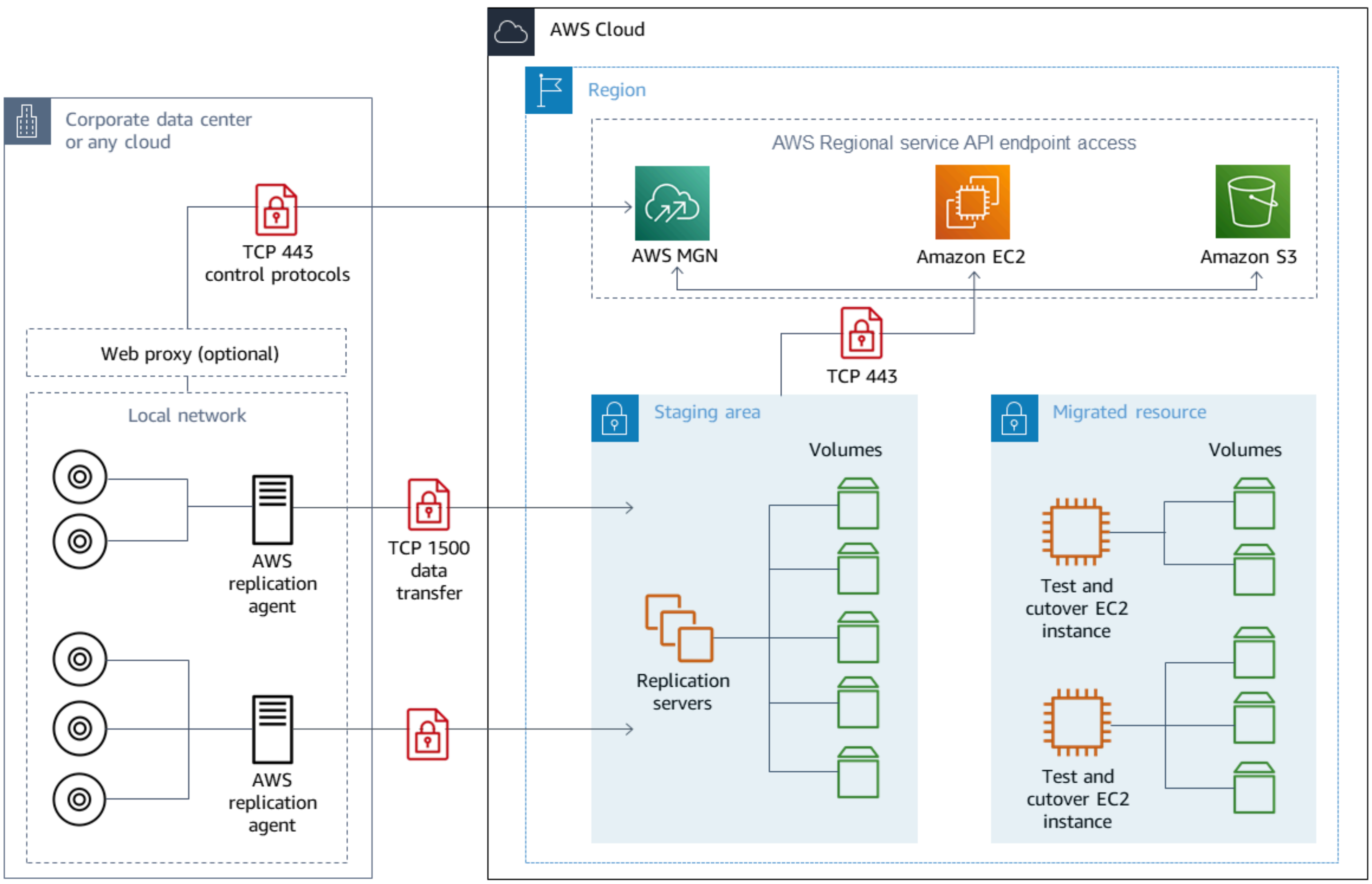
- An SAP ASE database

Target technology stack

- Amazon EC2
- Amazon Elastic Block Store (Amazon EBS)

Source and target architecture

The following diagram shows migration from the on-premises servers through the Replication Agent to the Application Migration Service endpoint. An Amazon Simple Storage Service (Amazon S3) endpoint is used to access installation and configuration files. The subnets for the staging area and the migrated resources contain EC2 instances, with data storage on EBS volumes. Port TCP 443 is used to connect the source machine network to Application Migration Service, and to connect the staging area subnets to the Application Migration Service, Amazon EC2, and Amazon S3 Regional endpoints. Port TCP 1500 is used for data replication between the local network and the staging area.



Tools

- AWS Application Migration Service helps you rehost (*lift-and-shift*) applications to the AWS Cloud without change and with minimal downtime.
- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances.

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Security Token Service (AWS STS) helps you request temporary, limited-privilege credentials for users.

Epics

Initialize Application Migration Service

Task	Description	Skills required
Initialize Application Migration Service.	Initialize Application Migration Service in the AWS Region where you want to deploy the SAP ASE database. AWS provides an automated setup the first time that you navigate to the Application Migration Service page in each Region.	AWS administrator
Manually create service roles.	(Optional) If you want to use automation (for example, AWS Control Tower) to set up the account, you can manually create the six AWS Identity and Access Management (IAM) roles that are required for installation, replication, and launches. For instructions, see the AWS documentation.	AWS administrator
Create a Replication Settings template.	The Replication Settings template defines the subnet,	General AWS

Task	Description	Skills required
	instance type, Amazon EBS encryption, and how data is routed. For detailed settings information, see the [AWS documentation]().	

Generate credentials for Agent installation

Task	Description	Skills required
Create a new IAM Role.	On the IAM console, navigate to **Roles**, and choose **Create role**. For the **Trusted entity type**, choose **AWS account**, and then choose **Next**.	AWS systems administrator
Attach AWSApplicationMigr ationAgentPolicy to the IAM Role.	The AWS managed `AWSApplicationMigr ationAgentPolicy` policy contains the necessary permissions to perform Application Migration Service Agent installation. After you attach the policy, choose **Next**.	AWS systems administrator
Complete the role creation.	Assign a friendly name, and choose **Create role**.	AWS systems administrator
Generate temporary credentia ls.	To generate the access key ID, secret access key, and session token, follow the instructions in the [AWS STS documenta]()	AWS systems administrator

Task	Description	Skills required
	tion. These credentials are used during Agent installat ion.	

Install the Application Migration Service Agent on the SAP source machine

Task	Description	Skills required
Download the Agent installer on the SAP source machine.	Download the Agent installer appropriate for your source operating system: Windows or Linux.	App owner
Install the AWS Replication Agent.	When you run the Agent installer file on a source machine, you are first asked to enter your access key, secret access key, session token, and the Region to replicate to. Use the temporary credentials from the IAM role that you previously created, and the same Region that you configured during initializ ation.	App owner
Wait for initial data replicati on.	After the Agent is installed, the source machine appears on the **Machines** tab on the Application Migration Service console.	App owner

Configure the target machine's Launch template

Task	Description	Skills required
Update the Launch template for the source server.	Each source server uses a unique EC2 Launch template that informs the configura tion of the target EC2 server. You can edit this template if you want to customize the Amazon EC2 configuration of your migrated server.	General AWS
Set the default Launch template version.	After you make the required changes to the Launch template, specify to use this updated version as the default Launch template. For more information, see the AWS documentation.	General AWS
Turn off Instance type right sizing.	(Optional) Instance type right sizing provides automatic instance type recommend ations based on the configura tion of the source SAP server. We recommend turning off this setting so that you can specify customized instance types in the Launch template.	General AWS

Perform a test

Task	Description	Skills required
Initiate a test launch.	On the Application Migration Service console, select one or	General AWS, Migration engineer, Migration lead

Task	Description	Skills required
	more servers, and then select **Launch test instances** under **Test and Cutover**.	
Wait for the conversion and launch process to be completed.	You can review the launch process on the **Launch history** tab. After the machine has successfully launched as an EC2 instance, the **Alerts** tab will be updated to **Launched**.	
Verify that the test completed successfully.	Connect to the launched instance through Remote Desktop Protocol (RDP) or SSH (Secure Shell), and perform the appropriate application checks. For example, log in to the SAP interface and validate functionality.	Migration engineer, App owner
Update the source lifecycle.	If testing was successful, update the source machine lifecycle to **Mark as "Ready for cutover"** on the **Test and Cutover** tab.	Migration engineer, Migration lead

Schedule and perform a cutover to the Amazon EC2 target

Task	Description	Skills required
Schedule a cutover window.		Cutover lead, Migration lead, App owner

Task	Description	Skills required
Initiate a cutover launch.	Select one or more servers. On the **Test and Cutover** tab, select **Launch cutover instances** under **Test and Cutover** on the Application Migration Service console.	Migration engineer
Wait for the conversion and launch processes to be completed.	You can review the launch process on the **Launch history** tab. After the machine has successfully launched as an EC2 instance, the **Alerts** tab will be updated to **Launched**.	
Verify that the cutover completed successfully.	Connect to the launched instance through RDP or SSH, and perform the appropriate application checks.	App owner, Migration engineer
Update the source lifecycle.	If cutover was successful, update the source machine lifecycle by selecting **Finalize cutover** on the **Test and Cutover** tab.	Migration engineer

Related resources

References

- AWS Application Migration Service
- AWS Application Migration FAQ

Video

- AWS Application Migration Service Architecture

Rehost on-premises workloads in the AWS Cloud: migration checklist

Srikanth Rangavajhala, Amazon Web Services

Summary

Rehosting on-premises workloads in the Amazon Web Services (AWS) Cloud involves the following migration phases: planning, pre-discovery, discovery, build, test, and cutover. This pattern outlines the phases and their related tasks. The tasks are described at a high level and support about 75% of all application workloads. You can implement these tasks over two to three weeks in an agile sprint cycle.

You should review and vet these tasks with your migration team and consultants. After the review, you can gather the input, eliminate or re-evaluate tasks as necessary to meet your requirements, and modify other tasks to support at least 75% of the application workloads in your portfolio. You can then use an agile project management tool such as Atlassian Jira or Rally Software to import the tasks, assign them to resources, and track your migration activities.

The pattern assumes that you're using AWS Cloud Migration Factory to rehost your workloads, but you can use your migration tool of choice.

Amazon Macie can help identify sensitive data in your knowledge bases, stored as data sources, model invocation logs, and prompt stores in Amazon Simple Storage Service (Amazon S3) buckets. For more information, see the Macie documentation.

Prerequisites and limitations

Prerequisites

- Project management tool for tracking migration tasks (for example, Atlassian Jira or Rally Software)
- Migration tool for rehosting your workloads on AWS (for example, Cloud Migration Factory)

Architecture

Source platform

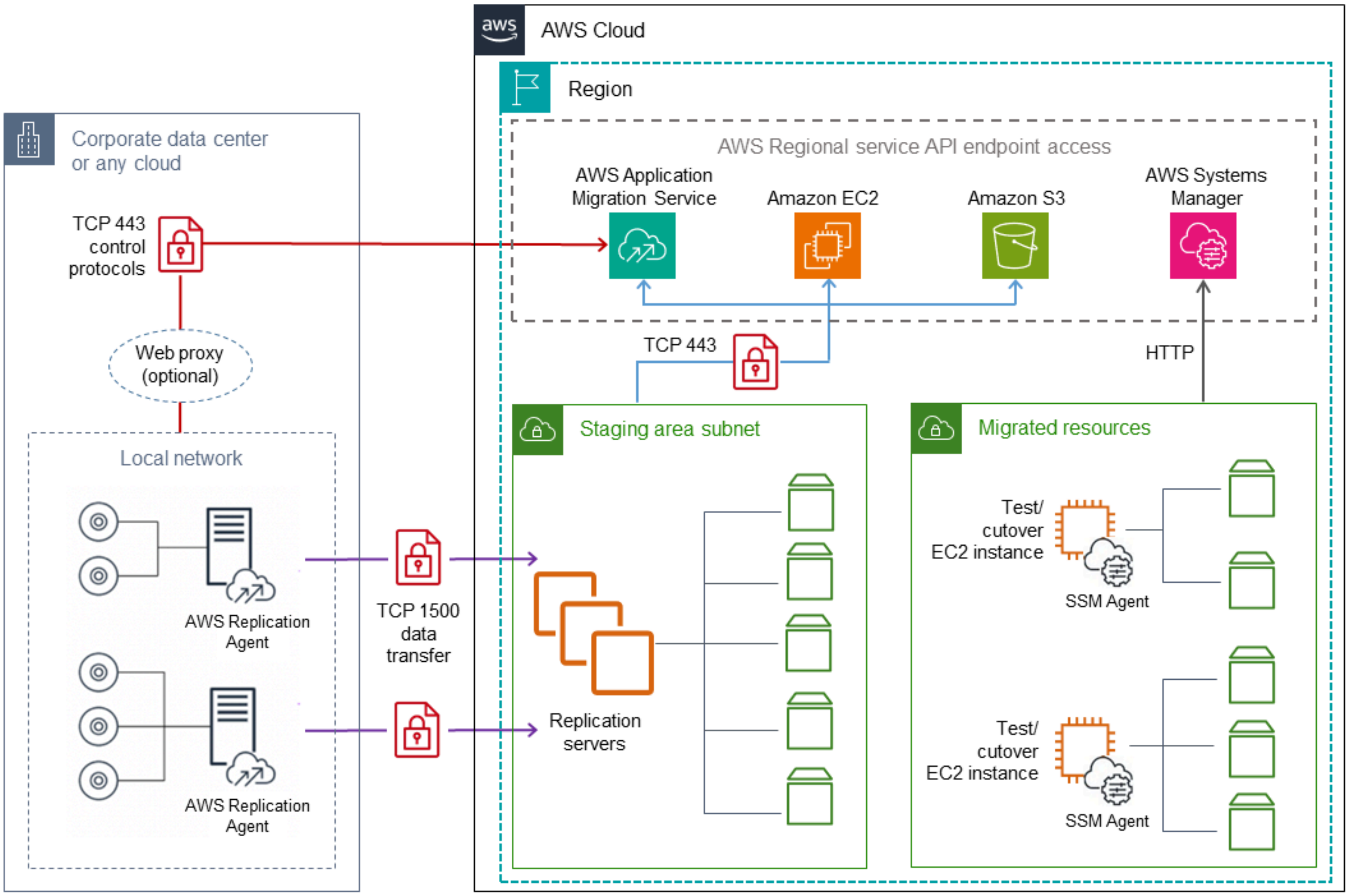
- On-premises source stack (including technologies, applications, databases, and infrastructure)

Target platform

- AWS Cloud target stack (including technologies, applications, databases, and infrastructure)

Architecture

The following diagram illustrates rehosting (discovering and migrating servers from an on-premises source environment to AWS) by using Cloud Migration Factory and AWS Application Migration Service.



Tools

- You can use a migration and project management tool of your choice.

Epics

Planning phase

Task	Description	Skills required
Groom the pre-discovery backlog.	Conduct the pre-discovery backlog grooming working session with department leads and application owners.	Project manager, Agile scrum leader
Conduct the sprint planning working session.	As a scoping exercise, distribute the applications that you want to migrate across sprints and waves.	Project manager, Agile scrum leader

Pre-discovery phase

Task	Description	Skills required
Confirm application knowledge.	Confirm and document the application owner and their knowledge of the application. Determine whether there's another point person for technical questions.	Migration specialist (intervie wer)
Determine application compliance requirements.	Confirm with the application owner that the application doesn't have to comply with requirements for Payment Card Industry Data Security Standard (PCI DSS), Sarbanes-Oxley Act (SOX), personall y identifiable information (PII), or other standards. If compliance requirements	Migration specialist (intervie wer)

Task	Description	Skills required
	exist, teams must finish their compliance checks on the servers that will be migrated.	
Confirm production release requirements.	Confirm the requirements for releasing the migrated application to production (such as release date and downtime duration) with the application owner or technical contact.	Migration specialist (intervie wer)
Get server list.	Get the list of servers that are associated with the targeted application.	Migration specialist (intervie wer)
Get the logical diagram that shows the current state.	Obtain the current state diagram for the application from the enterprise architect or the application owner.	Migration specialist (intervie wer)
Create a logical diagram that shows the target state.	Create a logical diagram of the application that shows the target architecture on AWS. This diagram should illustrate servers, connectiv ity, and mapping factors.	Enterprise architect, Business owner
Get server information.	Collect information about the servers that are associate d with the application, including their configuration details.	Migration specialist (intervie wer)

Task	Description	Skills required
Add server information to the discovery template.	Add detailed server informati on to the application discovery template (see `mobilize-applicati on-questionnaire.x lsx` in the attachment for this pattern). This template includes all the application-related security, infrastru cture, operating system, and networking details.	Migration specialist (intervie wer)
Publish the application discovery template.	Share the application discovery template with the application owner and migration team for common access and use.	Migration specialist (intervie wer)

Discovery phase

Task	Description	Skills required
Confirm server list.	Confirm the list of servers and the purpose of each server with the application owner or technical lead.	Migration specialist
Identify and add server groups.	Identify server groups such as web servers or applicati on servers, and add this information to the application discovery template. Select the tier of the application (web,	Migration specialist

Task	Description	Skills required
	application, database) that each server should belong to.	
Fill in the application discovery template.	Complete the details of the application discovery template with the help of the migration team, application team, and AWS.	Migration specialist
Add missing server details (middleware and OS teams).	Ask middleware and operating system (OS) teams to review the application discovery template and add any missing server details, including database informati on.	Migration specialist
Get inbound/outbound traffic rules (network team).	Ask the network team to get the inbound/outbound traffic rules for the source and destination servers. The network team should also add existing firewall rules, export these to a security group format, and add existing load balancers to the application discovery template.	Migration specialist
Identify required tagging.	Determine the tagging requirements for the applicati on.	Migration specialist

Task	Description	Skills required
Create firewall request details.	Capture and filter the firewall rules that are required to communicate with the application.	Migration specialist, Solutions architect, Network lead
Update the EC2 instance type.	Update the Amazon Elastic Compute Cloud (Amazon EC2) instance type to be used in the target environment, based on infrastructure and server requirements.	Migration specialist, Solutions architect, Network lead
Identify the current state diagram.	Identify or create the diagram that shows the current state of the application. This diagram will be used in the information security (InfoSec) request.	Migration specialist, Solutions architect
Finalize the future state diagram.	Finalize the diagram that shows the future (target) state for the application. This diagram will also be used in the InfoSec request.	Migration specialist, Solutions architect
Create firewall or security group service requests.	Create firewall or security group service requests (for development/QA, pre-produ ction, and production). If you're using Cloud Migration Factory, include replication-specific ports if they're not already open.	Migration specialist, Solutions architect, Network lead

Task	Description	Skills required
Review firewall or security group requests (InfoSec team).	In this step, the InfoSec team reviews and approves the firewall or security group requests that were created in the previous step.	InfoSec engineer, Migration specialist
Implement firewall security group requests (network team).	After the InfoSec team approves the firewall requests, the network team implements the required inbound/outbound firewall rules.	Migration specialist, Solutions architect, Network lead

Build phase (repeat for development/QA, pre-production, and production environments)

Task	Description	Skills required
Import the application and server data.	1. Verify that you are logged in to your migration execution server as a domain user with local administrator permissio ns on the in-scope source servers. 2. Use the migration intake form to import the attributes for the in-scope source servers. For additional information, see the [Cloud Migration Factory Implementation Guide]().	Migration specialist, Cloud administrator

Task	Description	Skills required
	If you aren't using Cloud Migration Factory, follow the instructions for setting up your migration tool.	
Check prerequisites for source servers.	Connect with the in-scope source servers to verify prerequisites such as TCP port 1500, TCP port 443, root volume free space, .NET Framework version, and other parameters. These are required for replication. For additional information, see the [Cloud Migration Factory Implementation Guide]().	Migration specialist, Cloud administrator
Create a service request to install replication agents.	Create a service request to install replication agents on the in-scope servers for development/QA, pre-produ ction, or production.	Migration specialist, Cloud administrator
Install the replication agents.	Install the replication agents on the in-scope source servers on the development/QA, pre-production, or productio n machines. For additional information, see the [Cloud Migration Factory Implement ation Guide]().	Migration specialist, Cloud administrator

Task	Description	Skills required
Push the post-launch scripts.	Application Migration Service supports post-launch scripts to help you automate OS-level activities such as installing or uninstalling software after you launch target instances. This step pushes the post-launch scripts to Windows or Linux machines, depending on the servers identified for migration. For instructions, see the [Cloud Migration Factory Implementation Guide].	Migration specialist, Cloud administrator
Verify the replication status.	Confirm the replication status for the in-scope source servers automatically by using the provided script. The script repeats every five minutes until the status of all source servers in the given wave changes to **Healthy**. For instructions, see the [Cloud Migration Factory Implementation Guide].	Migration specialist, Cloud administrator

Task	Description	Skills required
Create the admin user.	A local admin or sudo user on source machines might be needed to troubleshoot any issues after migration cutover from the in-scope source servers to AWS. The migration team uses this user to log in to the target server when the authentication server (for example, the DC or LDAP server) is not reachable. For instructions for this step, see the Cloud Migration Factory Implementation Guide.	Migration specialist, Cloud administrator
Validate the launch template.	Validate the server metadata to make sure it works successfully and has no invalid data. This step validates both test and cutover metadata. For instructions, see the Cloud Migration Factory Implementation Guide.	Migration specialist, Cloud administrator

Test phase (repeat for development/QA, pre-production, and production environments)

Task	Description	Skills required
Create a service request.	Create a service request for the infrastructure team and other teams to perform application cutover to development/QA, pre-produ	Migration specialist, Cloud administrator

Task	Description	Skills required
	ction, or production instances .	
Configure a load balancer (optional).	Configure required load balancers, such as an Application Load Balancer or an F5 load balancer with iRules.	Migration specialist, Cloud administrator
Launch instances for testing.	Launch all target machines for a given wave in Application Migration Service in test mode. For additional information, see the Cloud Migration Factory Implementation Guide.	Migration specialist, Cloud administrator
Verify the target instance status.	Verify the status of the target instance by checking the bootup process for all in-scope source servers in the same wave. It may take up to 30 minutes for the target instances to boot up. You can check the status manually by logging in to the Amazon EC2 console, searching for the source server name, and reviewing the **Status check** column. The status **2/2 checks passed** indicates that the instance is healthy from an infrastructure perspective.	Migration specialist, Cloud administrator

Task	Description	Skills required
Modify DNS entries.	Modify Domain Name System (DNS) entries. (Use `resolv.conf` or `host.conf` for a Microsoft Windows environment.) Configure each EC2 instance to point to the new IP address of this host. **Note** Make sure that there are no DNS conflicts between on-premises and AWS Cloud servers. This step and the following steps are optional, depending on the environment where the server is hosted.	Migration specialist, Cloud administrator
Test connectivity to backend hosts from EC2 instances.	Check the logins by using the domain credentials for the migrated servers.	Migration specialist, Cloud administrator
Update the DNS A record.	Update the DNS A record for each host to point to the new Amazon EC2 private IP address.	Migration specialist, Cloud administrator

Task	Description	Skills required
Update the DNS CNAME record.	Update the DNS CNAME record for virtual IPs (load balancer names) to point to the cluster for web and application servers.	Migration specialist, Cloud administrator
Test the application in applicable environments.	Log in to the new EC2 instance and test the applicati on in the development/ QA, pre-production, and production environments.	Migration specialist, Cloud administrator
Mark as ready for cutover.	When testing is complete, change the status of the source server to indicate that it's ready for cutover, so users can launch a cutover instance. For instructions, see the Cloud Migration Factory Implement ation Guide.	Migration specialist, Cloud administrator

Cutover phase

Task	Description	Skills required
Create production deploymen t plan.	Create a production deployment plan (including a backout plan).	Migration specialist, Cloud administrator
Notify operations team of downtime.	Notify the operations team of the downtime schedule for the servers. Some teams might require a change request or service request	Migration specialist, Cloud administrator

Task	Description	Skills required
	(CR/SR) ticket for this notification.	
Replicate production machines.	Replicate production machines by using Application Migration Service or another migration tool.	Migration specialist, Cloud administrator
Shut down in-scope source servers.	After you verify the source servers' replication status, you can shut down the source servers to stop transactions from client applications to the servers. You can shut down the source servers in the cutover window. For more information, see the [Cloud Migration Factory Implementation Guide]().	Cloud administrator
Launch instances for cutover.	Launch all target machines for a given wave in Application Migration Service in cutover mode. For more information, see the [Cloud Migration Factory Implementation Guide]().	Migration specialist, Cloud administrator

Task	Description	Skills required
Retrieve target instance IPs.	Retrieve the IPs for target instances. If the DNS update is a manual process in your environment, you would need to get the new IP addresses for all target instances. For more information, see the Cloud Migration Factory Implementation Guide.	Migration specialist, Cloud administrator
Verify target server connections.	After you update the DNS records, connect to the target instances with the host name to verify the connections. For more information, see the Cloud Migration Factory Implementation Guide.	Migration specialist, Cloud administrator

Related resources

- How to migrate
- AWS Cloud Migration Factory Implementation Guide
- Automating large-scale server migrations with Cloud Migration Factory
- AWS Application Migration Service User Guide
- AWS Migration Acceleration Program

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Set up Multi-AZ infrastructure for a SQL Server Always On FCI by using Amazon FSx

Manish Garg, T.V.R.L.Phani Kumar Dadi, Nishad Mankar, and RAJNEESH TYAGI, Amazon Web Services

Summary

If you need to migrate a large number of Microsoft SQL Server Always On Failover Cluster Instances (FCIs) quickly, this pattern can help you minimize provisioning time. By using automation and Amazon FSx for Windows File Server, it reduces manual efforts, human-made errors, and the time required to deploy a large number of clusters.

This pattern sets up the infrastructure for SQL Server FCIs in a Multi-Availability Zone (Multi-AZ) deployment on Amazon Web Services (AWS). The provisioning of the AWS services required for this infrastructure is automated by using [AWS CloudFormation](https://aws.amazon.com/cloudformation/) templates. SQL Server installation and cluster node creation on an [Amazon Elastic Compute Cloud (Amazon EC2)](https://aws.amazon.com/ec2/) instance is performed by using PowerShell commands.

This solution uses a highly available Multi-AZ [Amazon FSx for Windows](https://aws.amazon.com/fsx/windows/) file system as the shared witness for storing the SQL Server database files. The Amazon FSx file system and EC2 Windows instances that host SQL Server are joined to the same AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) domain.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS user with sufficient permissions to provision resources using AWS CloudFormation templates
- AWS Directory Service for Microsoft Active Directory
- Credentials in AWS Secrets Manager to authenticate to AWS Managed Microsoft AD in a key-value pair:
  - `ADDomainName`: <Domain Name>
  - `ADDomainJoinUserName`: <Domain Username>
  - `ADDomainJoinPassword`:<Domain User Password>
  - `TargetOU`: <Target OU Value>

> **Note**
>
> You will use the same key name in AWS Systems Manager automation for the AWS Managed Microsoft AD join activity.

- SQL Server media files for SQL Server installation and Windows service or domain accounts created, which will be used during cluster creation
- A virtual private cloud (VPC), with two public subnets in separate Availability Zones, two private subnets in the Availability Zones, an internet gateway, NAT gateways, route table associations, and a jump server

Product versions

- Windows Server 2012 R2 and Microsoft SQL Server 2016

Architecture

Source technology stack

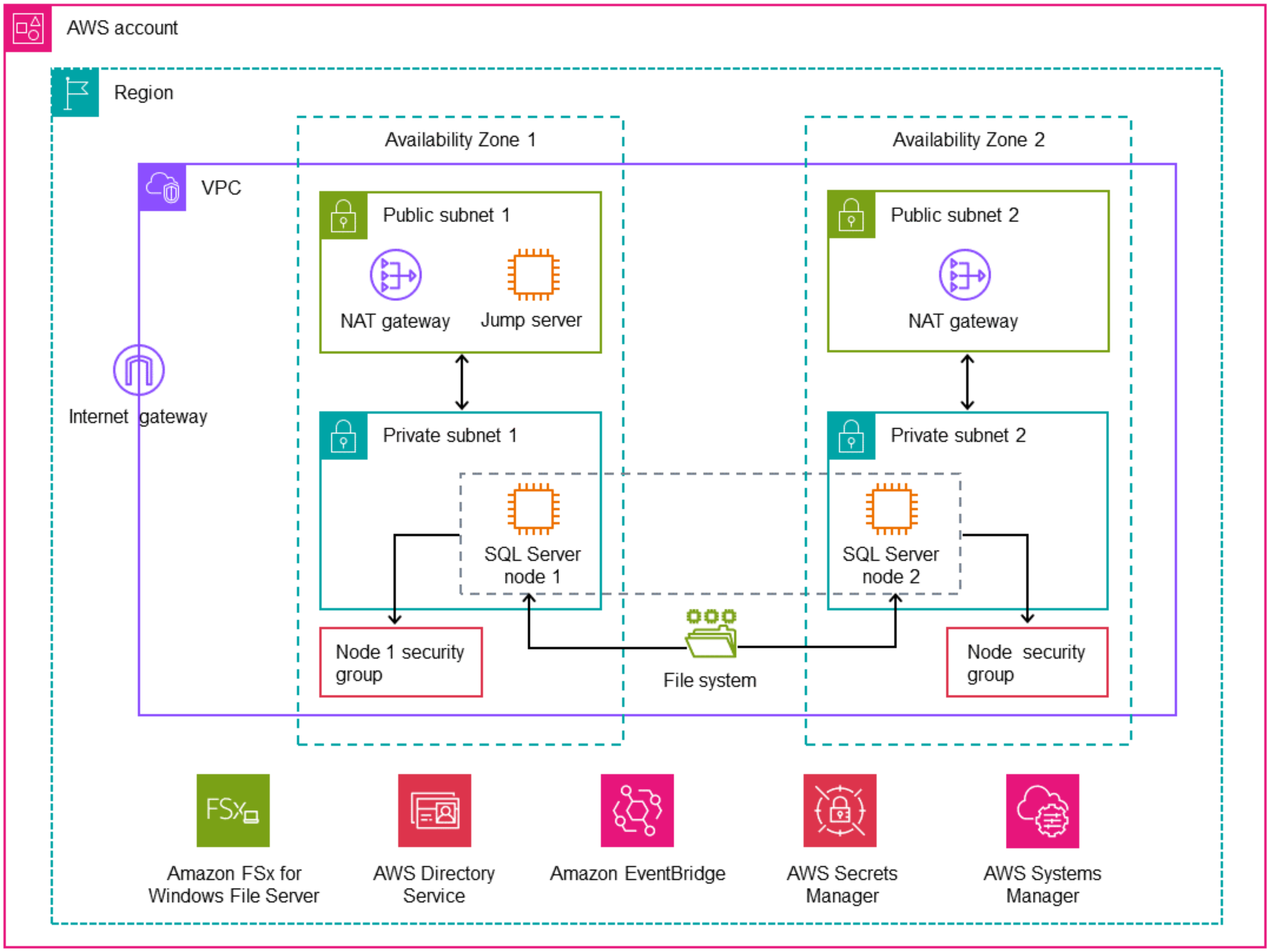
- On-premises SQL Server with FCIs using a shared drive

Target technology stack

- AWS EC2 instances
- Amazon FSx for Windows File Server
- AWS Systems Manager Automation runbook
- Network configurations (VPC, subnets, internet gateway, NAT gateways, jump server, security groups)
- AWS Secrets Manager
- AWS Managed Microsoft AD
- Amazon EventBridge
- AWS Identity and Access Management (IAM)

Target architecture

The following diagram shows an AWS account in a single AWS Region, with a VPC that includes two Availability Zones, two public subnets with NAT gateways, a jump server in the first public subnet, two private subnets, each with an EC2 instance for a SQL Server node in a node security group, and an Amazon FSx file system connecting to each of the SQL Server nodes. AWS Directory Service, Amazon EventBridge, AWS Secrets Manager, and AWS Systems Manager are also included.



Automation and scale

- You can use AWS Systems Manager to join AWS Managed Microsoft AD and perform the SQL Server installation.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Directory Service provides multiple ways to use Microsoft Active Directory (AD) with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Relational Database Service (Amazon RDS) for SQL Server, and Amazon FSx for Windows File Server.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.

Other tools

- PowerShell is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS. This pattern uses PowerShell scripts.

Code repository

The code for this pattern is available in the GitHub aws-windows-failover-cluster-automation repository.

Best practices

- The IAM roles that are used to deploy this solution should adhere to the principle of least privilege. For more information, see the IAM documentation.
- Follow the AWS CloudFormation best practices.

Epics

Deploy the infrastructure

Task	Description	Skills required
Deploy the Systems Manager CloudFormation stack.	1. Sign in to your AWS account, and open the AWS Management Console. 2. Navigate to the CloudFormation console, and create the Systems Manager CloudFormation stack by uploading the `ssm.yaml` template. Provide values for the following parameters: • **StateUnJoinAssociationLoggingBucketName** – Provide a name for the S3 bucket that the template will create for logging purposes. • **SSMAssociationADUnjoinName** – Provide a name for the `AWS::SSM::Association` resource. • **SSMAutomationDocumentName** – Provide a name for the Systems Manager Automation runbook. • **EventBridgeName** – Provide a name for the EventBridge event bus.	AWS DevOps, DevOps engineer

Task	Description	Skills required
	3. Deploy the Systems Manager CloudFormation stack by launching the `ssm.yaml` CloudFormation template. The template will create the Systems Manager Atomation runboook that is initiated when a new EC2 instance with the tag `ADJoined: FSXADD` launches. The Automation runbook will add the instance to AWS Managed Microsoft AD directory.	

Task	Description	Skills required
Deploy the infrastructure stack.	After successful deploymen t of the Systems Manager stack, create the `infra` stack, which includes EC2 instance nodes, security groups, the Amazon FSx for Windows File Server file system, and the IAM role. 1. Navigate to the CloudForm ation console and launch the `infra-cf.yaml` template. To deploy this stack, the following parameters are required: • `ActiveDirectoryId` – ID for AWS Managed Microsoft AD • `ADDnsIpAddresses1` – Primary DNS IP address of AWS Managed Microsoft AD • `ADDnsIpAddresses2` – Secondary DNS IP address of AWS Managed Microsoft AD • `FSxSecurityGroupName` – Name of the Amazon FSx security group • `FSxWindowsFileSystemName` – Name of the Amazon FSx drive	AWS DevOps, DevOps engineer

Task	Description	Skills required
	• `ImageID` – ID of the base Windows 2012 R2 image or Amazon Machine Image (AMI) used to create the SQL Server instance node • `KeyPairName` – Key-value pair to attach to the EC2 instance nodes for access • `Node1SecurityGroupName` – Name of the first node security group • `Node2SecurityGroupName` – Name of the second node security group • `OUSecretName` – Name of the secret that contains the AWS Managed Microsoft AD information • `PrivateSubnet1` – ID of the first private subnet • `PrivateSubnet2` – ID of the second private subnet • `SqlFSxFCIName` – Name of the tag applied to the primary and secondary nodes and to Amazon FSx.	

Task	Description	Skills required
	• `SqlFSxServerNetBIOSName1` – Name of the primary EC2 instance node (maximum of 15 characters) • `SqlFSxServerNetBIOSName2` – Name of the secondary EC2 instance node (maximum of 15 characters) • `VPC` – VPC ID • `WorkloadInstanceType` – Type of EC2 instance Deploy the `infra` stack. The stack will create all the infrastructure components that are required to set up Windows SQL Server FCI. 2. After the EC2 instance nodes are launched, the Systems Manager Automation document will be invoked to join these instances to AWS Managed Microsoft AD. You can track the progress on the Systems Manager console **Automation** page.	

Set up the Windows SQL Server Always On FCI

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install Windows tools.</td><td>1. Log in to the primary EC2 instance, which is node 1. To install the Windows features (Active Directory and FCI Tools), run the following PowerShell script.
<pre>Install-WindowsFeature -Name RSAT-AD-Powershell,Failover-Clustering -IncludeManagementTools
Install-WindowsFeature -Name RSAT-Clustering,RSAT-ADDS-Tools,RSAT-AD-Powershell,RSAT-DHCP,RSAT-DNS-Server</pre>2. Log in to the secondary EC2 instance, which is node 2, and run same script to enable features on node 2.</td><td>AWS DevOps, DevOps engineer, DBA</td></tr>
<tr><td>Prestage the cluster computer objects in Active Directory Domain Services.</td><td>To prestage the cluster name object (CNO) in Active Directory Domain Services (AD DS) and prestage a virtual computer object (VCO) for a clustered role, follow the instructions in the Windows Server documentation.</td><td>AWS DevOps, DBA, DevOps engineer</td></tr>
</table>

Task	Description	Skills required
Create the WSFC.	To create the Windows Server Failover Clustering (WSFC) cluster, do the following: 1. Log in to the primary EC2 instance, which is node 1. To create the Amazon FSx file share and grant full access to the listed AD service account, run the following code. `Invoke-Command -ComputerName "<FSx Windows Remote PowerShell Endpoint>" -ConfigurationName FSxRemoteAdmin -scriptblock { New-FSxSmbShare -Name "SQLDB" -Path "D:\share" -Description "SQL Databases Share" -ContinuouslyAvailable $true -FolderEnumerationMode AccessBased -EncryptData $true grant-fsxsmbshareaccess -name SQLDB -AccountName "<domain\user>" -accessRight Full }` This command will also create the continuously available (CA) file share,	AWS DevOps, DBA, DevOps engineer

Task	Description	Skills required
	which is optimized for use by Microsoft SQL Server. 2. To create the failover cluster on the primary instance (node 1), run the following command. `New-Cluster -Name <CNO Name> -Node <Node1 Name>, <Node2 Name> -StaticAddress <Node1 Secondary Private IP>, <Node2 Secondary Private IP>` The command requires the following parameters: • `Name` – The name of the cluster (CNO) • `Node` – The names of the primary and secondary nodes, respectively • `StaticAddress` – The secondary IP addresses of the primary and secondary nodes, respectively **Important** A domain administrator or regular user must have administrator permission on	

Task	Description	Skills required
	both the nodes to create the Windows Server Failover Clustering (WSFC) cluster. Otherwise , the previous command will fail and return the message, You `do not have administrator privilege on servers.` 3. After the cluster is created, run the following command to attach the file share witness. `Set-ClusterQuorum -FileShareWitness \\<FSx Windows Remote PowerShell Endpoint>\share\witness`	

Task	Description	Skills required
Install the SQL Server failover cluster.	After the WSFC cluster is set up, install the SQL Server cluster on the primary instance (node1). 1. In the T drive on both nodes, create `tempdb` and `log` folders. The folders are used in the PowerShell commands. 2. After you copy the SQL Server media files for SQL Server installation on both nodes, run the following PowerShell command on node 1 to install SQL Server on node 1. `D:\setup.exe /Q `` ` `/ACTION=InstallFailoverCluster `` ` `/IACCEPTSQLSERVERLICENSETERMS `` ` `/FEATURES="SQL,IS,BC,Conn" `` ` `/INSTALLSHAREDDIR="C:\Program Files\Microsoft SQL Server" `` ` `/INSTALLSHAREDWOWDIR="C:\Program Files (x86)\Microsoft SQL Server" `` ` `/RSINSTALLMODE="FilesOnlyMode" `` ` `/INSTANCEID="MSSQLSERVER" `` `	AWS DevOps, DBA, DevOps engineer

Task	Description	Skills required
	`/INSTANCENAME="MSSQLSERVER" `` `/FAILOVERCLUSTERGROUP="SQL Server (MSSQLSERVER)" `` `/FAILOVERCLUSTERIPADDRESSES="IPv4;<2nd Sec Private Ip node1>;Cluster Network 1;<subnet mask>" `` `/FAILOVERCLUSTERNETWORKNAME="<Fail over cluster Network Name>" `` `/INSTANCEDIR="C:\Program Files\Microsoft SQL Server" `` `/ENU="True" `` `/ERRORREPORTING=0 `` `/SQMREPORTING=0 `` `/SAPWD="<Domain User password>" `` `/SQLCOLLATION="SQL_Latin1_General_CP1_CI_AS" `` `/SQLSYSADMINACCOUNTS="<domain\username>" `` `/SQLSVCACCOUNT="<domain\username>" /SQLSVCPASSWORD="<Domain User password>" `` `/AGTSVCACCOUNT="<domain\username>" /AGTSVCPASSWORD="<Domain User password>" `` `/ISSVCACCOUNT="<domain\username>" /ISSVCPASSWORD="<Domain User password>" ``	

Task	Description	Skills required

	`/FTSVCACCOUNT="NT Service\MSSQLFDLauncher"` `` ` ``
	`/INSTALLSQLDATADIR="\\<FSX DNS name>\share\Program Files\Microsoft SQL Server"` `` ` ``
	`/SQLUSERDBDIR="\\<FSX DNS name>\share\data"` `` ` ``
	`/SQLUSERDBLOGDIR="\\<FSX DNS name>\share\log"` `` ` ``
	`/SQLTEMPDBDIR="T:\tempdb"` `` ` ``
	`/SQLTEMPDBLOGDIR="T:\log"` `` ` ``
	`/SQLBACKUPDIR="\\<FSX DNS name>\share\SQLBackup"` `` ` ``
	`/SkipRules=Cluster_VerifyForErrors` `` ` ``
	`/INDICATEPROGRESS`	

Task	Description	Skills required
Add a secondary node to the cluster.	To add SQL Server to the secondary node (node 2), run the following PowerShell command. `D:\setup.exe /Q ` /ACTION=AddNode ` /IACCEPTSQLSERVERLICENSETERMS ` /INSTANCENAME="MSSQLSERVER" ` /FAILOVERCLUSTERGROUP="SQL Server (MSSQLSERVER)" ` /FAILOVERCLUSTERIPADDRESSES="IPv4;<2nd Sec Private Ip node2>;Cluster Network 2;<subnet mask>" ` /FAILOVERCLUSTERNETWORKNAME="<Fail over cluster Network Name>" ` /CONFIRMIPDEPENDENCYCHANGE=1 ` /SQLSVCACCOUNT="<domain\username>" /SQLSVCPASSWORD="<Domain User password>" ` /AGTSVCACCOUNT="domain\username>" /AGTSVCPASSWORD="<Domain User password>" ` /FTSVCACCOUNT="NT Service\MSSQLFDLauncher" ` /SkipRules=Cluster_VerifyForErrors ``	AWS DevOps, DBA, DevOps engineer

Task	Description	Skills required
	`/INDICATEPROGRESS`	
Test the SQL Server FCI.	1. On the Windows instance for one of the nodes, in **Administrative Tools**, launch the **Failover Cluster Manager**. 2. Navigate to **Nodes**, and confirm that the node status is **Status Running**. 3. Select **Roles**, open the context (right-click) menu for SQL Server (MSSQLSER VER), and select **Move and Select Node**. 4. After the node selection , SQL Server should be running on the other node.	DBA, DevOps engineer

Clean up resources

Task	Description	Skills required
Clean up resources.	To clean up the resources, use the AWS CloudFormation stack deletion process: 1. Open the [AWS CloudForm ation console](https://console.aws.amazon.com/cloudformation/). 2. On the **Stacks** page, select the `infra` stack. The stack must be currently running. 3. In the stack details pane, choose **Delete**.	AWS DevOps, DBA, DevOps engineer

Task	Description	Skills required
	4. Select **Delete stack** when prompted. 5. Repeat steps 2-4 for the `ssm` stack. After the stack deletion is complete, the stacks will be in the `DELETE_COMPLETE` state. Stacks in the `DELETE_COMPLETE` state aren't displayed in the CloudFormation console by default. To display deleted stacks, you must change the stack view filter as described in [Viewing deleted stacks on the AWS CloudFormation console](#). If the deletion failed, a stack will be in the `DELETE_FAILED` state. For solutions, see [Delete stack fails](#) in the CloudFormation documentation.	

Troubleshooting

Issue	Solution
AWS CloudFormation template failure	If the CloudFormation template fails during deployment, do the following: 1. Open the [AWS CloudFormation console](#).

Issue	Solution
	2. On the **Stacks** page in the CloudFormation console, select the stack. 3. Choose **Events**, and check the stack status.
AWS Managed Microsoft AD join failure	To troubleshoot the join issues, follow these steps: 1. Open the Systems Manager console. 2. Select the deployment Region. 3. In the left pane, choose **Automation**, and locate the failed Automation runbook. 4. Open the Automation runbook, and check for the **Execution status and Execution steps**. 5. Investigate the details of the failed step to see the exact error or failure.

Related resources

- Simplify your Microsoft SQL Server high availability deployments using Amazon FSx for Windows File Server
- Using FSx for Windows File Server with Microsoft SQL Server

Use BMC Discovery queries to extract migration data for migration planning

Ben Tailor-Hamblin, Emma Baldry, Simon Cunningham, and Shabnam Khan, Amazon Web Services

Summary

This guide provides query examples and steps to help you extract data from your on-premises infrastructure and applications by using BMC Discovery. The pattern shows you how to use BMC Discovery queries to scan your infrastructure and extract software, service, and dependency information. The extracted data is required for the assess and mobilize phases of a large-scale migration to the Amazon Web Services (AWS) Cloud. You can use this data to make critical decisions about which applications to migrate together as part of your migration plan.

Prerequisites and limitations

Prerequisites

- A license for BMC Discovery (formerly BMC ADDM) or the software as a service (SaaS) version of BMC Helix Discovery
- On-premises or SaaS version of BMC Discovery, installed

> **Note**
>
> For on-premises versions of BMC Discovery, you must install the application on a client network with access to all networking and server devices that are in scope for a migration across multiple data centers. Access to the client network must be provided according to application installation instructions. If the scanning of Windows Server information is required, then you must set up a Windows proxy manager device in the network.

- Networking access to allow the application to scan devices across data centers, if you're using BMC Helix Discovery

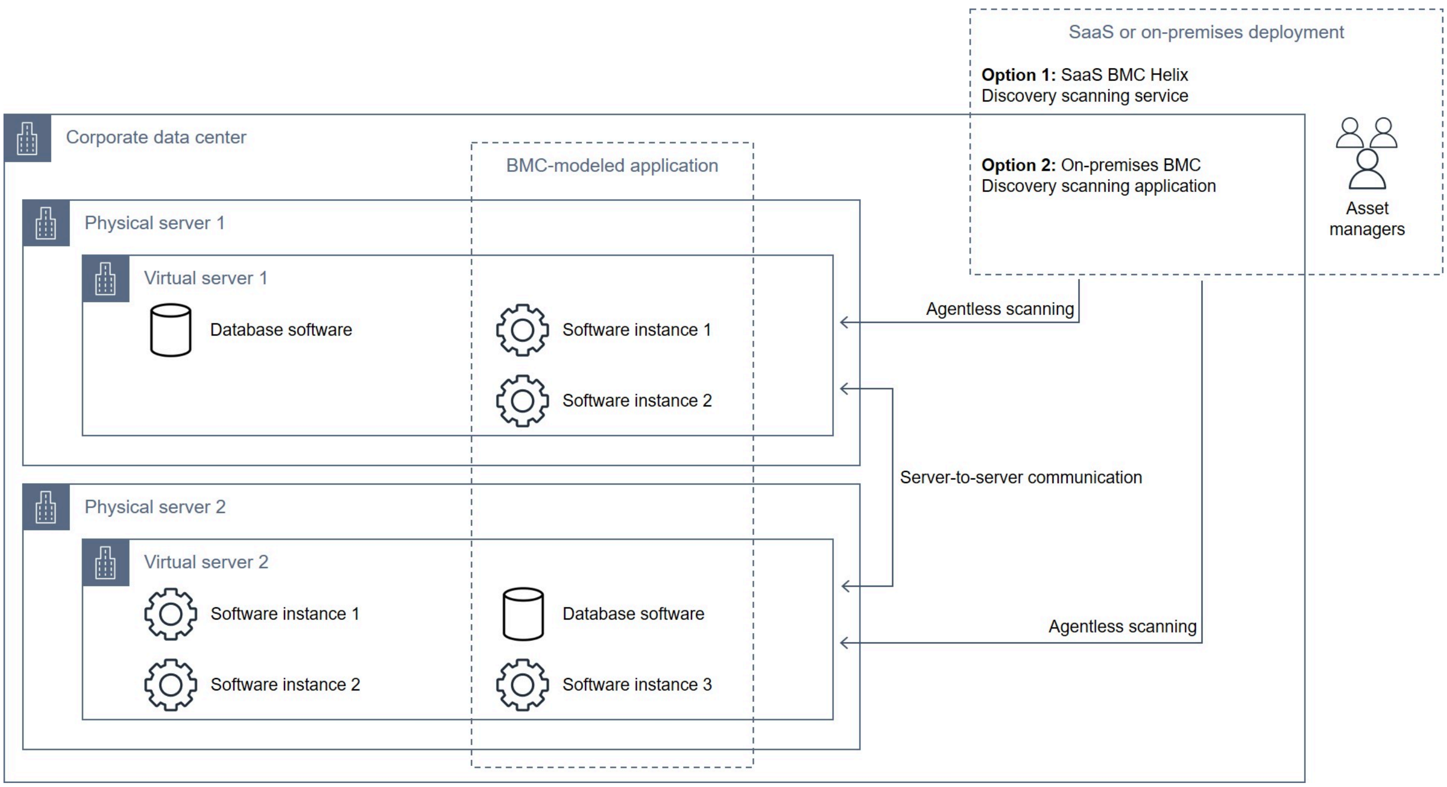
Product versions

- BMC Discovery 22.2 (12.5)
- BMC Discovery 22.1 (12.4)
- BMC Discovery 21.3 (12.3)
- BMC Discovery 21.05 (12.2)

- BMC Discovery 20.08 (12.1)
- BMC Discovery 20.02 (12.0)
- BMC Discovery 11.3
- BMC Discovery 11.2
- BMC Discovery 11.1
- BMC Discovery 11.0
- BMC Atrium Discovery 10.2
- BMC Atrium Discovery 10.1
- BMC Atrium Discovery 10.0

Architecture

The following diagram shows how asset managers can use BMC Discovery queries to scan BMC-modeled applications in both SaaS and on-premises environments.



The diagram shows the following workflow: An asset manager uses BMC Discovery or BMC Helix Discovery to scan database and software instances running on virtual servers hosted on multiple physical servers. The tool can model applications with components spanning multiple virtual and physical servers.

Technology stack

- BMC Discovery
- BMC Helix Discovery

Tools

- BMC Discovery is a data center discovery tool that helps you automatically discover your data center.
- BMC Helix Discovery is a SaaS-based discovery and dependency modeling system that helps you dynamically model your data assets and their dependencies.

Best practices

It's a best practice to map application, dependency, and infrastructure data when you migrate to the cloud. Mapping helps you understand the complexity of your current environment and the dependencies among various components.

The asset information these queries provide is important for several reasons:

1. **Planning –** Understanding the dependencies between components helps you plan the migration process more effectively. For example, you might need to migrate certain components first in order to ensure that others can be migrated successfully.
2. **Risk assessment –** Mapping the dependencies between components can help you identify any potential risks or issues that can arise during the migration process. For example, you might discover that certain components rely on outdated or unsupported technologies that could cause issues in the cloud.
3. **Cloud architecture –** Mapping your application and infrastructure data can also help you to design a suitable cloud architecture that meets your organizational needs. For example, you might need to design a multi-tier architecture to support high availability or scalability requirements.

Overall, mapping application, dependency, and infrastructure data is a crucial step in the cloud migration process. The mapping exercise can help you better understand your current environment, identify any potential issues or risks, and design a suitable cloud architecture.

Epics

Identify and evaluate discovery tooling

Task	Description	Skills required
Identify ITSM owners.	Identify the IT Service Management (ITSM) owners (usually by reaching out to the operational support teams).	Migration lead
Check CMDB.	Identify the number of configuration managemen t databases (CMDBs) that contain asset information, and then identify the sources of that information.	Migration lead
Identify discovery tools and check for use of BMC Discovery.	If your organization is using BMC Discovery to send data about your environment to the CMDB tool, check the scope and coverage of its scans. For example, check if BMC Discovery is scanning all data centers and if the access servers are located in perimeter zones.	Migration lead
Check the level of application modelling.	Check if applications are modelled in BMC Discovery . If not, recommend the use of the BMC Discovery tool to model which running software instances provide an application and business service.	Migration engineer, Migration lead

Extract infrastructure data

Task	Description	Skills required
Extract data on physical and virtual servers.	To extract data on the physical and virtual servers scanned by BMC Discovery, use [Query Builder](#) to run the following query: `search Host show key as 'Serverid', virtual, name as 'HOSTNAME', os_type as 'osName', os_version as 'OS Version', num_logical_processors as 'Logical Processor Counts', cores_per_processor as 'Cores per Processor', logical_ram as 'Logical RAM', #Consumer:StorageUse:Provider:DiskDrive.size as 'Size'` **Note** You can use extracted data to determine the appropriate instance sizes for migration.	Migration engineer, Migration lead
Extract data on modeled applications.	If your applications are modeled in BMC Discovery , you can extract data about the servers that run the application software. To get	BMC Discovery application owner

Task	Description	Skills required
	the server names, use Query Builder to run the following query: `search SoftwareInstance show key as 'ApplicationID', #RunningSoftware:HostedSoftware:Host:Host.key as 'ReferenceID', type, name` **Note** Applications are modeled in BMC Discovery by a collection of running software instances. The application is dependent on all the servers that run the application software.	

Task	Description	Skills required
Extract data on databases.	To get a list of all scanned databases and the servers these databases are running on, use [Query Builder](#) to run the following query: `search Database show key as 'Key', name, type as 'Source Engine Type', #Detail:Detail:ElementWithDetail:SoftwareInstance.name as 'Software Instance', #Detail:Detail:ElementWithDetail:SoftwareInstance.product_version as 'Product Version', #Detail:Detail:ElementWithDetail:SoftwareInstance.edition as 'Edition', #Detail:Detail:ElementWithDetail:SoftwareInstance.#RunningSoftware:HostedSoftware:Host:Host.key as 'ServerID'`	App owner

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Extract data on server communication.</td><td>To get information on all the network communications between servers that's collected by BMC Discovery from historic network communications logs, use Query Builder to run the following query:<pre><code>search Host
 TRAVERSE InferredElement:Inference:Associate:DiscoveryAccess
 TRAVERSE DiscoveryAccess:DiscoveryAccessResult:DiscoveryResult:NetworkConnectionList
 TRAVERSE List:List:Member:DiscoveredNetworkConnection
 PROCESS WITH networkConnectionInfo</code></pre></td><td>BMC Discovery application owner</td></tr>
<tr><td>Extract data on application discovery.</td><td>To get information on application dependencies, use Query Builder to run the following query:<pre><code>search SoftwareInstance
 show key as 'SRC App ID', #Dependant:Dependency:DependedUpon:SoftwareInstance.key as 'DEST App ID'</code></pre></td><td>BMC Discovery application owner</td></tr>
</table>

Task	Description	Skills required
Extract data on business services.	To extract data on business services provided by hosts, use [Query Builder](#) to run the following query: `search Host show name, #Host:HostedSoftware:AggregateSoftware:BusinessService.name as 'Name'`	BMC Discovery application owner

Troubleshooting

Issue	Solution
A query fails to run or contains unpopulated columns.	Review the asset records in BMC Discovery and determine which fields you require. Then, replace these fields in the query by using the [Query Builder](#).
The details of a dependent asset aren't populated.	This is likely due to access permissions or network connectivity. The discovery tool might not have the necessary permissions to access certain assets, particularly if they are on different networks or in different environments. We recommend that you work closely with discovery subject matter experts to ensure that all relevant assets are identified.

Related resources

References

- BMC Discovery Licensing entitlement (BMC documentation)
- BMC Discovery features and components (BMC documentation)
- BMC Discovery User Guide (BMC documentation)
- Searching for data (on BMC Discovery) (BMC documentation)
- Portfolio discovery and analysis for migration (AWS Prescriptive Guidance)

Tutorials and videos

- BMC Discovery: Webinar - Reporting Query Best Practices (Part 1) (YouTube)

Relocate

Topics

- Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication
- Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX
- Migrate an Amazon RDS DB instance to another VPC or account
- Migrate an Amazon RDS for Oracle DB instance to another VPC
- Migrate an Amazon Redshift cluster to an AWS Region in China
- Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport

Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication

Durga Prasad Cheepuri and Eduardo Valentim, Amazon Web Services

Summary

Warning: IAM users have long-term credentials, which presents a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed.

This pattern walks you through the steps for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle source database to a different AWS account and AWS Region. The pattern uses a DB snapshot for a one-time full data load, and enables AWS Database Migration Service (AWS DMS) for ongoing replication.

Prerequisites and limitations

Prerequisites

- An active AWS account that contains the source Amazon RDS for Oracle database, which has been encrypted using a non-default AWS Key Management Service (AWS KMS) key
- An active AWS account in a different AWS Region from the source database, to use for the target Amazon RDS for Oracle database
- Virtual private cloud (VPC) peering between the source and target VPCs
- Familiarity with using an Oracle database as a source for AWS DMS
- Familiarity with using an Oracle database as a target for AWS DMS

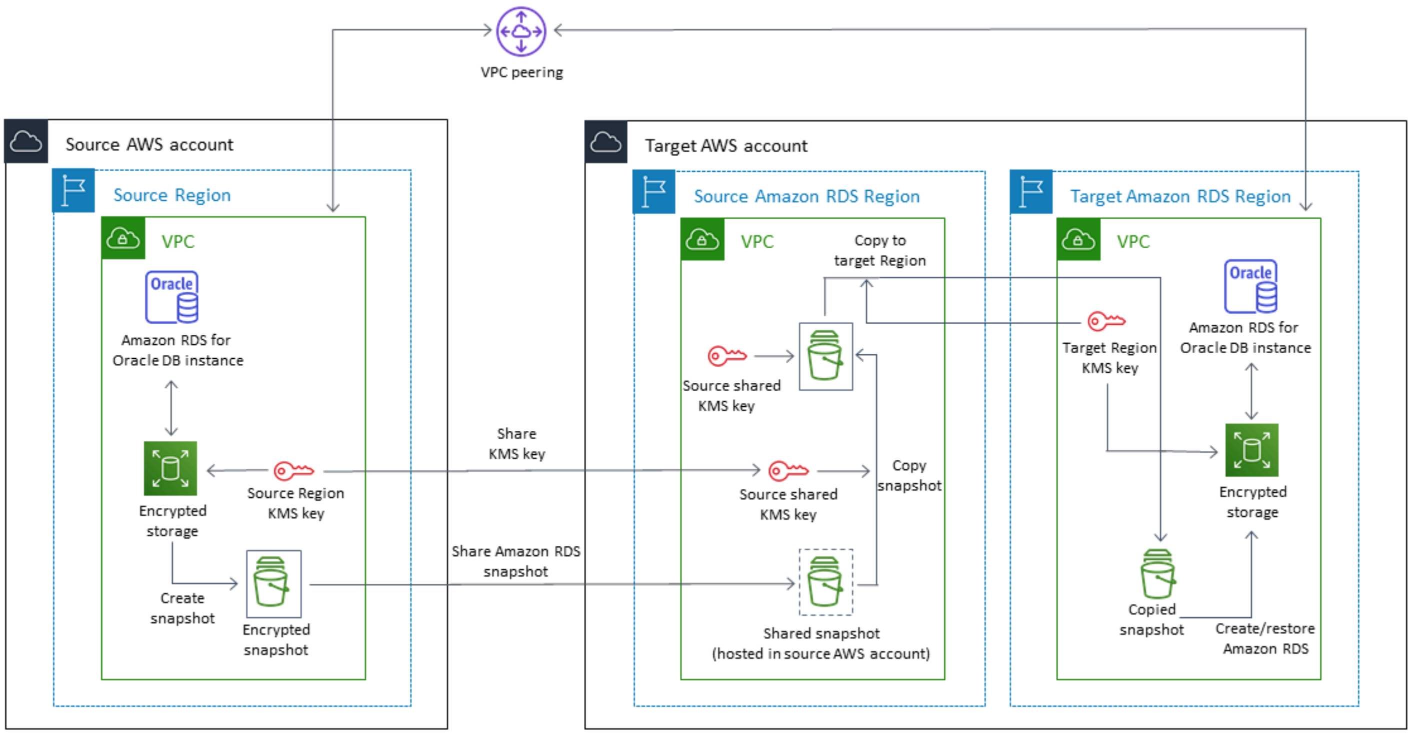
Product versions

- Oracle versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2, and 18c. For the latest list of supported versions and editions, see Using an Oracle Database as a Source for AWS DMS and with Using an Oracle database as a target for AWS DMS in the AWS documentation. For Oracle versions supported by Amazon RDS, see Oracle on Amazon RDS.

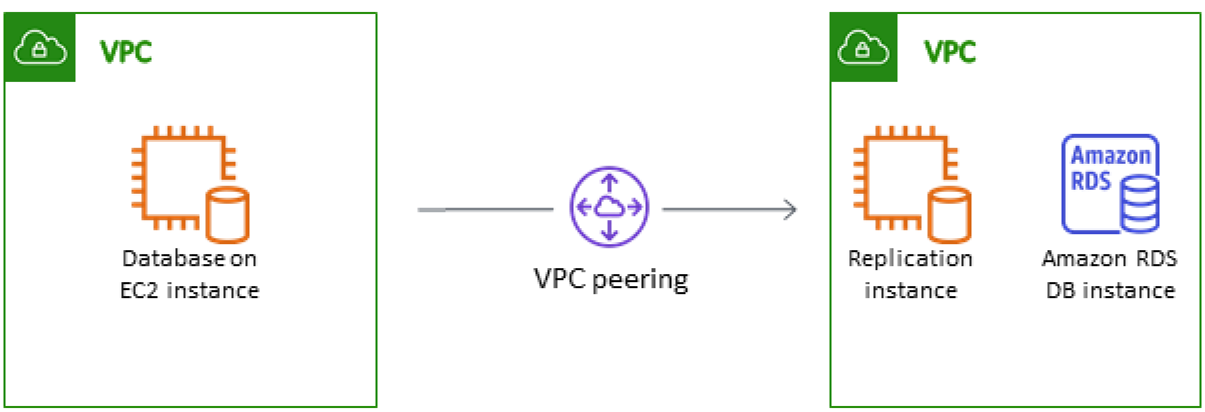
Architecture

Source and target technology stacks

- Amazon RDS for Oracle DB instance



Ongoing replication architecture



Tools

Tools used for one-time full data load

- Amazon Relational Database Service (Amazon RDS) creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases. When you create a DB snapshot, you need to identify which DB instance you are going to back up, and then give your DB snapshot a name so you can restore from it later. The amount of time it takes to create a snapshot varies with the size of your databases. Because the snapshot includes the entire storage volume, the size of files, such as temporary files, also affects the amount of time it takes to create the snapshot. For more information about using DB snapshots, see Creating a DB Snapshot in the Amazon RDS documentation.
- AWS Key Management Service (AWS KMS) creates a key for Amazon RDS encryption. When you create an encrypted DB instance, you can also supply the AWS KMS key identifier for your encryption key. If you don't specify an AWS KMS key identifier, Amazon RDS uses your default encryption key for your new DB instance. AWS KMS creates your default encryption key for your AWS account. Your AWS account has a different default encryption key for each AWS Region. For this pattern, the Amazon RDS DB instance should be encrypted using the non-default AWS KMS key. For more information about using AWS KMS keys for Amazon RDS encryption, see Encrypting Amazon RDS resources in the Amazon RDS documentation.

Tools used for ongoing replication

- AWS Database Migration Service (AWS DMS) is used to replicate ongoing changes and to keep the source and target databases in sync. For more information about using AWS DMS for ongoing replication, see Working with an AWS DMS replication instance in the AWS DMS documentation.

Epics

Configure your source AWS account

Task	Description	Skills required
Prepare the source Oracle DB instance.	Let the Amazon RDS for Oracle DB instance run in ARCHIVELOG mode, and set the retention period. For details, see Working with an AWS managed Oracle	DBA

Task	Description	Skills required
	database as a source for AWS DMS.	
Set supplemental logging for the source Oracle DB instance.	Set database-level and table-level supplemental logging for the Amazon RDS for Oracle DB instance. For details, see Working with an AWS managed Oracle database as a source for AWS DMS.	DBA
Update the AWS KMS key policy in the source account.	Update the AWS KMS key policy in the source AWS account to allow the target AWS account to use the encrypted Amazon RDS AWS KMS key. For details, see the AWS KMS documentation.	SysAdmin
Create a manual Amazon RDS DB snapshot of the source DB instance.		AWS IAM user
Share the manual, encrypted Amazon RDS snapshot with the target AWS account.	For details, see Sharing a DB snapshot.	AWS IAM user

Configure your target AWS account

Task	Description	Skills required
Attach a policy.	In the target AWS account, attach an AWS Identity and Access Management (IAM)	SysAdmin

Task	Description	Skills required
	policy to the root IAM user, to allow the IAM user to copy an encrypted DB snapshot using the shared AWS KMS key.	
Switch to the source AWS Region.		AWS IAM user
Copy the shared snapshot.	In the Amazon RDS console, in the **Snapshots** pane, choose **Shared with Me**, and select the shared snapshot. Copy the snapshot to the same AWS Region as the source database by using the Amazon Resource Name (ARN) for the AWS KMS key used by the source database. For details, see [Copying a DB snapshot]().	AWS IAM user
Switch to the target AWS Region, and create a new AWS KMS key.		AWS IAM user
Copy the snapshot.	Switch to the source AWS Region. On the Amazon RDS console, in the **Snapshots** pane, choose **Owned by Me**, and select the copied snapshot. Copy the snapshot to the target AWS Region by using the AWS KMS key for the new target AWS Region.	AWS IAM user

Task	Description	Skills required
Restore the snapshot.	Switch to the target AWS Region. On the Amazon RDS console, in the **Snapshots** pane, choose **Owned by Me**. Select the copied snapshot and restore it to an Amazon RDS for Oracle DB instance. For details, see [Restoring from a DB snapshot]().	AWS IAM user

Prepare your source database for ongoing replication

Task	Description	Skills required
Create an Oracle user with the appropriate permissions.	Create an Oracle user with the required privileges for Oracle as a source for AWS DMS. For details, see the [AWS DMS documentation]().	DBA
Configure the source database for Oracle LogMiner or Oracle Binary Reader.		DBA

Prepare your target database for ongoing replication

Task	Description	Skills required
Create an Oracle user with the appropriate permissions.	Create an Oracle user with the required privileges for Oracle as a target for AWS DMS. For details, see the [AWS DMS documentation]().	DBA

Create AWS DMS components

Task	Description	Skills required
Create a replication instance in the target AWS Region.	Create a replication instance in the VPC of the target AWS Region. For details, see the AWS DMS documentation.	AWS IAM user
Create source and target endpoints with required encryption, and test connections.	For details, see the AWS DMS documentation.	DBA
Create replication tasks.	1. For the migration type, choose ongoing replication. 2. For the change data capture (CDC) start point, use the Oracle system change number (SCN) when the Amazon RDS snapshot was taken for full load, or the timestamp when the full load was taken. 3. For `TargetTablePrepMode` , choose **DO_NOTHING**. If the task has large binary object (LOB) data tables, choose **Limited LOB mode**, and set the **max LOB size** to the maximum size of the LOB data in the table. 4. Enable logging.	IAM user

Task	Description	Skills required
	5. Group tables that are related through keys into a single task. If there are tables with a large amount of LOB data and the table has no relationship with other tables, create a separate task for it with the LOB settings described previously. For details, see the [AWS DMS documentation]().	
Start the tasks and monitor them.	For details, see the [AWS DMS documentation]().	AWS IAM user
Enable validation on the task if needed.	Note that enabling validatio n does have a performance impact on the replication. For details, see the [AWS DMS documentation]().	AWS IAM user

Related resources

- Changing a key policy
- Creating a manual Amazon RDS DB snapshot
- Sharing a manual Amazon RDS DB snapshot
- Copying a snapshot
- Restoring from an Amazon RDS DB snapshot
- Getting started with AWS DMS
- Using an Oracle database as a source for AWS DMS
- Using an Oracle database as a target for AWS DMS

- [AWS DMS setup using VPC peering]
- [How do I share manual Amazon RDS DB snapshots or DB cluster snapshots with another AWS account?] (AWS Knowledge Center article)

Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX

Deepak Kumar, Amazon Web Services

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

This pattern describes the use of VMware Hybrid Cloud Extension (HCX) to migrate your on-premises virtual machines (VMs) and applications to VMware Cloud on Amazon Web Services (AWS). The migration uses VMware enterprise-class software-defined data center (SDDC) software on the AWS Cloud to provide optimized access to AWS services.

VMware Cloud on AWS integrates compute, storage, and network virtualization products (vSphere, vSAN, and VMware NSX) with VMware vCenter server management, which is optimized to run on dedicated, elastic, bare-metal AWS infrastructure. The resulting infrastructure is low-maintenance, simplified, and hyper-converged.

With this service, IT teams can manage their cloud-based resources with familiar VMware tools. For more information, see VMware Cloud on AWS on the VMware website.

VMware HCX supports three types of cloud migrations:

- **Hybridity (data center extension):** Extending an existing, on-premises VMware SDDC to AWS to provide footprint expansion, on-demand capacity, a testing/development environment, and virtual desktops.
- **Cloud evacuation (data center-wide infrastructure refresh):** Consolidating data centers and moving completely to the AWS Cloud (including handling data center co-location or end of lease).
- **Application-specific migration:** Moving individual applications to the AWS Cloud to meet specific business needs.

Prerequisites and limitations

Prerequisites

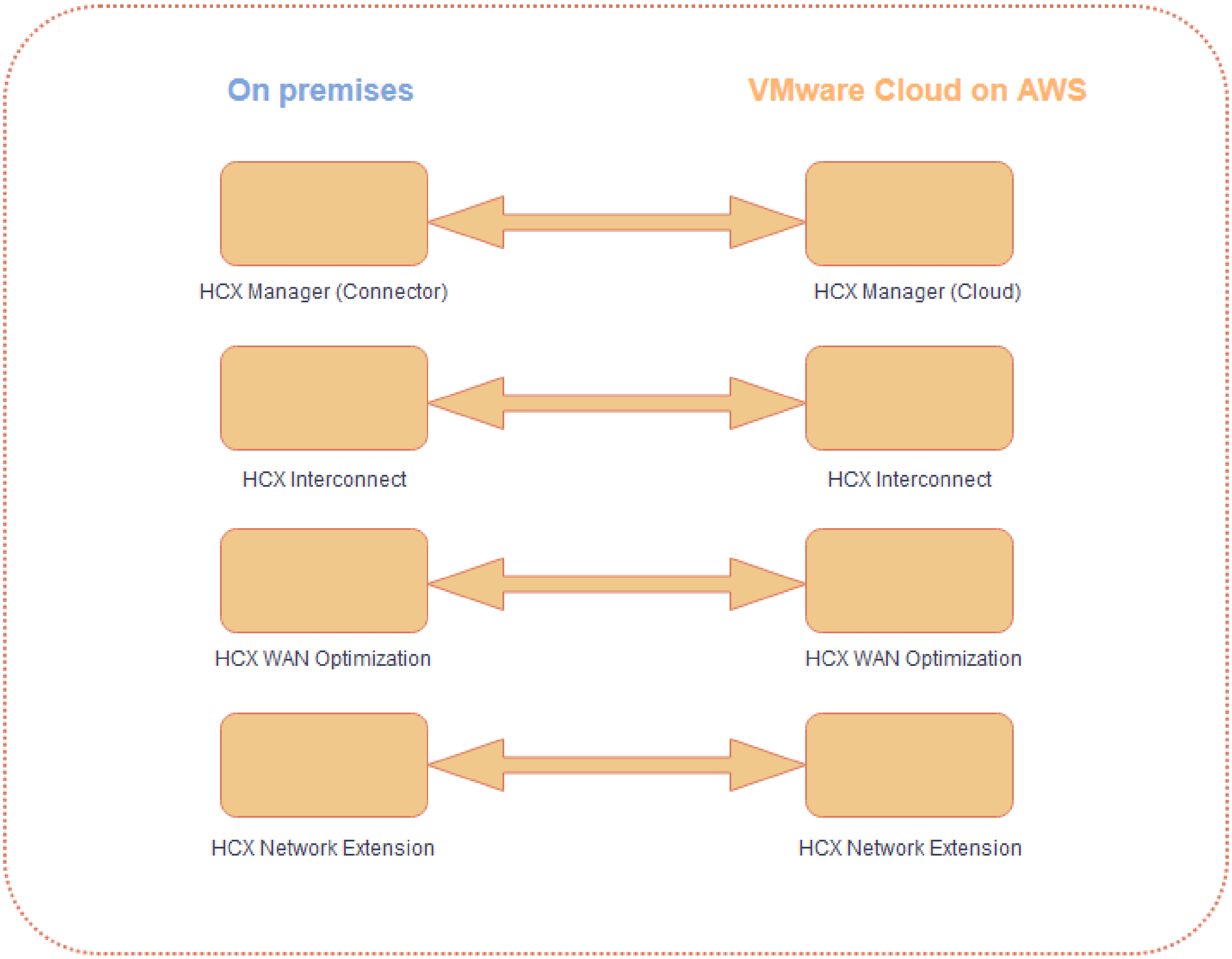
- Sign up for an AWS account (required for VMware Cloud SDDC creation).
- Sign up for a My VMware account. Register at https://my.vmware.com/web/vmware/ and fill out all fields.
- Check the version of vCenter and hosts, and collect the number of VMs. If possible, ask for an RVTools export to display information about your virtual environments. We recommend vCenter version 6.0 or later.
- You must deploy distributed virtual switches if you want to extend data center networks (L2), test vMotion by using HCX, or analyze application dependency by using vRealize Network Insight.
- Pick a non-conflicting on-premises current management subnet network to create the SDDC on VMware Cloud on AWS.
- Validate HCX requirements by reviewing the prerequisites provided in the VMware HCX User Guide.
- Identify and group VMs for waves of migration. Check for VMs that you can use for testing.
- Collect any data about relative bandwidth consumption, WAN compression, and data transfer speed.

Notes

- No need for VMware NSX-V or NSX-T on premises.
- No additional costs for HCX (it's included in VMware Cloud on AWS).

Architecture

The following diagram shows the HCX solution built on multiple component services. Each component supports a specific function in the HCX solution. For more information about each HCX component, see the blog post Migrating Workloads to VMware Cloud on AWS with Hybrid Cloud Extension (HCX).



Source technology stack

- On-premises VMs and applications managed by VMware vSphere

Target technology stack

- VMware Cloud on AWS

Tools

- VMware HCX – VMware HCX is a tool that you can use to migrate your applications and workloads across data centers and cloud environments. It is included with VMware Cloud on AWS.

Epics

Plan the migration

Task	Description	Skills required
Choose a migration strategy.	Decide whether you want to extend your data center (hybridity), move all your data centers (cloud evacuation), or move specific applications to AWS.	SysAdmin, App owner
Validate HCX requirements.	For migration information, review the VMware HCX User Guide.	SysAdmin, App owner

Migrate to VMware Cloud on AWS

Task	Description	Skills required
Migrate your VMs or applicati ons.	For more information, see Hybrid Migration with VMware HCX in the VMware documentation.	SysAdmin, App owner

Related resources

- VMware Cloud on AWS: Getting Started
- Hybrid Migration with VMware HCX

- VMware HCX User Guide
- VMware Cloud on AWS Pricing
- VMware Cloud on AWS Roadmap

Migrate an Amazon RDS DB instance to another VPC or account

Dhrubajyoti Mukherjee, Amazon Web Services

Summary

This pattern provides guidance for migrating an Amazon Relational Database Service (Amazon RDS) DB instance from one virtual private cloud (VPC) to another in the same AWS account, or from one AWS account to another AWS account.

This pattern is useful if you want to migrate your Amazon RDS DB instances to another VPC or account for separation or security reasons (for example, when you want to place your application stack and database in different VPCs).

Migrating a DB instance to another AWS account involves steps such as taking a manual snapshot, sharing it, and restoring the snapshot in the target account. This process can be time-consuming, depending on database changes and transaction rates. It also causes database downtime, so plan ahead for the migration. Consider a blue/green deployment strategy to minimize downtime. Alternatively, you can evaluate AWS Data Migration Service (AWS DMS) to minimize downtime for the change. However, this pattern doesn't cover this option. To learn more, see the AWS DMS documentation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Identity and Access Management (IAM) permissions required for the VPC, subnets, and Amazon RDS console

Limitations

- Changes to a VPC cause a database reboot, resulting in application outages. We recommend that you migrate during low peak times.
- Limitations when migrating Amazon RDS to another VPC:
  - The DB instance you're migrating must be a single instance with no standby. It must not be a member of a cluster.
  - Amazon RDS must not be in multiple Availability Zones.

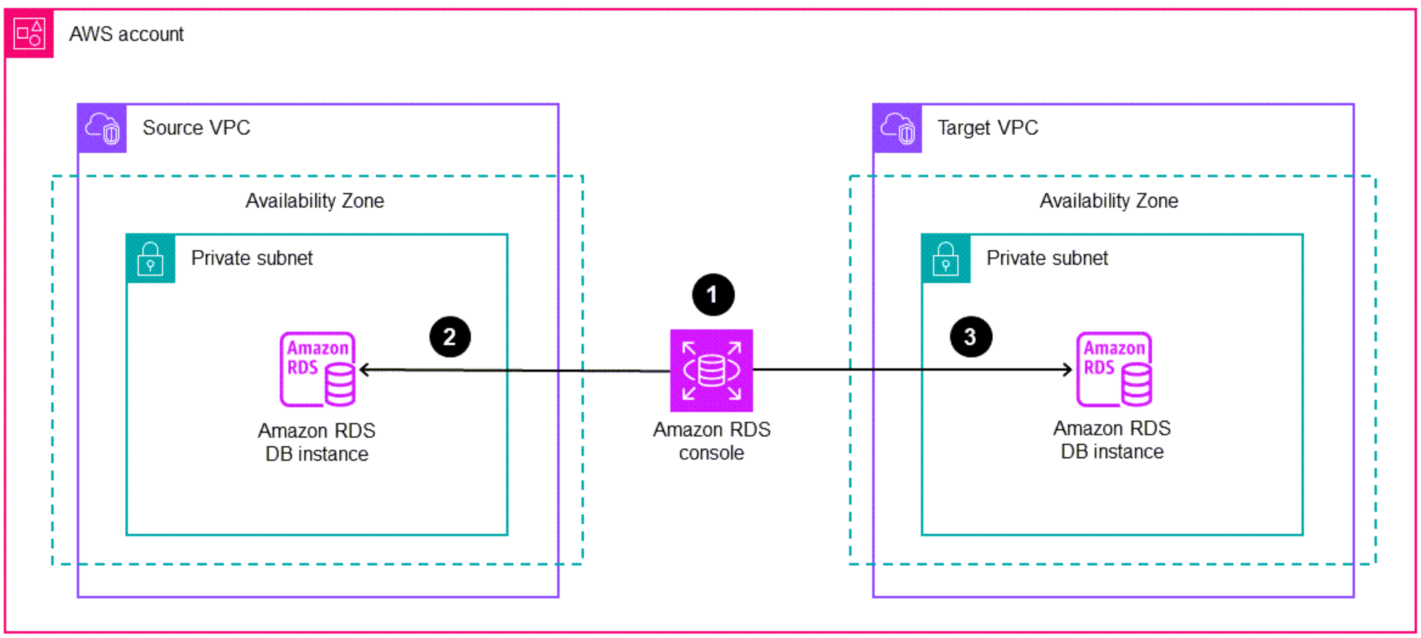
- Amazon RDS must not have any read replicas.
- The subnet group created in the target VPC must have subnets from the Availability Zone where the source database is running.

- Limitations when migrating Amazon RDS to another AWS account:
  - Sharing snapshots encrypted with the default service key for Amazon RDS isn't currently supported.

Architecture

Migrating to a VPC in the same AWS account

The following diagram shows the workflow for migrating an Amazon RDS DB instance to a different VPC in the same AWS account.

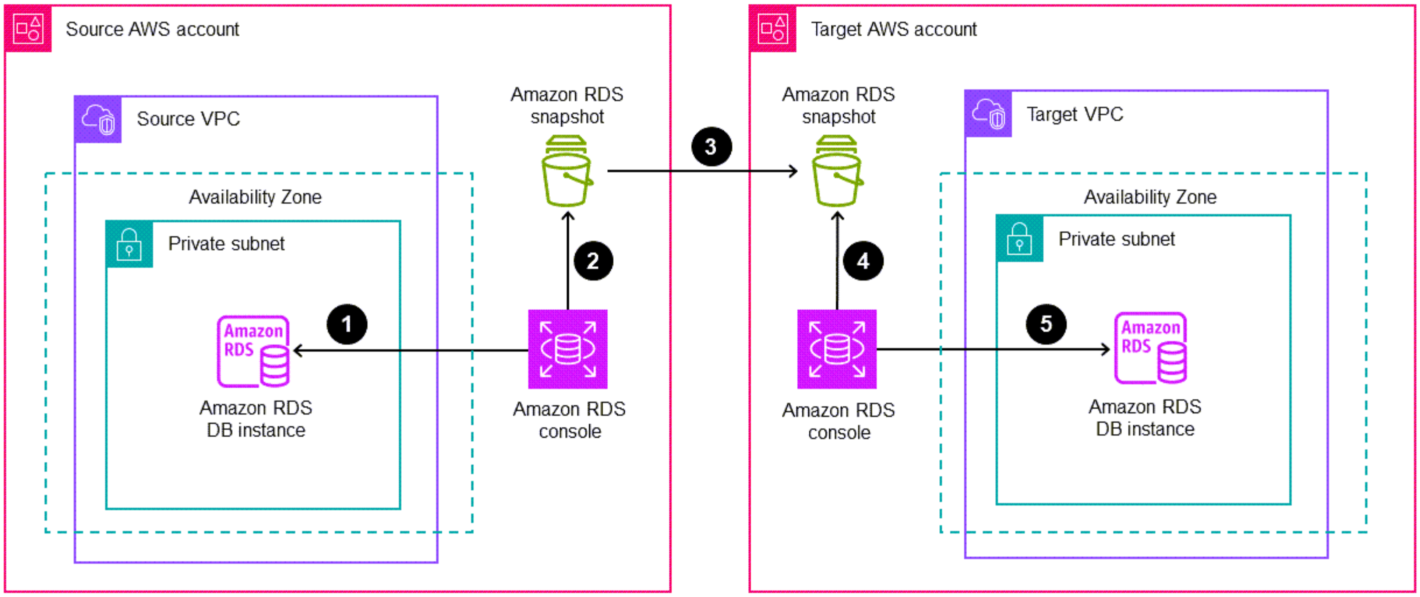


The steps consist of the following. See the Epics section for detailed instructions.

1. Create a DB subnet group in the target VPC. A DB subnet group is a collection of subnets that you can use to specify a specific VPC when you create DB instances.
2. Configure the Amazon RDS DB instance in the source VPC to use the new DB subnet group.
3. Apply the changes to migrate the Amazon RDS DB to the target VPC.

Migrating to a different AWS account

The following diagram shows the workflow for migrating an Amazon RDS DB instance to a different AWS account.



The steps consist of the following. See the Epics section for detailed instructions.

1. Access the Amazon RDS DB instance in the source AWS account.
2. Create an Amazon RDS snapshot in the source AWS account.
3. Share the Amazon RDS snapshot with the target AWS account.
4. Access the Amazon RDS snapshot in the target AWS account.
5. Create an Amazon RDS DB instance in the target AWS account.

Tools

AWS services

- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Best practices

- If database downtime is a concern when migrating an Amazon RDS DB instance to another account, we recommend that you use AWS DMS. This service provides data replication, which causes less than five minutes of outage time.

Epics

Migrate to a different VPC in the same AWS account

Task	Description	Skills required
Create a new VPC.	On the Amazon VPC console, create a new VPC and subnets with the desired propertie s and IP address ranges. For detailed instructions, see the Amazon VPC documentation.	Administrator
Create a DB subnet group.	On the Amazon RDS console: 1. Choose **Subnet groups**, **Create DB subnet group.** 2. Enter the subnet group name, description, and VPC ID. 3. Add the subnets that belong to the subnet group. Add subnets to cover at least two Availabil ity Zones. 4. Choose **Create**. For additional informati on, see the Amazon RDS documentation.	Administrator

Task	Description	Skills required
Modify the Amazon RDS DB instance to use the new subnet group.	On the Amazon RDS console: 1. In the navigation pane, choose **Databases**, and then choose the Amazon RDS DB instance to be migrated. 2. In the **Connectivity** section, choose the subnet group that's associated with the target VPC. 3. In the **Schedule modificat ions** section, choose **Apply immediately**. When the migration to the target VPC is complete, the target VPC's default security group is assigned to the Amazon RDS DB instance. You can configure a new security group for that VPC with the required inbound and outbound rules to your DB instance. Alternatively, use the AWS Command Line Interface (AWS CLI) to perform the migration to the target VPC by explicitly providing the new VPC security group ID. For example:	Administrator

Task	Description	Skills required
	`aws rds modify-db-instance \ --db-instance-identifier testrds \ --db-subnet-group-name new-vpc-subnet-group \ --vpc-security-group-ids sg-idxxxx \ --apply-immediately`	

Migrate to a different AWS account

Task	Description	Skills required
Create a new VPC and subnet group in the target AWS account.	1. On the Amazon VPC console, create a new VPC with the desired properties and IP address ranges. For detailed instructions, see the Amazon VPC documentation. 2. Create subnets for the new VPC by following the instructions in the Amazon VPC documentation. 3. On the Amazon RDS console, create DB subnet groups. For instructions, see the Amazon RDS documentation.	Administrator

Task	Description	Skills required
Share a manual snapshot of the database and share it with the target account.	1. Take a manual snapshot of the source database by following the instructi ons in the Amazon RDS documentation. 2. Share the snapshot with the target AWS account by providing the target account ID. For instructi ons, see the re:Post article about sharing DB snapshots with other accounts.	Administrator
Launch a new Amazon RDS DB instance.	Launch a new Amazon RDS DB instance from the shared snapshot in the target AWS account. For instructions, see the Amazon RDS documenta tion.	Administrator

Related resources

- Amazon VPC documentation
- Amazon RDS documentation
- How do I change the VPC for an RDS DB instance? (AWS re:Post article)
- How do I transfer ownership of Amazon RDS resources to a different AWS account? (AWS re:Post article)
- How do I share manual Amazon RDS DB snapshots or Aurora DB cluster snapshots with another AWS account? (AWS re:Post article)
- AWS DMS documentation

Migrate an Amazon RDS for Oracle DB instance to another VPC

Pinesh Singal, Amazon Web Services

Summary

This migration pattern provides step-by-step guidance for migrating an Amazon Relational Database Service (Amazon RDS) for Oracle database (DB) instance from one virtual private cloud (VPC) to another VPC in same Amazon Web Services (AWS) account. For example, you can use this pattern if your business requires that the database and the Amazon Elastic Compute Cloud (Amazon EC2) application server are in the same VPC.

The pattern describes an online migration strategy with almost no downtime for a multi-terabyte Oracle source database with a high number of transactions.

To move an Amazon RDS for Oracle DB instance to another VPC, you must change the Amazon RDS subnet group. This subnet group needs to be preconfigured with the new VPC and required subnets. During the VPC change from one network to another, the Amazon RDS instance reboots, so the database won't be accessible while the movement is in progress.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Two VPCs with private subnets
- An Amazon RDS for Oracle database instance (up and running), configured with inbound and outbound security groups

Limitations

- A DB instance that spans multiple Availability Zones (Multi-AZ) is not supported. This pattern, however, provides a way to work around this limitation.
- The DB instance can’t be migrated while a read replica is turned on.
- The subnet group in the new VPC should be in the same Availability Zone as the database.
- Migration should occur during scheduled maintenance period or low-traffic times, because moving the DB to another VPC causes a database reboot, resulting in application outages for few minutes.

Product versions

- Amazon RDS for Oracle DB instance, 12.1.0.2 and later

Architecture

Source technology stack

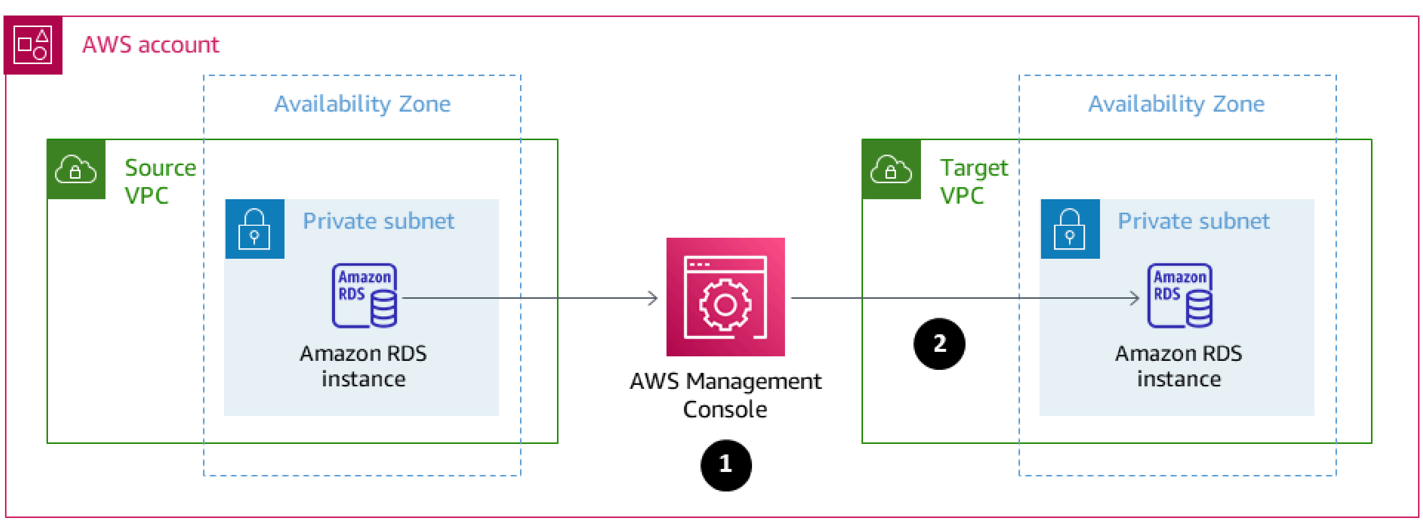
- An Amazon RDS for Oracle 12.1.0.2.v22 DB instance in a VPC
- A VPC configured in a separate route table
- Amazon RDS subnet groups configured in a VPC
- Amazon RDS option groups (if needed)

Target technology stack

- Amazon RDS for Oracle database instance with version 12.1.0.2.v22 in another VPC
- Amazon VPC configured in separate route
- Amazon RDS Subnet Groups configured in new VPC
- Amazon RDS Option Groups (if needed)

Source and target architecture

The following diagram shows using the console to move the Amazon RDS for Oracle DB from a private subnet in one VPC to a private subnet in a different VPC.



1. Use the console to modify the source Amazon RDS for Oracle DB instance.
2. In the target VPC, modify the subnet group, and modify the option group if used.

Tools

- Amazon RDS – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for a relational database and manages common database administration tasks. This pattern uses Amazon RDS for Oracle.

Epics

Change the configuration of the Amazon RDS for Oracle database in the existing VPC

Task	Description	Skills required
Create a subnet group.	Configure a subnet group in Amazon RDS.	General AWS
Create an option group.	(Optional) Configure an option group in Amazon RDS.	General AWS
Modify the Amazon RDS for Oracle DB instance.	Modify the database with the subnet group and option group.	General AWS, DBA
Update the Oracle database, if necessary.	To migrate the source Amazon RDS for Oracle database, make the following changes: • Remove read replicas, if they exist. • Turn off the Multi-AZ feature, if it's turned on.	General AWS

Configure the Amazon RDS for Oracle database in the target VPC

Task	Description	Skills required
Create a subnet group.	In Amazon RDS, configure a subnet group using the subnet of the new VPC and the Availability Zone of the database.	General AWS
Create an option group.	(Optional) Configure an option group in Amazon RDS.	General AWS
Modify the Amazon RDS for Oracle database.	Modify the database with new subnet group and option group of the new VPC. You can apply these changes immediately or in a maintenance window. The modification can take several minutes to complete. During the modification, you will see the following status changes: • moving-to-vpc • Configuring-enhanced-monitoring • Modifying • Available The modification will attach the default security group of the new VPC. Attach a new security group as needed by Amazon RDS for Oracle.	General AWS, DBA

Task	Description	Skills required
Update the Amazon RDS for Oracle database, if necessary.	After migrating to the target Amazon RDS for Oracle database in the new VPC, make the following modifications, if needed: • Turn on read replicas, if they existed in the source database. • Turn on the Multi-AZ feature, if it was turned on in the source database.	General AWS
Test application connectivity.	Perform a database connectivity test from any application. Confirm that the modified Amazon RDS for Oracle DB in the new VPC is connected and is accessible from the application.	App owner

Related resources

- Amazon VPC documentation
- VPCs and subnets
- Working with a DB instance in a VPC
- Amazon RDS documentation
- Oracle on Amazon RDS
- Amazon RDS console
- How do I change the VPC for an Amazon RDS DB instance?

Migrate an Amazon Redshift cluster to an AWS Region in China

Jing Yan, Amazon Web Services

Summary

This pattern provides a step-by-step approach to migrate an Amazon Redshift cluster to an AWS Region in China from another AWS Region.

This pattern uses SQL commands to recreate all the database objects, and uses the UNLOAD command to move this data from Amazon Redshift to an Amazon Simple Storage Service (Amazon S3) bucket in the source Region. The data is then migrated to an S3 bucket in the AWS Region in China. The COPY command is used to load data from the S3 bucket and transfer it to the target Amazon Redshift cluster.

Amazon Redshift doesn't currently support cross-Region features such as snapshot copying to AWS Regions in China. This pattern provides a way to work around that limitation. You can also reverse the steps in this pattern to migrate data from an AWS Region in China to another AWS Region.

Prerequisites and limitations

Prerequisites

- Active AWS accounts in both a China Region and an AWS Region outside China
- Existing Amazon Redshift clusters in both a China Region and an AWS Region outside China

Limitations

- This is an offline migration, which means the source Amazon Redshift cluster cannot perform write operations during the migration.

Architecture

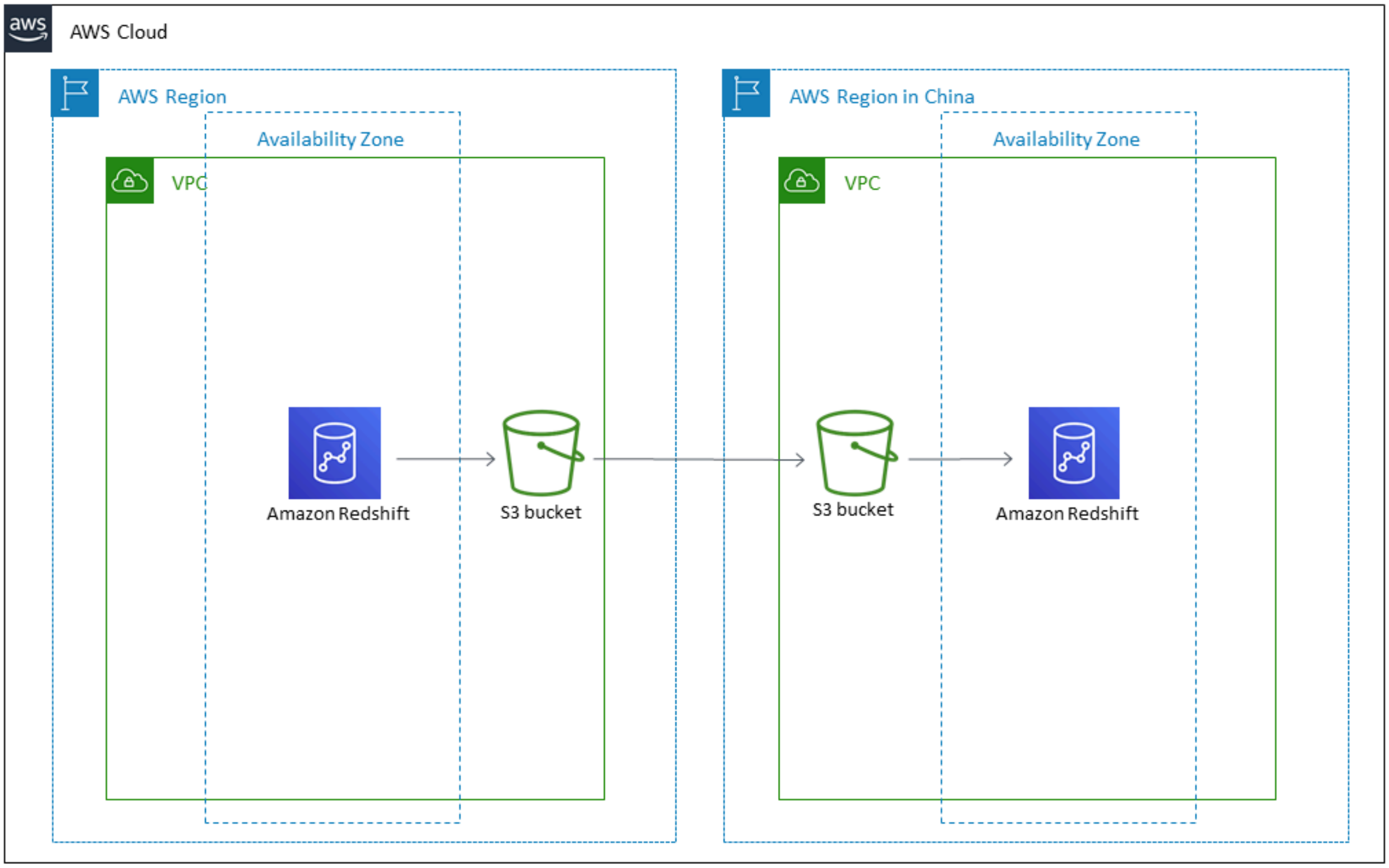
Source technology stack

- Amazon Redshift cluster in an AWS Region outside China

Target technology stack

- Amazon Redshift cluster in an AWS Region in China

Target architecture



Tools

Tools

- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers scalability, data availability, security, and performance. You can use Amazon S3 to store data from Amazon Redshift, and you can copy data from an S3 bucket to Amazon Redshift.
- Amazon Redshift – Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud.
- psql – psql is a terminal-based front-end to PostgreSQL.

Epics

Prepare for migration in the source Region

Task	Description	Skills required
Launch and configure an EC2 instance in the source Region.	Sign in to the AWS Management Console and open the Amazon Elastic Compute Cloud (Amazon EC2) console. Your current Region is displayed in the navigation bar at the top of the screen. This Region cannot be an AWS Region in China. From the Amazon EC2 console dashboard, choose "Launch instance," and create and configure an EC2 instance. Important: Ensure your EC2 security groups for inbound rules allow unrestricted access to TCP port 22 from your source machine. For instructi ons on how to launch and configure an EC2 instance, see the "Related resources" section.	DBA, Developer
Install the psql tool.	Download and install PostgreSQL. Amazon Redshift does not provide the psql tool, it is installed with PostgreSQL. For more information about using psql and installing PostgreSQ	DBA

Task	Description	Skills required
	L tools, see the "Related resources" section.	
Record the Amazon Redshift cluster details.	Open the Amazon Redshift console, and choose "Clusters " in the navigation pane. Then choose the Amazon Redshift cluster name from the list. On the "Properties" tab, in the "Database configurations" section, record the "Database name" and "Port." Open the "Connection details" section and record the "Endpoint ," which is in the "endpoint :<port>/<databasename>" format. Important: Ensure your Amazon Redshift security groups for inbound rules allow unrestricted access to TCP port 5439 from your EC2 instance.	DBA

Task	Description	Skills required
Connect psql to the Amazon Redshift cluster.	At a command prompt, specify the connection information by running the "psql -h <endpoint> -U <userid> -d <databasename> -p <port>" command. At the psql password prompt, enter the password for the "<userid>" user. You are then connected to the Amazon Redshift cluster and can interactively enter commands.	DBA
Create an S3 bucket.	Open the Amazon S3 console, and create an S3 bucket to hold the files exported from Amazon Redshift. For instructi ons on how to create an S3 bucket, see the "Related resources" section.	DBA, AWS General

Task	Description	Skills required
Create an IAM policy that supports unloading data.	Open the AWS Identity and Access Management (IAM) console and choose "Policies ." Choose "Create policy," and choose the "JSON" tab. Copy and paste the IAM policy for unloading data from the "Additional information" section. Important: Replace "s3_bucket_name" with your S3 bucket's name. Choose "Review policy," and enter a name and description for the policy. Choose "Create policy."	DBA
Create an IAM role to allow UNLOAD operation for Amazon Redshift.	Open the IAM console and choose "Roles." Choose "Create role," and choose "AWS service" in "Select type of trusted entity." Choose "Redshift" for the service, choose "Redshift – Customiza ble," and then choose "Next." Choose the "Unload" policy you created earlier, and choose "Next." Enter a "Role name," and choose "Create role."	DBA

Task	Description	Skills required
Associate IAM role with the Amazon Redshift cluster.	Open the Amazon Redshift console, and choose "Manage IAM roles." Choose "Availabl e roles" from the dropdown menu and choose the role you created earlier. Choose "Apply changes." When the "Status" for the IAM role on the "Manage IAM roles" shows as "In-sync", you can run the UNLOAD command.	DBA
Stop write operations to the Amazon Redshift cluster.	You must remember to stop all write operations to the source Amazon Redshift cluster until the migration is complete.	DBA

Prepare for migration in the target Region

Task	Description	Skills required
Launch and configure an EC2 instance in the target Region.	Sign in to the AWS Management Console for a Region in China, either Beijing or Ningxia. From the Amazon EC2 console, choose "Launch instance," and create and configure an EC2 instance. Important: Make sure your Amazon EC2 security groups for inbound rules allow unrestricted access to TCP port 22 from your source	DBA

Task	Description	Skills required
	machine. For further instructions on how to launch and configure an EC2 instance, see the "Related resources" section.	
Record the Amazon Redshift cluster details.	Open the Amazon Redshift console, and choose "Clusters " in the navigation pane. Then choose the Amazon Redshift cluster name from the list. On the "Properties" tab, in the "Database configurations" section, record the "Database name" and "Port." Open the "Connection details" section and record the "Endpoint ," which is in the "endpoint :<port>/<databasename>" format. Important: Make sure your Amazon Redshift security groups for inbound rules allow unrestricted access to TCP port 5439 from your EC2 instance.	DBA

Task	Description	Skills required
Connect psql to the Amazon Redshift cluster.	At a command prompt, specify the connection information by running the "psql -h <endpoint> -U <userid> -d <databasename> -p <port>" command. At the psql password prompt, enter the password for the "<userid>" user. You are then connected to the Amazon Redshift cluster and can interactively enter commands.	DBA
Create an S3 bucket.	Open the Amazon S3 console, and create an S3 bucket to hold the exported files from Amazon Redshift. For help with this and other stories, see the "Related resources" section.	DBA
Create an IAM policy that supports copying data.	Open the IAM console and choose "Policies." Choose "Create policy," and choose the "JSON" tab. Copy and paste the IAM policy for copying data from the "Additional information" section. Important: Replace "s3_bucket_name" with your S3 bucket's name. Choose "Review policy," enter a name and description for the policy. Choose "Create policy."	DBA

Task	Description	Skills required
Create an IAM role to allow COPY operation for Amazon Redshift.	Open the IAM console and choose "Roles." Choose "Create role," and choose "AWS service" in "Select type of trusted entity." Choose "Redshift" for the service, choose "Redshift – Customiza ble," and then choose "Next." Choose the "Copy" policy you created earlier, and choose "Next." Enter a "Role name," and choose "Create role."	DBA
Associate IAM role with the Amazon Redshift cluster.	Open the Amazon Redshift console, and choose "Manage IAM roles." Choose "Availabl e roles" from the dropdown menu and choose the role you created earlier. Choose "Apply changes." When the "Status" for the IAM role on the "Manage IAM roles" shows as "In-sync", you can run the "COPY" command.	DBA

Verify source data and object information before beginning the migration

Task	Description	Skills required
Verify the rows in the source Amazon Redshift tables.	Use the scripts in the "Additional information" section to verify and record the number of rows in the source Amazon Redshift	DBA

Task	Description	Skills required
	tables. Remember to split the data evenly for the UNLOAD and COPY scripts. This will improve the data unloading and loading efficiency, because the data quantity covered by each script will be balanced.	
Verify the number of database objects in the source Amazon Redshift cluster.	Use the scripts in the "Additional information" section to verify and record the number of databases, users, schemas, tables, views, and user-defined functions (UDFs) in your source Amazon Redshift cluster.	DBA
Verify SQL statement results before migration.	Some SQL statements for data validation should be sorted according to actual business and data situations. This is to verify the imported data to ensure it is consistent and displayed correctly.	DBA

Migrate data and objects to the target Region

Task	Description	Skills required
Generate Amazon Redshift DDL scripts.	Generate Data Definition Language (DDL) scripts by using the links from the "SQL statements to query Amazon Redshift" section	DBA

Task	Description	Skills required
	in the "Additional informati on" section. These DDL scripts should include the "create user," "create schema," "privileges on schema to user," "create table/view," "privileges on objects to user," and "create function" queries.	
Create objects in the Amazon Redshift cluster for the target Region.	Run the DDL scripts by using the AWS Command Line Interface (AWS CLI) in the AWS Region in China. These scripts will create objects in the Amazon Redshift cluster for the target Region.	DBA
Unload source Amazon Redshift cluster data to the S3 bucket.	Run the UNLOAD command to unload data from the Amazon Redshift cluster in the source Region to the S3 bucket.	DBA, Developer
Transfer source Region S3 bucket data to target Region S3 bucket.	Transfer the data from your source Region S3 bucket to the target S3 bucket. Because the "$ aws s3 sync" command cannot be used, make sure you use the process outlined in the "Transferring Amazon S3 data from AWS Regions to AWS Regions in China" article in the "Related resources" section.	Developer

Task	Description	Skills required
Load data into the target Amazon Redshift cluster.	In the psql tool for your target Region, run the COPY command to load data from the S3 bucket to the target Amazon Redshift cluster.	DBA

Verify the data in the source and target Regions after the migration

Task	Description	Skills required
Verify and compare the number of rows in the source and target tables.	Verify and compare the number of table rows in the source and target Regions to ensure all are migrated.	DBA
Verify and compare the number of source and target database objects.	Verify and compare all database objects in the source and target Regions to ensure all are migrated.	DBA
Verify and compare SQL script results in the source and target Regions.	Run the SQL scripts prepared before the migration. Verify and compare the data to ensure that the SQL results are correct.	DBA
Reset the passwords of all users in the target Amazon Redshift cluster.	After the migration is complete and all data is verified, you should reset all user passwords for the Amazon Redshift cluster in the AWS Region in China.	DBA

Related resources

- Transferring Amazon S3 data from AWS Regions to AWS Regions in China
- Creating an S3 bucket
- Resetting an Amazon Redshift user password
- psql documentation

Additional information

IAM policy for unloading data

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::s3_bucket_name"]
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject", "s3:DeleteObject"],
      "Resource": ["arn:aws:s3:::s3_bucket_name/*"]
    }
  ]
}
```

IAM policy for copying data

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::s3_bucket_name"]
    },
    {
      "Effect": "Allow",
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::s3_bucket_name/*"]
```

```
    }
  ]
}
```

SQL statements to query Amazon Redshift

```
##Database

select * from pg_database where datdba>1;

##User

select * from pg_user where usesysid>1;

##Schema

SELECT n.nspname AS "Name",

  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"

FROM pg_catalog.pg_namespace n

WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'

ORDER BY 1;

##Table

select count(*) from pg_tables where schemaname not in
 ('pg_catalog','information_schema');

select schemaname,count(*) from pg_tables where schemaname not in
 ('pg_catalog','information_schema') group by schemaname order by 1;

##View

SELECT

    n.nspname AS schemaname,c.relname AS
 viewname,pg_catalog.pg_get_userbyid(c.relowner) as "Owner"

FROM
```

```
    pg_catalog.pg_class AS c

INNER JOIN

    pg_catalog.pg_namespace AS n

    ON c.relnamespace = n.oid

WHERE relkind = 'v' and n.nspname not in ('information_schema','pg_catalog');

##UDF

SELECT

   n.nspname AS schemaname,

   p.proname AS proname,

   pg_catalog.pg_get_userbyid(p.proowner) as "Owner"

FROM pg_proc p

LEFT JOIN pg_namespace n on n.oid = p.pronamespace

WHERE p.proowner != 1;
```

SQL scripts to generate DDL statements

- Get_schema_priv_by_user script
- Generate_tbl_ddl script
- Generate_view_ddl
- Generate_user_grant_revoke_ddl
- Generate_udf_ddl

Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport

Raunak Rishabh and Jitender Kumar, Amazon Web Services

Summary

This pattern describes the steps for migrating extremely large databases between two Amazon Relational Database Service (Amazon RDS) for PostgreSQL DB instances by using the **pg_transport** extension. This extension provides a physical transport mechanism to move each database. By streaming the database files with minimal processing, it provides an extremely fast method for migrating large databases between DB instances with minimal downtime. This extension uses a pull model where the target DB instance imports the database from the source DB instance.

Prerequisites and limitations

Prerequisites

- Both DB instances must run the same major version of PostgreSQL.
- The database must not exist on the target. Otherwise, the transport fails.
- No extension other than **pg_transport** must be enabled in the source database.
- All source database objects must be in the default **pg_default** tablespace.
- The security group of the source DB instance should allow traffic from the target DB instance.
- Install a PostgreSQL client like [psql] or [PgAdmin] to work with the Amazon RDS PostgreSQL DB instance. You can install the client either in your local system or use an Amazon Elastic Compute Cloud (Amazon EC2) instance. In this pattern, we use psql on an EC2 instance.

Limitations

- You can't transport databases between different major versions of Amazon RDS for PostgreSQL.
- The access privileges and ownership from the source database are not transferred to the target database.
- You can't transport databases on read replicas or on parent instances of read replicas.
- You can't use **reg** data types in any database tables that you plan to transport with this method.
- You can run up to 32 total transports (including both imports and exports) at the same time on a DB instance.

- You cannot rename or include/exclude tables. Everything is migrated as is.

Caution

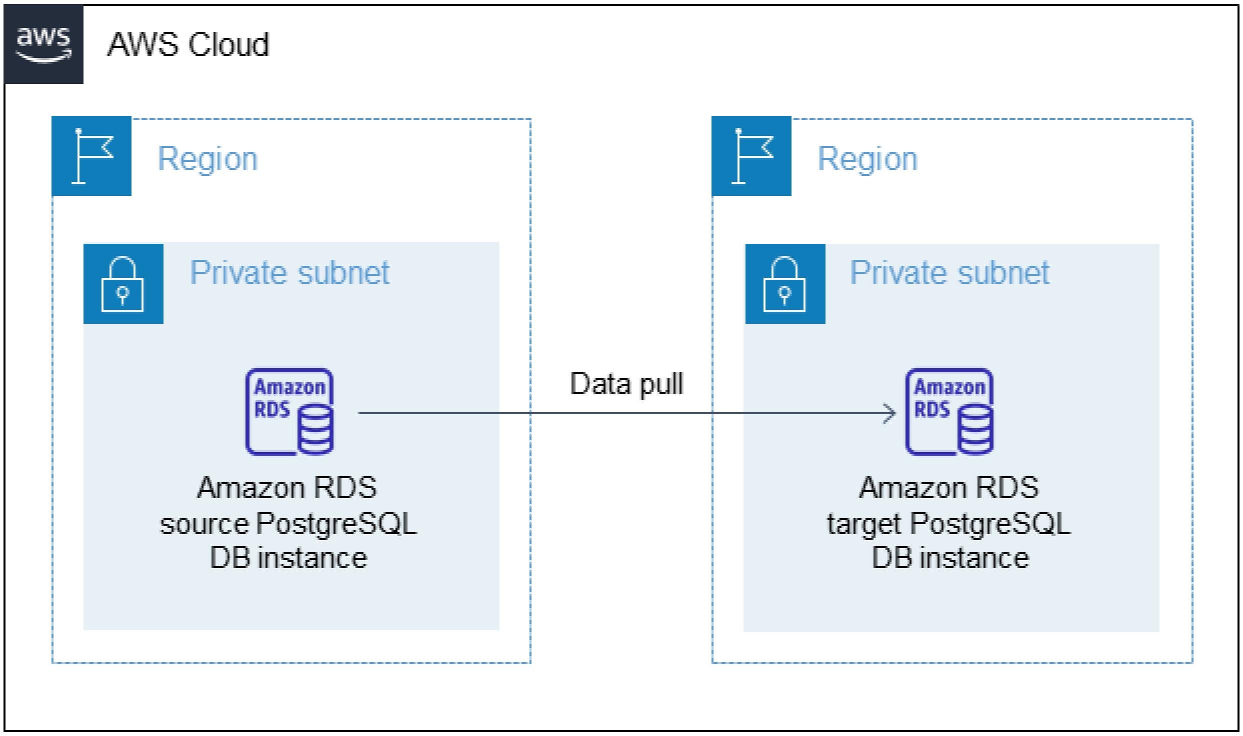
- Make backups before removing the extension, because removing the extension also removes dependent objects and some data that's critical to the operation of the database.
- Consider the instance class and processes running on other databases on the source instance when you determine the number of workers and `work_mem` values for **pg_transport**.
- When the transport starts, all connections on the source database are ended and the database is put into read-only mode.

> **Note**
>
> When the transport is running on one database, it doesn't affect other databases on the same server.

Product versions

- Amazon RDS for PostgreSQL 10.10 and later, and Amazon RDS for PostgreSQL 11.5 and later. For the latest version information, see Transporting PostgreSQL Databases Between DB Instances in the Amazon RDS documentation.

Architecture



Tools

- **pg_transport** provides a physical transport mechanism to move each database. By streaming the database files with minimal processing, physical transport moves data much faster than traditional dump and load processes and requires minimal downtime. PostgreSQL transportable databases use a pull model where the destination DB instance imports the database from the source DB instance. You install this extension on your DB instances when you prepare the source and target environments, as explained in this pattern.
- psql enables you to connect to, and work with, your PostgreSQL DB instances. To install **psql** on your system, see the PostgreSQL Downloads page.

Epics

Create the target parameter group

Task	Description	Skills required
Create a parameter group for the target system.	Specify a group name that identifies it as a target	DBA

Task	Description	Skills required
	parameter group; for example, `pgtarget-param-group` . For instructions, see the [Amazon RDS documentation]().	

Task	Description	Skills required
Modify the parameters for the parameter group.	Set the following parameters: 1. Add `pg_transport` to the `shared_preload_libraries` parameter. `shared_preload_libraries = pg_stat_statements, pg_transport` 2. Set the `pg_transport.num_workers` parameter. Choose the number of workers you want to run the transport with. The value you set determines the number of `transport.send_file` workers that will be created in the source. 3. Increase the the value of `max_worker_processes` to more than three times the value of `pg_transport.num_workers`. For example, if you set the value of `pg_transport.num_workers` to 4, the `max_worker_processes` value should be at least 13. If this fails,	DBA

Task	Description	Skills required
	pg_transport recommends a minimum value. 4. Set `pg_transport.timing` to 1. This setting enables the reporting of timing information during the transport. 5. Set the `pg_transport.work_mem` parameter. This parameter specifies the maximum memory to allocate to each worker. The default value is 128 MB. For more information about these parameters, see the [Amazon RDS documentation]().	

Create the source parameter group

Task	Description	Skills required
Create a parameter group for the source system.	Specify a group name that identifies it as a source parameter group; for example, `pgsource-param-group` . For instructions, see the [Amazon RDS documentation]().	DBA
Modify the parameters for the parameter group.	Set the following parameters:	DBA

Task	Description	Skills required
	1. Add `pg_transport` to the `shared_preload_libraries` parameter. `shared_preload_libraries = pg_stat_statements, pg_transport` 2. Set the `pg_transport.num_workers` parameter. The value of this parameter defined in the target determines the number of `transport.send_file` workers to be used. If you have an import running on this instance, increase this value, but consider the number of workers that are already running. 3. Increase the the value of `max_worker_processes` to more than three times the value of `pg_transport.num_workers` on the target. For example, if you set the value of `pg_transport.num_workers` to 4 on the target, the `max_worker_processes` value on the source	

Task	Description	Skills required
	should be at least 13. If this fails, **pg_transport** recommends a minimum value. 4. Set the `pg_transport.work_mem` parameter. This parameter specifies the maximum memory to allocate to each worker. The default value is 128 MB. For more information about these parameters, see the [Amazon RDS documentation]().	

Prepare the target environment

Task	Description	Skills required
Create a new Amazon RDS for PostgreSQL DB instance to transport your source database to.	Determine the instance class and PostgreSQL version based on your business requirements.	DBA, Systems administrator, Database architect
Modify the security group of the target to allow connections on the DB instance port from the EC2 instance.	By default, the port for the PostgreSQL instance is 5432. If you're using another port, connections to that port must be open for the EC2 instance.	DBA, Systems administrator
Modify the instance, and assign the new target parameter group.	For example, `pgtarget-param-group` .	DBA

Task	Description	Skills required
Restart the target Amazon RDS DB instance.	The parameters `shared_preload_libraries` and `max_worker_processes` are static parameters and require a reboot of the instance.	DBA, Systems administrator
Connect to the database from the EC2 instance using psql.	Use the command: `psql -h <rds_end_point> -p PORT -U username -d database -W`	DBA
Create the pg_transport extension.	Run the following query as a user with the `rds_superuser` role: `create extension pg_transport;`	DBA

Prepare the source environment

Task	Description	Skills required
Modify the security group of the source to allow connections on the DB instance port from the Amazon EC2 instance and target DB instance	By default, the port for PostgreSQL instance is 5432. If you're using another port, connections to that port must be open for the EC2 instance.	DBA, Systems administrator
Modify the instance and assign the new source parameter group.	For example, `pgsource-param-group` .	DBA

Task	Description	Skills required
Restart the source Amazon RDS DB instance.	The parameters `shared_preload_libraries` and `max_worker_processes` are static parameters and require a reboot of the instance.	DBA
Connect to the database from the EC2 instance using psql.	Use the command: `psql -h <rds_end_point> -p PORT -U username -d database -W`	DBA
Create the pg_transport extension and remove all other extensions from the databases to be transported.	The transport will fail if there are any extensions other than **pg_transport** installed on the source database. This command must by run by a user with the `rds_superuser` role.	DBA

Perform the transport

Task	Description	Skills required
Perform a dry run.	Use the `transport.import_from_server` function to perform a dry run first: `SELECT transport.import_from_server( 'source-db-instance-endpoint', source-db-instance-port,`	DBA

Task	Description	Skills required
	``` 'source-db-instance-user', 'source-user-password', 'source-database-name', 'destination-user-password', 'true'); ``` The last parameter of this function (set to `true`) defines the dry run. This function displays any errors that you would see when you run the main transport. Resolve the errors before you run the main transport.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>If the dry run is successful, initiate the database transport.</td><td>Run the <code>transport.import_from_server</code> function to perform the transport. It connects to the source and imports the data.<pre><code>SELECT transport.import_from_server('source-db-instance-endpoint', source-db-instance-port,
 'source-db-instance-user', 'source-user-password', 'source-database-name',
 'destination-user-password', false);</code></pre>The last parameter of this function (set to <code>false</code>) indicates that this isn't a dry run.</td><td>DBA</td></tr>
<tr><td>Perform post-transport steps.</td><td>After the database transport is complete:
Validate the data in the target environment.
Add all the roles and permissions to the target.
Enable all required extensions in the target and source, if required.
Revert the value of the <code>max_worker_processes</code> parameter.</td><td>DBA</td></tr>
</table>

Related resources

- Amazon RDS documentation
- pg_transport documentation
- Migrating databases using RDS PostgreSQL Transportable Databases (blog post)
- PostgreSQL downloads
- psql utility
- Creating a DB Parameter Group
- Modify Parameters in a DB Parameter Group
- PostgreSQL downloads

Replatform

Topics

- Export a Microsoft SQL Server database to Amazon S3 by using AWS DMS
- Migrate OpenText TeamSite workloads to the AWS Cloud
- Migrate Oracle CLOB values to individual rows in PostgreSQL on AWS
- Migrate an on-premises Oracle database to Amazon RDS for Oracle by using direct Oracle Data Pump Import over a database link
- Migrate Oracle PeopleSoft to Amazon RDS Custom
- Migrate Oracle ROWID functionality to PostgreSQL on AWS
- Migrate Oracle Database error codes to an Amazon Aurora PostgreSQL-Compatible database
- Migrate Redis workloads to Redis Enterprise Cloud on AWS
- Migrate SAP ASE on Amazon EC2 to Amazon Aurora PostgreSQL-Compatible using AWS SCT and AWS DMS
- Migrate Windows SSL certificates to an Application Load Balancer using ACM
- Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS
- Migrate a relational database to MongoDB Atlas on AWS
- Migrate a self-hosted MongoDB environment to MongoDB Atlas on AWS
- Migrate an Oracle JD Edwards EnterpriseOne database to AWS by using Oracle Data Pump and AWS DMS
- Migrate an Oracle PeopleSoft database to AWS by using AWS DMS
- Migrate an on-premises MySQL database to Amazon RDS for MySQL
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server
- Migrate data from Microsoft Azure Blob to Amazon S3 by using Rclone
- Migrate from Couchbase Server to Couchbase Capella on AWS
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling
- Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk
- Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS

- Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash
- Migrate an on-premises Oracle database to Amazon RDS for Oracle
- Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump
- Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical
- Migrate an on-premises PostgreSQL database to Aurora PostgreSQL
- Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using native backup and restore methods
- Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT
- Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools
- Migrate an on-premises MySQL database to Aurora MySQL
- Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3
- Migrate on-premises Java applications to AWS using AWS App2Container
- Migrate shared file systems in an AWS large migration
- Migrate an Oracle database to Amazon RDS for Oracle by using Oracle GoldenGate flat file adapters
- Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition
- Stream data from IBM Db2, SAP, Sybase, and other databases to MongoDB Atlas on AWS

Export a Microsoft SQL Server database to Amazon S3 by using AWS DMS

Sweta Krishna, Amazon Web Services

Summary

Organizations often need to copy databases to Amazon Simple Storage Service (Amazon S3) for database migration, backup and restore, data archiving, and data analytics. This pattern describes how you can export a Microsoft SQL Server database to Amazon S3. The source database can be hosted on premises or on Amazon Elastic Compute Cloud (Amazon EC2) or Amazon Relational Database Service (Amazon RDS) for Microsoft SQL Server on the Amazon Web Services (AWS) Cloud.

The data is exported by using AWS Database Migration Service (AWS DMS). By default, AWS DMS writes full load and change data capture (CDC) data in comma-separated value (.csv) format. For more compact storage and faster query options, this pattern uses the Apache Parquet (.parquet) format option.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS Identity and Access Management (IAM) role for the account with write, delete, and tag access to the target S3 bucket, and AWS DMS (`dms.amazonaws.com`) added as a trusted entity to this IAM role
- An on-premises Microsoft SQL Server database (or Microsoft SQL Server on an EC2 instance or an Amazon RDS for SQL Server database)
- Network connectivity between the virtual private cloud (VPC) on AWS and the on-premises network provided by AWS Direct Connect or a virtual private network (VPN)

Limitations

- A VPC-enabled (gateway VPC) S3 bucket isn't currently supported in AWS DMS versions earlier than 3.4.7.
- Changes to the source table structure during full load are not supported.
- AWS DMS full large binary object (LOB) mode is not supported.

Product versions

- Microsoft SQL Server versions 2005 or later for the Enterprise, Standard, Workgroup, and Developer editions.
- Support for Microsoft SQL Server version 2019 as a source is available in AWS DMS versions 3.3.2 and later.

Architecture

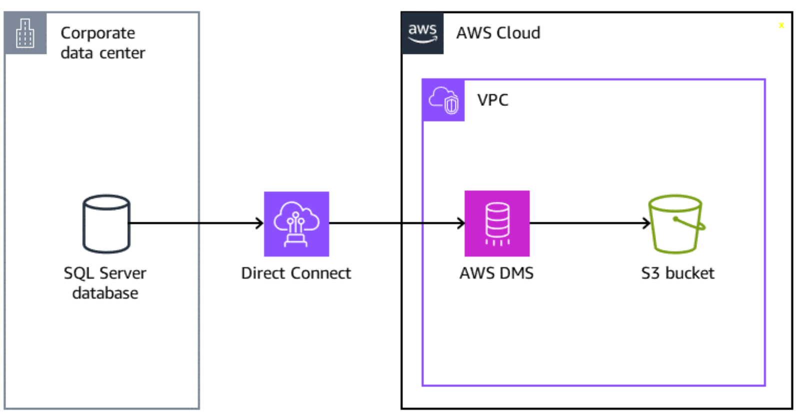
Source technology stack

- An on-premises Microsoft SQL Server database (or Microsoft SQL Server on an EC2 instance or an Amazon RDS for SQL Server database)

Target technology stack

- AWS Direct Connect
- AWS DMS
- Amazon S3

Target architecture



Tools

- [AWS Database Migration Service (AWS DMS)](#) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.

- AWS Direct Connect links your internal network to a Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Epics

Prepare for the migration

Task	Description	Skills required
Validate the database version.	Validate the source database version and make sure that it’s supported by AWS DMS. For information about supported SQL Server database versions, see Using a Microsoft SQL Server database as a source for AWS DMS.	DBA
Create a VPC and security group.	In your AWS account, create a VPC and security group. For more information, see the Amazon VPC documentation.	System Administrator
Create a user for the AWS DMS task.	Create an AWS DMS user in the source database and grant it READ permissions. This user will be used by AWS DMS.	DBA
Test the DB connectivity.	Test the connectivity to the SQL Server DB instance from the AWS DMS user.	DBA

Task	Description	Skills required
Create an S3 bucket.	Create the target S3 bucket. This bucket will hold the migrated table data.	Systems administrator
Create an IAM policy and role.	1. To create an IAM policy with bucket permissio ns, use the code in the *Additional information* section. 2. Create the role for AWS DMS, and attach the policy to the role.	Systems administrator

Migrate data by using AWS DMS

Task	Description	Skills required
Create an AWS DMS replicati on instance.	Sign in to the AWS Management Console, and open the AWS DMS console. In the navigation pane, choose **Replication instances** , **Create replication instance**. For instructions, see step 1 in the AWS DMS documentation.	DBA
Create source and target endpoints.	Create source and target endpoints. Test the connectio n from the replication instance to both source and target endpoints. For instructi ons, see step 2 in the AWS DMS documentation.	DBA

Task	Description	Skills required
Create a replication task.	Create a replication task, and select full load or full load with change data capture (CDC) to migrate data from SQL Server to the S3 bucket. For instructions, see step 3 in the AWS DMS documentation.	DBA
Start the data replication.	Start the replication task, and monitor the logs for any errors.	DBA

Validate the data

Task	Description	Skills required
Validate the migrated data.	On the console, navigate to your target S3 bucket. Open the subfolder that has the same name as the source database. Confirm that the folder contains all the tables that were migrated from the source database.	DBA

Clean up resources

Task	Description	Skills required
Shut down and delete temporary AWS resources.	Shut down temporary AWS resources that you created for the data migration, such as the AWS DMS replication	DBA

Task	Description	Skills required
	instance, and delete them after you validate the export.	

Related resources

- AWS Database Migration Service User Guide
- Using a Microsoft SQL Server database as a source for AWS DMS
- Using Amazon S3 as a target for AWS Database Migration Service
- Using an S3 bucket as an AWS DMS target (AWS re:Post)

Additional information

Use the following code to add an IAM policy with S3 bucket permissions for the AWS DMS role. Replace `bucketname` with the name of your bucket.

```
{
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": [
                 "s3:PutObject",
                 "s3:DeleteObject"
             ],
             "Resource": [
                 "arn:aws:s3:::bucketname*"
             ]
         },
         {
             "Effect": "Allow",
             "Action": [
                 "s3:ListBucket"
             ],
             "Resource": [
                 "arn:aws:s3:::bucketname*"
             ]
         }
     ]
```

```
}
```

Migrate OpenText TeamSite workloads to the AWS Cloud

Battulga Purevragchaa, Amazon Web Services

Michael Stewart and Carlos Marruenda Molina, None

Summary

> Warning: This scenario requires IAM users with programmatic access and long-term credentia ls, which presents a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed. Access keys can be updated if necessary. For more information, see Updating access keys in the *IAM user guide*.

Many OpenText Experience Platform instances are hosted on premises or on traditional hosting solutions with fixed capacity and legacy cost models. Migrating your OpenText Experience Platform workloads to the Amazon Web Services (AWS) Cloud provides additional capabilities and value by increasing your business agility and integration opportunities, in addition to reducing your overall ownership cost.

This pattern provides steps and a template to migrate OpenText TeamSite workloads to the AWS Cloud. The pattern helps you understand how to scope and budget your migration projects by providing a detailed *Epics* section that guides you through an OpenText TeamSite migration process.

This pattern was developed by AWS and TBSCG, an AWS Partner, and accompanies the guide Migrating OpenText TeamSite and Media Management workloads to the AWS Cloud on the AWS Prescriptive Guidance website.

Prerequisites and limitations

Prerequisites

- At least one active AWS account
- An OpenText workload hosted in an on-premises data center or on another cloud provider
- Active OpenText licenses

The migration process also requires the roles and responsibilities that are described in the following table.

Role	Responsibilities
Sponsor	Internal sponsorship
Delivery manager	Migration delivery
Solutions architect	Define the current and new architecture
DevOps engineer	DevOps activities
QA tester	System-level testing
Product owner	Task prioritization based on business requirements
TeamSite authors	Migration user acceptance testing (UAT)
TeamSite administrator	Migration UAT
OpenText lead	OpenText product specialist
OpenText developer	OpenText product specialist
Pricing specialist	AWS and OpenText licensing
IT security	IT security baseline
Third-party integration developer	Rework existing integrations
Front-end developer	Make changes to migrated front-end code
Database administrator	Database configuration

Limitations

- Ensure compatibility with your target operating systems (OSs). You can use the compatibility matrix from the product release notes of the OpenText product version that you are migrating.

Architecture

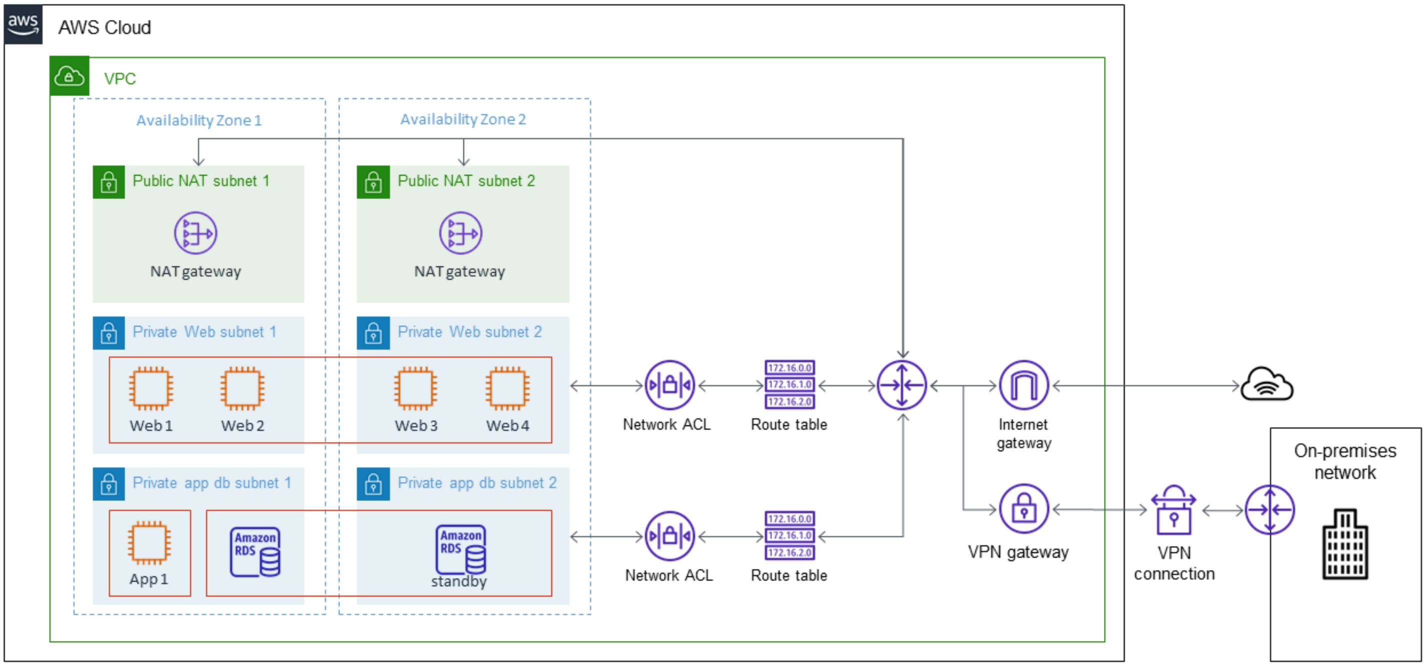
Source technology stack

- OpenText customer experience solutions hosted on premises or on another cloud provider:
  - OpenText TeamSite
  - OpenText LiveSite
  - OpenText Media Management
  - OpenText MediaBin

Target technology stack

- An OpenText Customer Experience platform hosted on the AWS Cloud and that uses the following AWS services:
  - Amazon Elastic Compute Cloud (Amazon EC2)
  - Amazon Elastic Container Service (Amazon ECS)
  - Amazon OpenSearch Service
  - Elastic Load Balancing
  - AWS Lambda
  - Amazon API Gateway
  - Amazon Relational Database Service (Amazon RDS)
  - Amazon Elastic Block Store (Amazon EBS)
  - Amazon Simple Storage Service (Amazon S3)

Target architecture



Tools

- [AWS Database Migration Service (AWS DMS)](https://docs.aws.amazon.com/dms/) is a cloud service that makes it easy to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores.
- [AWS Application Migration Service](https://docs.aws.amazon.com/mgn/) automates the conversion of your source servers to run natively on AWS. It also simplifies application modernization with built-in and custom optimization options.

Epics

Discovery and assessment

Task	Description	Skills required
Hold workshops on discovery requirements.	Hold workshops with business and technical teams to discover the current landscape, gather requirements, and validate the migration strategy.	Sponsor (optional), Delivery manager, Solutions architect , OpenText lead, Product owner

Task	Description	Skills required
	Depending on your migration's complexity and scope, your organization might require several workshops. **Duration**: Two weeks	
Analyze solution and migration requirements.	Analyze and document the business, functional, and technical requirements that influence the design of the planned solution and migration process. **Duration**: One week	Solutions architect, OpenText lead, Product owner
Document your existing OpenText architecture.	Document your existing OpenText architecture, including core components and all related applications and services. **Duration**: One week	Solutions architect, OpenText lead, Product owner
Define the planned AWS architecture.	Define your planned AWS architecture based on the identified components, requirements, and using the OpenText compatibility matrix. You can find the OpenText compatibility matrix in the release notes of your OpenText TeamSite version. **Duration**: One week	Solutions architect, OpenText lead, Product owner, IT security

Task	Description	Skills required
Assess the size of your planned AWS architecture.	Size requirements vary for different architectural components depending on the workload and other non-functional requirements. **Duration**: Two days	Solutions architect, OpenText lead
Calculate the TCO.	Calculate the total cost of ownership (TCO) for your proposed solution. **Duration**: Two days	Solutions architect, Pricing specialist
Define the migration strategy for each component.	Define and document which of the seven common migration strategies (7 Rs) to use for each core or additiona l component that must be migrated to the AWS Cloud. **Duration**: One week	Solutions architect, OpenText lead, Product owner
Define the migration process for the components.	Define the detailed migration process for each of your workload's components. **Duration**: One week	Solutions architect, OpenText lead, Product owner, IT security
Define the global migration process and dependencies.	Create a global migration process and calendar that includes the migration details for components, dependenc ies, and business continuity. **Duration**: Three days	Solutions architect, OpenText lead, Product owner, IT security

Security and compliance activities

Task	Description	Skills required
Create security policies.	Configure the customer managed security policies in your AWS accounts. These should include password complexity and rotation, in addition to automatically turning off unused accounts. For more information about customer managed policies, see Customer managed policies in the AWS Identity and Access Management (IAM) documentation.	Solutions architect
Create IAM users.	Create the IAM users that require access to the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK. For more information about creating IAM users, see Creating an IAM user in your AWS account in the IAM documentation.	Solutions architect
Create IAM groups.	Create the required IAM user groups (for example, administrator or developer groups) and add IAM users to those groups.	Solutions architect

Task	Description	Skills required
	For more information about IAM user groups, see IAM user groups in the IAM documenta tion.	
Attach security policies.	Attach security policies to the IAM groups or roles. For more information about this, see Attaching a policy to an IAM user group in the IAM documentation.	Solutions architect
Turn on detailed billing.	For more information about billing, see Monitoring your usage and costs in the AWS Billing and Cost Management documentation.	Solutions architect
Check the contact details for your accounts.	Make sure that the contact details for your accounts are up to date and map to more than one individual in your organization. For more information, see Managing an AWS account in the AWS Billing and Cost Management documentation.	Solutions architect, Product owner

Task	Description	Skills required
Add security contact information.	Configure your contact information with your security contact information. For more information about this, see Managing an AWS account in the AWS Billing and Cost Management documentation.	Solutions architect, IT security
Set up IAM roles for EC2 instances.	Configure the IAM roles for the EC2 instances. For more information about this, see IAM roles for Amazon EC2 in the Amazon EC2 documentation.	Solutions architect
Configure access to AWS Support.	Attach an IAM policy to IAM users that require access to AWS Support for Support Center and to create support cases. For more information about this, see Access permissions for AWS Support in the AWS Support documentation.	Solutions architect

Task	Description	Skills required
Enable CloudTrail.	Automatically enable AWS CloudTrail in all your AWS Regions. For more information about this, see Using `create-trail` in the AWS CloudTrail documentation.	Solutions architect
Enable CloudTrail log file validation.	Enable the validation of CloudTrail log files. For more information about this, see Enabling log file integrity validation for CloudTrail in the AWS CloudTrail documentation.	Solutions architect
Restrict access to any S3 buckets that contain CloudTrail logs.	Apply a bucket policy restricting access to S3 buckets that contain CloudTrail log files. For more information about this, see Amazon S3 bucket policy for CloudTrail in the AWS CloudTrail documentation.	Solutions architect
Integrate CloudTrail with CloudWatch Logs	Integrate trails generated by CloudTrail with Amazon CloudWatch Logs. For more information about this, see Sending events to CloudWatch Logs in the AWS CloudTrail documentation	Solutions architect

Task	Description	Skills required
Enable AWS Config in all required Regions.	Automatically enable AWS Config in all required Regions. You can set up AWS Config by using AWS CLI. For more information, see Setting Up AWS Config with the AWS CLI in the AWS Config documenta tion.	Solutions architect
Enable logging of S3 bucket access.	Automate S3 bucket access logging with CloudTrail. For more information about this, see Enabling CloudTrail event logging for S3 buckets and objects in the Amazon S3 documentation.	Solutions architect
Configure AWS KMS key policies for CloudTrail.	Automate the configuration of AWS Key Managemen t Service (AWS KMS) key policies for CloudTrail. For more information about this, see Configure AWS KMS key policies for CloudTrai l in the AWS CloudTrail documentation.	Solutions architect

Task	Description	Skills required
Encrypt CloudTrail logs at rest.	Configure server-side encryption of CloudTrail logs using customer managed keys held in AWS KMS. For more information about this, see Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS) in the AWS CloudTrail documentation.	Solutions architect
Automatically rotate KMS keys.	Configure the rotation of AWS KMS keys. For more information about this, see How to enable and disable automatic key rotation in the AWS KMS documentation.	Solutions architect
Configure CloudWatch alarms.	Configure the Amazon CloudWatch alarms that are initiated by specific events. For example, unauthorized requests to APIs or use of the root account. For more information about this, see How to receive notifications when your AWS account’s root access keys are used from the AWS Security Blog.	Solutions architect

Task	Description	Skills required
Configure security groups.	Configure security groups to ensure that unrestricted inbound traffic is not allowed on ports 22 and 3389.	Solutions architect
Turn on VPC flow logging.	Capture rejected IP traffic to and from network interface s in your virtual private cloud (VPC) and configure CloudWatch to capture it. For more information about this, see Creating a flow log in the Amazon VPC documenta tion.	Solutions architect
Modify the default security group to restrict all traffic.	Modify each VPC's default security group so that traffic is denied by default and access is explicitly granted through your security groups. For more information about this, see Security groups for your VPC in the Amazon VPC documentation.	Solutions architect
Configure routing tables between the VPCs.	Configure the routing tables for VPC peering with the least access necessary. For more information about this, see Updating your route tables for a VPC peering connection in the Amazon VPC documentation.	Solutions architect

Setup activities for the new AWS infrastructure

Task	Description	Skills required
Provision the AWS infrastru cture.	Create the AWS accounts and resources. **Duration**: Two weeks	DevOps engineer, Solutions architect
Set up DevOps tools and processes.	Set up DevOps tools and procedures, such as continuou s integration and continuous delivery (CI/CD) pipelines and automated testing framework s.	DevOps engineer, Solutions architect
Automate the migration of core components.	Use existing templates or scripts to automate the installation and configura tion of OpenText products including TeamSite, LiveSite, OpenDeploy and MediaBin. **Duration**: One week	DevOps engineer, Solutions architect, OpenText lead
Automate the migration of additional components.	Analyze and automate the migration of additiona l applications that are integrated with OpenText core components (for example, additional databases, communica tion, monitoring, or cache components). **Duration**: Two weeks	DevOps engineer, Solutions architect, OpenText lead
Adapt core components.	Make any required changes to customizations of OpenText	Solutions architect, OpenText lead, OpenText developer

Task	Description	Skills required
	core components (for example, integrations).	, Third-party integration developer, Front-end developer
Implement and configure additional services.	Provision, configure, and implement any new AWS services, such as AWS Lambda functions or Amazon API Gateway.	DevOps engineer, Solutions architect, Third-party integration developer, Front-end developer
Migrate or refactor other components.	Migrate additional components, including any required refactoring. This includes external applications such as custom-made reporting portals or existing API integration layers.	DevOps engineer, Solutions architect, Third-party integration developer, Front-end developer
Carry out migration in development environment.	Automated migration activities for the development environment, including system provisioning, data migration, application migration, installation, and configuration.	DevOps engineer
Carry out migration in production environment.	Automated migration activities for the production environment, including system provisioning, data migration, application migration, installation, and configuration.	DevOps engineer

Networking activities

Task	Description	Skills required
Define CIDR blocks for each VPC.	Define the Classless Inter-Domain Routing (CIDR) block (the IP range and mask) for each non-default VPC. **Duration**: Less than one week	DevOps engineer, Solutions architect
Define subnets and Availability Zones.	Define the subnets and Availability Zones that are used in each non-default VPC. **Duration**: Less than one week	DevOps engineer, Solutions architect
Define security groups.	Define security groups and security group rules for controlling security on AWS resources. **Duration**: Less than one week	DevOps engineer, Solutions architect
Define network ACLs.	Define the network access control lists (ACLs) to control security at subnet boundaries. **Duration**: Less than one week	DevOps engineer, Solutions architect

Migrate databases

Task	Description	Skills required
Prepare the source databases.	Use AWS DMS to prepare each source database for ongoing replication to the AWS Cloud.	DevOps engineer, Solutions architect

Task	Description	Skills required
Create the databases for the OpenText core components.	Create the databases required by the Opentext TeamSite, LiveSite, and MediaBin components. Make sure that users and access rights are correctly configured according to the OpenText installation documentation.	Solutions architect, OpenText lead, OpenText developer
Copy data from source database servers.	Automate the process of copying data for OpenText core components from the source database server to the target database server.	Solutions architect, OpenText lead, OpenText developer
Synchronize data from the database servers.	Automate the process of performing regular data synchronization from the source databases to the target databases.	OpenText developer

Content migration activities

Task	Description	Skills required
Copy the OpenText TeamSite content stores.	Automate the process of copying the content stores from the source OpenText TeamSite server to the target OpenText TeamSite server.	Solutions architect, OpenText lead, OpenText developer
Map users and groups.	Internal mapping of internal OpenText TeamSite user IDs to target system IDs.	OpenText lead

Task	Description	Skills required
Synchronize the OpenText TeamSite content stores.	Automate the process of performing regular synchroni zing of source and target content stores. This is implemented as part of the migration and QA process.	OpenText developer
Copy data from web servers.	Automate the process of copying data from the source web servers to the target web servers.	Solutions architect, OpenText lead, OpenText developer
Synchronize the web server data.	Automate the process of performing regular synchroni zing of source and target web server data.	OpenText developer
Copy data from web server file system.	Automate the process of copying content and other web assets from the source web server file system to the target web servers.	Solutions architect, OpenText lead, OpenText developer
Synchronize the web server file systems.	Automate the process of performing regular synchroni zing of content and other web assets from the source web server file system to the target web servers.	OpenText developer

Task	Description	Skills required
Generate feeds and indexes.	Automate the process of running any processes that generate feeds or other indexes (for example, web search) that uses OpenText TeamSite or web server content as a data source.	Solutions architect, OpenText lead, OpenText developer
Synchronize the generation of feeds and indexes.	Automate the process of performing regular regenerat ion of feeds and indexes after data synchronizations.	OpenText developer

Testing and QA activities

Task	Description	Skills required
Perform migration QA.	Test the target AWS environment, applicati ons, and services to ensure the automated migration processes are correctly built and configured.	DevOps engineer, OpenText lead, QA tester
Carry out performance testing.	Test the performance in terms of responsiveness and stability under a particula r workload. Investigate, measure, validate, or verify other quality attributes of the destination system, such as scalability and reliability. For this test to be useful, you must have a testing	DevOps engineer, OpenText lead

Task	Description	Skills required
	environment that is the same size as your production environment. **Duration**: Between one and two weeks	
Security testing.	Vulnerability scanning and penetration testing to reveal potential flaws in the security mechanisms of an applicati on that protect data and maintain functionality as required. For this test to be useful, you must have a testing environment that is equivalen t to your production environment in terms of networking and security. **Duration**: Between one and two weeks	DevOps engineer, OpenText lead

Operational integration activities

Task	Description	Skills required
Check operational readiness.	Understand how you currently perform IT operations and how you will operate in the AWS Cloud. You can achieve this business outcome by defining a cloud operating model.	DevOps engineer, OpenText lead, Service delivery manager

Task	Description	Skills required
	Duration: One week	
Invest in operations automation.	Invest in automation to deliver an AWS operating model.	DevOps engineer, OpenText lead, Service delivery manager
Integrate operations.	Continue using current IT tools and extend them through integration to the AWS Cloud.	DevOps engineer, OpenText lead, Service delivery manager

Cutover activities

Task	Description	Skills required
Switch DNS.	Manually switch the domain name system (DNS) from existing hosts to hosts based in the AWS Cloud. **Duration**: One hour	DevOps engineer, OpenText lead
Test disaster recovery.	Test disaster recovery, backup restore, and run your automated tests. **Duration**: One day	DevOps engineer, OpenText lead, QA tester
Validate monitoring and analytics.	Validate that the monitoring and analytics are working. **Duration**: Two hours	DevOps engineer, OpenText lead
Turn off old environment and request the server’s shutdown.	**Duration**: Three days	DevOps engineer, OpenText lead

Related resources

- Customer managed policies
- Creating an IAM user in your AWS account
- IAM user groups
- Attaching a policy to an IAM user group
- Monitoring your usage and cost
- Managing an AWS account
- IAM roles for Amazon EC2
- Access permissions for AWS Support
- Using create-trail
- Enabling log file integrity validation for CloudTrail
- Amazon S3 bucket policy for CloudTrail
- Sending events to CloudWatch Logs
- Setting Up AWS Config with the AWS CLI
- Enabling CloudTrail event logging for S3 buckets and objects
- Configure AWS KMS key policies for CloudTrail
- Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS)
- How to enable and disable automatic key rotation
- How to receive notifications when your AWS account's root access keys are used
- Creating a flow log
- Security groups for your VPC
- Updating your route tables for a VPC peering connection

Migrate Oracle CLOB values to individual rows in PostgreSQL on AWS

Sai Krishna Namburu and Sindhusha Paturu, Amazon Web Services

Summary

This pattern describes how to split Oracle character large object (CLOB) values into individual rows in Amazon Aurora PostgreSQL-Compatible Edition and Amazon Relational Database Service (Amazon RDS) for PostgreSQL. PostgreSQL doesn't support the CLOB data type.

Tables with interval partitions are identified in the source Oracle database, and the table name, the type of partition, the interval of the partition, and other metadata are captured and loaded into the target database. You can load CLOB data that is less than 1 GB in size into target tables as text by using AWS Database Migration Service (AWS DMS), or you can export the data in CSV format, load it into an Amazon Simple Storage Service (Amazon S3) bucket, and migrate it to your target PostgreSQL database.

After migration, you can use the custom PostgreSQL code that is provided with this pattern to split the CLOB data into individual rows based on the new line character identifier (`CHR(10)`) and populate the target table.

Prerequisites and limitations

Prerequisites

- An Oracle database table that has interval partitions and records with a CLOB data type.
- An Aurora PostgreSQL-Compatible or Amazon RDS for PostgreSQL database that has a table structure that's similar to the source table (same columns and data types).

Limitations

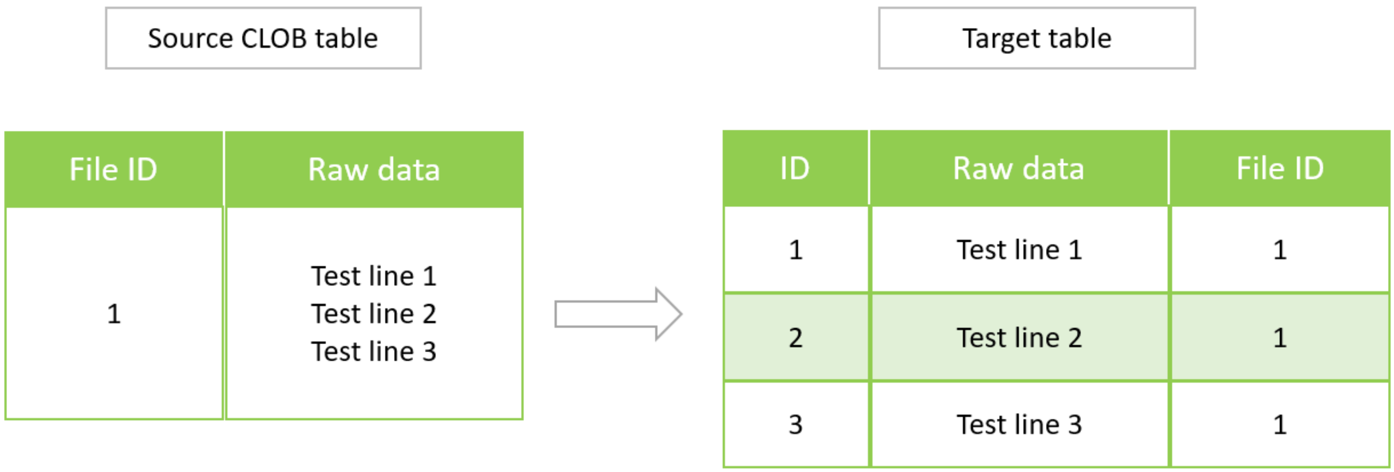
- The CLOB value cannot exceed 1 GB.
- Each row in the target table must have a new line character identifier.

Product versions

- Oracle 12c
- Aurora Postgres 11.6

Architecture

The following diagram shows a source Oracle table with CLOB data, and the equivalent PostgreSQL table in Aurora PostgreSQL-Compatible version 11.6.



Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- Amazon Relational Database Service (Amazon RDS) for PostgreSQL helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.
- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

You can use the following client tools to connect to, access, and manage your Aurora PostgreSQL-Compatible and Amazon RDS for PostgreSQL databases. (These tools aren't used within this pattern.)

- pgAdmin is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.

- DBeaver is an open-source database tool for developers and database administrators. You can use the tool to manipulate, monitor, analyze, administer, and migrate your data.

Best practices

For best practices for migrating your database from Oracle to PostgreSQL, see the AWS blog post Best practices for migrating an Oracle database to Amazon RDS PostgreSQL or Amazon Aurora PostgreSQL: Migration process and infrastructure considerations.

For best practices for configuring the AWS DMS task for migrating large binary objects, see Migrating large binary objects (LOBs) in the AWS DMS documentation.

Epics

Identify the CLOB data

Task	Description	Skills required
Analyze the CLOB data.	In the source Oracle database, analyze the CLOB data to see whether it contains column headers, so you can determine the method for loading the data into the target table. To analyze the input data, use the following query. `SELECT * FROM clobdata_or;`	Developer
Load the CLOB data to the target database.	Migrate the table that has CLOB data to an interim (staging) table in the Aurora or Amazon RDS target database. You can use AWS DMS or upload the data as	Migration engineer, DBA

Task	Description	Skills required
	a CSV file to an Amazon S3 bucket. For information about using AWS DMS for this task, see [Using an Oracle database as a source]() and [Using a PostgreSQL database as a target]() in the AWS DMS documentation. For information about using Amazon S3 for this task, see [Using Amazon S3 as a target]() in the AWS DMS documentation.	

Validate the target PostgreSQL table.	Validate the target data, including headers, against the source data by using the following queries in the target database.
	
	`SELECT * FROM clobdata_pg;`
	`SELECT * FROM clobdatatarget;`
	
	Compare the results against the query results from the source database (from the first step).
		Developer

Task	Description	Skills required
Split the CLOB data into separate rows.	Run the custom PostgreSQL code provided in the Additional information section to split the CLOB data and insert it into separate rows in the target PostgreSQL table.	Developer

Validate the data.

Task	Description	Skills required

Validate the data in the target table.	Validate the data inserted into the target table by using the following queries.
	`SELECT * FROM clobdata_pg;`
	`SELECT * FROM clobdatatarget;`
		Developer

Related resources

- CLOB data type (Oracle documentation)
- Data types (PostgreSQL documentation)

Additional information

PostgreSQL function for splitting CLOB data

```
do
$$
declare
totalstr varchar;
```

```
str1 varchar;
str2 varchar;
pos1 integer := 1;
pos2 integer ;
len integer;


begin
        select rawdata||chr(10) into totalstr from clobdata_pg;
        len :=  length(totalstr) ;
        raise notice 'Total length : %',len;
        raise notice 'totalstr : %',totalstr;
        raise notice 'Before while loop';


        while pos1 < len  loop


                  select position (chr(10) in totalstr) into pos2;
                 raise notice '1st position of new line : %',pos2;


                str1 := substring (totalstr,pos1,pos2-1);
                raise notice 'str1 : %',str1;


                   insert into clobdatatarget(data) values (str1);
                   totalstr := substring(totalstr,pos2+1,len);
                   raise notice 'new totalstr :%',totalstr;
                len :=  length(totalstr) ;


        end loop;
end
$$
LANGUAGE 'plpgsql' ;
```

Input and output examples

You can use the following examples to try out the PostgreSQL code before you migrate your data.

Create an Oracle database with three input lines.

```
CREATE TABLE clobdata_or (
```

```
id INTEGER GENERATED ALWAYS AS IDENTITY,
rawdata clob  );


insert into clobdata_or(rawdata) values (to_clob('test line 1') || chr(10) ||
 to_clob('test line 2') || chr(10) || to_clob('test line 3') || chr(10));
COMMIT;


SELECT * FROM clobdata_or;
```

This displays the following output.

id	rawdata
1	test line 1 test line 2 test line 3

Load the source data into a PostgreSQL staging table (`clobdata_pg`) for processing.

```
SELECT * FROM clobdata_pg;

CREATE TEMP TABLE clobdatatarget (id1 SERIAL,data VARCHAR );

<Run the code in the additional information section.>

SELECT * FROM clobdatatarget;
```

This displays the following output.

id1	data
1	test line 1
2	test line 2
3	test line 3

Migrate an on-premises Oracle database to Amazon RDS for Oracle by using direct Oracle Data Pump Import over a database link

Rizwan Wangde, Amazon Web Services

Summary

Numerous patterns cover migrating on-premises Oracle databases to Amazon Relational Database Service (Amazon RDS) for Oracle by using Oracle Data Pump, a native Oracle utility that is the preferred way to migrate large Oracle workloads. These patterns typically involve exporting application schemas or tables into dump files, transferring the dump files to a database directory on Amazon RDS for Oracle, and then importing the application schemas and data from the dump files.

Using that approach, a migration can take longer depending on the size of the data and the time that it takes to transfer the dump files to the Amazon RDS instance. In addition, the dump files reside on the Amazon RDS instance's Amazon Elastic Block Store (Amazon EBS) volume, which must be large enough for the database and the dump files. When the dump files are deleted after import, the empty space cannot be retrieved, so you continue to pay for unused space.

This pattern mitigates those issues by performing a direct import on the Amazon RDS instance by using the Oracle Data Pump API (`DBMS_DATAPUMP`) over a database link. The pattern initiates a simultaneous export and import pipeline between the source and target databases. This pattern doesn't require sizing an EBS volume for the dump files because no dump files are created or stored on the volume. This approach saves the monthly cost of unused disk space.

Prerequisites and limitations

Prerequisites

- An active Amazon Web Services (AWS) account.
- A virtual private cloud (VPC) configured with private subnets across at least two Availability Zones, to provide the network infrastructure for the Amazon RDS instance.
- An Oracle database in an on-premises data center, or self-managed on Amazon Elastic Compute Cloud (Amazon EC2).
- An existing Amazon RDS for Oracle instance in a single Availability Zone. Using a single Availability Zone improves write performance during migration. A Multi-AZ deployment can be enabled 24–48 hours before cutover.

This solution can also use Amazon RDS Custom for Oracle as a target.

- AWS Direct Connect (recommended for large sized databases).
- Network connectivity and firewall rules on premises configured to allow an inbound connection from the Amazon RDS instance to the on-premises Oracle database.

Limitations

- The database size limit on Amazon RDS for Oracle is 64 tebibytes (TiB) as of December 2022.
- The maximum size of a single file on an Amazon RDS for Oracle DB instances is 16 TiB. This is important to know because you might need to spread tables across multiple tablespaces.

Product versions

- Source database: Oracle Database version 10g Release 1 and later.
- Target database: For the latest list of supported versions and editions on Amazon RDS, see Amazon RDS for Oracle in the AWS documentation.

Architecture

Source technology stack

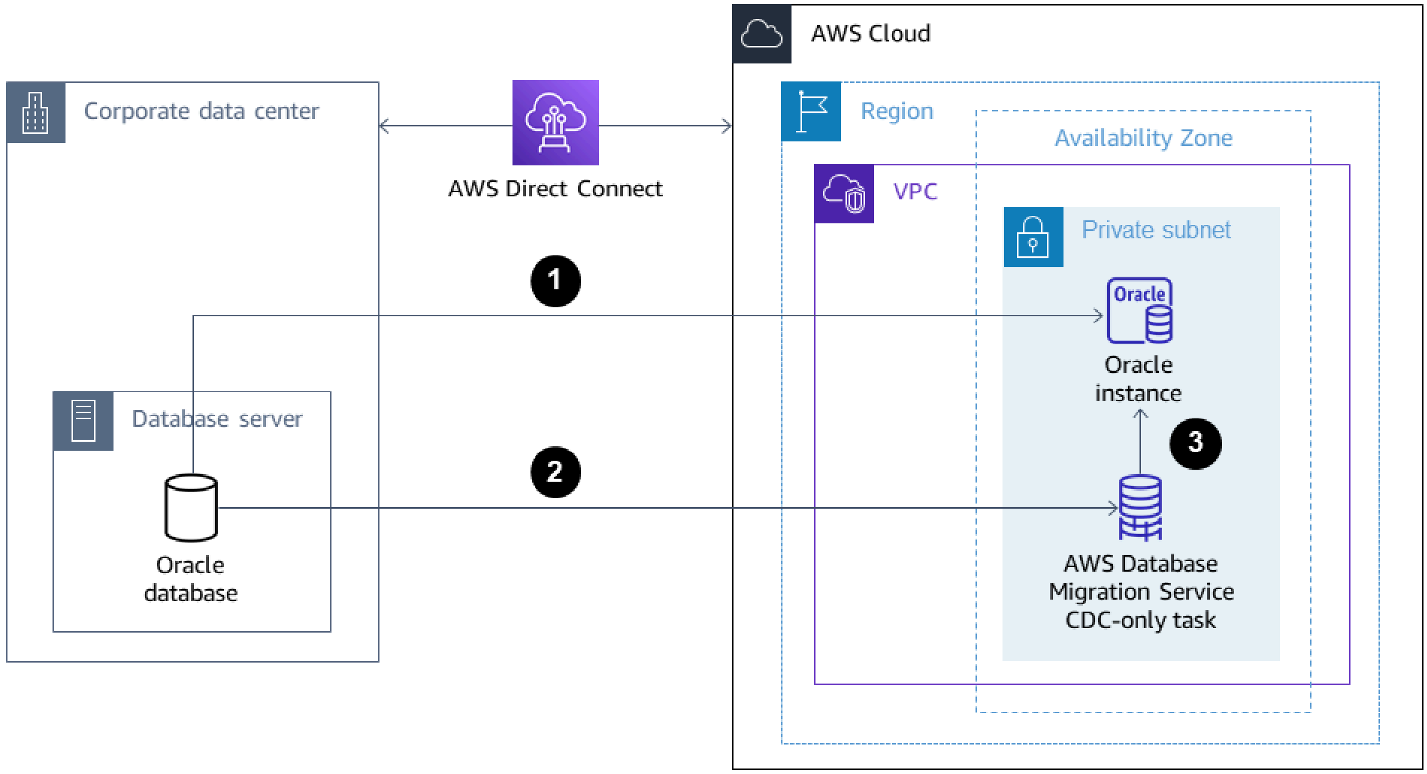
- Self-managed Oracle database on premises or in the cloud

Target technology stack

- Amazon RDS for Oracle or Amazon RDS Custom for Oracle

Target architecture

The following diagram shows the architecture for migrating from an on-premises Oracle database to Amazon RDS for Oracle in a Single-AZ environment. The arrow directions depict the data flow in the architecture. The diagram doesn't show what component is initiating the connection.



1. The Amazon RDS for Oracle instance connects to the on-premises source Oracle database to perform a full-load migration over the database link.
2. AWS Database Migration Service (AWS DMS) connects to the on-premises source Oracle database to perform ongoing replication by using change data capture (CDC).
3. CDC changes are applied to the Amazon RDS for Oracle database.

Tools

AWS services

- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups. This pattern uses CDC and the **Replicate data changes only** setting.
- AWS Direct Connect links your internal network to a AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- Amazon Relational Database Service helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.

Other tools

- Oracle Data Pump helps you move data and metadata from one database to another at high speeds.
- Client tools such as Oracle Instant Client or SQL Developer are used to connect and run SQL queries on the database.

Best practices

Although AWS Direct Connect uses dedicated, private network connections between the on-premises network and AWS, consider the following options for additional security and data encryption for data in transit:

- A virtual private network (VPN) using AWS Site-to-Site VPN or an IPsec VPN connection from the on-premises network to the AWS network
- Oracle Database Native Network Encryption configured on the on-premises Oracle database
- Encryption using TLS

Epics

Prepare the on-premises source Oracle database

Task	Description	Skills required
Set up network connectivity from the target database to the source database.	Configure the on-premises network and firewall to allow incoming connection from the target Amazon RDS instance to the on-premises source Oracle database.	Network administrator, Security engineer
Create a database user with the appropriate privileges.	Create a database user in the on-premises source Oracle database with privileges to migrate data between the source and target using Oracle Data Pump:	DBA

Task	Description	Skills required
	```GRANT CONNECT to <migration_user>; GRANT DATAPUMP_EXP_FULL_DATABASE to <migration_user>; GRANT SELECT ANY TABLE to <migration_user>;```	

Task	Description	Skills required

Prepare the on-premises source database for AWS DMS CDC migration.	(Optional) Prepare the on-premises source Oracle database for AWS DMS CDC migration after completion of Oracle Data Pump Full Load:
	
	1. Configure the additional privileges required to manage FLASHBACK during Oracle Data Pump migration:
	`GRANT FLASHBACK ANY TABLE to <migration_user>;`
	`GRANT FLASHBACK ARCHIVE ADMINISTER to <migration_user>;`
	2. To configure user account privileges required on a self-managed Oracle source for AWS DMS, see the [AWS DMS documentation](#).
	3. To prepare an Oracle self-managed source database for CDC using AWS DMS, see the [AWS DMS documentation](#).
		DBA

Install and configure SQL Developer.	Install and configure [SQL Developer](#) to connect and run SQL queries on the source and target databases.	DBA, Migration engineer

Task	Description	Skills required
Generate a script to create the tablespaces.	Use the following example SQL query to generate the script on the source database: `SELECT 'CREATE TABLESPACE ' tablespace_name ' DATAFILE SIZE 1G AUTOEXTEND ON MAXSIZE UNLIMITED;' from dba_tablespaces where tablespace_name not in ('SYSTEM', 'SYSAUX','TEMP','UNDOTBS1') order by 1;` The script will be applied on the target database.	DBA
Generate a script to create users, profiles, roles, and privileges.	To generate a script to create the database users, profiles, roles, and privileges, use the scripts from the Oracle Support document [How to Extract DDL for User including Privileges and Roles Using dbms_metadata.get_ddl (Doc ID 2739952.1)]() (Oracle account required). The script will be applied on the target database.	DBA

Prepare the target Amazon RDS for Oracle instance

Task	Description	Skills required

Create a database link to the source database and verify connectivity.	To create a database link to the on-premises source database, you can use the following example command:
	`CREATE DATABASE LINK link2src CONNECT TO <migration_user_account> IDENTIFIED BY <password> USING '(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<dns or ip address of remote db>) (PORT=<listener port>))(CONNECT_DATA=(SID=<remote SID>)))';`
	To verify connectivity, run the following SQL command:
	`select * from dual@link2src;`
	Connectivity is successful if the response is X.
		DBA

Run the scripts to prepare the target instance.	Run the previously generated scripts to prepare the target Amazon RDS for Oracle instance: 1. Tablespaces	DBA, Migration engineer

Task	Description	Skills required
	2. Profiles 3. Roles This helps ensure that the Oracle Data Pump migration can create the schemas and their objects.	

Perform a full-load migration by using Oracle Data Pump Import over a database link

Task	Description	Skills required
Migrate the required schemas.	To migrate the required schemas from the source on-premises database to the target Amazon RDS instance, use the code in the Additional information section: • To migrate a single schema, run *Code 1* from the Additional information section. • To migrate multiple schemas, run *Code 2* from the Additional information section. To tune the performance of the migration, you can adjust the number of parallel processes by running the following command:	DBA

Task	Description	Skills required
	`DBMS_DATAPUMP.SET_ PARALLEL (handle => v_hdnl, degree => 4);`	
Gather schema statistics to improve performance.	The Gather Schema Statistics command returns the Oracle query optimizer statistic s gathered for database objects. By using this information, the optimizer can select the best execution plan for any query against these objects: `EXECUTE DBMS_STAT S.GATHER_SCHEMA_ST ATS(ownname => '<schema_name>');`	DBA

Perform a full-load migration and CDC replication by using Oracle Data Pump and AWS DMS

Task	Description	Skills required
Capture the SCN on the source on-premises Oracle database.	Capture the system change number (SCN) on the source on-premises Oracle database. You will use the SCN for full-load import and as the starting point for CDC replication. To generate the current SCN on the source database, run the following SQL statement:	DBA

Task	Description	Skills required
	`SELECT current_scn FROM V$DATABASE;`	

Task	Description	Skills required
Perform the full-load migration of the schemas.	To migrate the required schemas (`FULL LOAD`) from the source on-premises database to the target Amazon RDS instance, do the following: • To migrate a single schema, run *Code 3* from the [Additional information](#) section. • To migrate multiple schemas, run *Code 4* from the [Additional information](#) section. In the code, replace `<CURRENT_SCN_VALUE_IN_SOURCE_DATABASE>` with the SCN that you captured from the source database: `DBMS_DATAPUMP.SET_PARAMETER (handle => v_hdnl, name => 'FLASHBACK_SCN', value => <CURRENT_SCN_VALUE_IN_SOURCE_DATABASE>);` To tune the performance of the migration, you can adjust the number of parallel processes:	DBA

Task	Description	Skills required
	`DBMS_DATAPUMP.SET_ PARALLEL (handle => v_hdnl, degree => 4);`	
Disable the triggers under the migrated schemas.	Before you begin the AWS DMS CDC-only task, disable the `TRIGGERS` under the migrated schemas.	DBA
Gather schema statistics to improve performance.	The Gather Schema Statistics command returns the Oracle query optimizer statistic s gathered for database objects: `EXECUTE DBMS_STAT S.GATHER_SCHEMA_ST ATS(ownname => '<schema_name>');` By using this information, the optimizer can select the best execution plan for any query against these objects.	DBA

Task	Description	Skills required
Use AWS DMS to perform an ongoing replication from the source to target.	Use AWS DMS to perform an ongoing replication from the source Oracle database to the target Amazon RDS for Oracle instance. For more information, see Creating tasks for ongoing replication using AWS DMS and the blog post How to work with native CDC support in AWS DMS.	DBA, Migration engineer

Cut over to Amazon RDS for Oracle

Task	Description	Skills required
Enable Multi-AZ on the instance 48 hours before cutover.	If this is a production instance, we recommend enabling Multi-AZ deploymen t on the Amazon RDS instance to provide the benefits of high availability (HA) and disaster recovery (DR).	DBA, Migration engineer
Stop the AWS DMS CDC-only task (if CDC was turned on).	1. Ensure the source latency and target latency on the AWS DMS task's Amazon CloudWatch metrics show 0 seconds. 2. Stop the AWS DMS CDC-only task.	DBA

Task	Description	Skills required
Enable the triggers.	Enable the TRIGGERS that you disabled before the CDC task was created.	DBA

Related resources

AWS

- Preparing an Oracle self-managed source database for CDC using AWS DMS
- Creating tasks for ongoing replication using AWS DMS
- Multi-AZ deployments for high availability
- How to work with native CDC support in AWS DMS (blog post)

Oracle documentation

- DBMS_DATAPUMP

Additional information

Code 1: Full-load migration only, single application schema

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE( handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''<schema_name>'')');  --
 To migrate one selected schema
    DBMS_DATAPUMP.METADATA_FILTER (hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')'); --
 To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
```

```
/
```

Code 2: Full-load migration only, multiple application schemas

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE( handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'SCHEMA_LIST',
 '''<SCHEMA_1>'',''<SCHEMA_2>'', ''<SCHEMA_3>'''); -- To migrate multiple schemas
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')');
 -- To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

Code 3: Full-load migration before CDC-only task, single application schema

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN(operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE( handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER(v_hdnl,'SCHEMA_EXPR','IN (''<schema_name>'')');  --
 To migrate one selected schema
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')');
 -- To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARAMETER (handle => v_hdnl, name => 'FLASHBACK_SCN', value =>
 <CURRENT_SCN_VALUE_IN_SOURCE_DATABASE>); -- SCN required for AWS DMS CDC only task.
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

Code 4: Full-load migration before CDC-only task, multiple application schemas

```
DECLARE
    v_hdnl NUMBER;
BEGIN
    v_hdnl := DBMS_DATAPUMP.OPEN (operation => 'IMPORT', job_mode  => 'SCHEMA',
 remote_link => '<DB LINK Name to Source Database>', job_name  => null);
    DBMS_DATAPUMP.ADD_FILE (handle => v_hdnl, filename  => 'import_01.log',  directory
 => 'DATA_PUMP_DIR', filetype  => dbms_datapump.ku$_file_type_log_file);
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'SCHEMA_LIST',
 '''<SCHEMA_1>'',''<SCHEMA_2>'', ''<SCHEMA_3>'''); -- To migrate multiple schemas
    DBMS_DATAPUMP.METADATA_FILTER (v_hdnl, 'EXCLUDE_PATH_EXPR','IN (''STATISTICS'')');
 -- To prevent gathering Statistics during the import
    DBMS_DATAPUMP.SET_PARAMETER (handle => v_hdnl, name => 'FLASHBACK_SCN', value =>
 <CURRENT_SCN_VALUE_IN_SOURCE_DATABASE>); -- SCN required for AWS DMS CDC only task.
    DBMS_DATAPUMP.SET_PARALLEL (handle => v_hdnl, degree => 4);  -- Number of parallel
 processes performing export and import
    DBMS_DATAPUMP.START_JOB(v_hdnl);
END;
/
```

Scenario where a mixed migration approach can work better

In rare scenarios where the source database contains tables with millions of rows and very large-sized LOBSEGMENT columns, this pattern will slow down the migration. Oracle migrates LOBSEGMENTs over the network link one at a time. It extracts a single row (along with the LOB column data) from the source table, and inserts the row into the target table, repeating the process until all rows are migrated. Oracle Data Pump over the database link doesn't support bulk load or direct path load mechanisms for LOBSEGMENTs.

In this situation, we recommend the following:

- Skip the identified tables during the Oracle Data Pump migration by adding the following metadata filter:

```
dbms_datapump.metadata_filter(handle =>h1, name=>'NAME_EXPR', value => 'NOT IN
 (''TABLE_1'',''TABLE_2'')');
```

- Use an AWS DMS task (full-load migration, with CDC replication if required) to migrate the identified tables. AWS DMS will extract multiple rows from the source Oracle database, and insert them in a batch to the target Amazon RDS instance, which improves performance.

Migrate Oracle PeopleSoft to Amazon RDS Custom

Gaurav Gupta, Amazon Web Services

Summary

Oracle PeopleSoft is an enterprise resource planning (ERP) solution for enterprise-wide processes. PeopleSoft has a three-tier architecture: client, application, and database. PeopleSoft can be run on Amazon Relational Database Service (Amazon RDS). Now, you can also run PeopleSoft on Amazon RDS Custom, which provides access to the underlying operating system.

Amazon RDS Custom for Oracle is a managed database service for legacy, custom, and packaged applications that require access to the underlying operating system and database environment. When you migrate your Oracle database to Amazon RDS Custom, Amazon Web Services (AWS) can manage backup tasks and high availability, while you can focus on maintaining your PeopleSoft application and functionality. For key factors to consider for a migration, see Oracle database migration strategies in AWS Prescriptive Guidance.

This pattern focuses on the steps to migrate a PeopleSoft database on Amazon Elastic Compute Cloud (Amazon EC2) to Amazon RDS Custom by using an Oracle Recovery Manager (RMAN) backup. It uses an Amazon Elastic File System (Amazon EFS) shared file system between the EC2 instance and Amazon RDS Custom, although you can also use Amazon FSx or any shared drive. The pattern uses an RMAN full backup (sometimes referred to as a level 0 backup).

Prerequisites and limitations

Prerequisites

- An Oracle version 19C source database that is running on Amazon EC2 with Oracle Linux 7, Oracle Linux 8, Red Hat Enterprise Linux (RHEL) 7, or RHEL 8. In the examples for this pattern, the source database name is `FSDM092`, but this isn't a requirement.

> **Note**
>
> You can also use this pattern with on-premises Oracle source databases. You must have the appropriate network connectivity between the on-premises network and a virtual private cloud (VPC).

- A PeopleSoft 9.2 demo instance.

- A single PeopleSoft application tier. However, you can adapt this pattern to work with multiple application tiers.
- Amazon RDS Custom configured with at least 8 GB of swap space.

Limitations

This pattern doesn't support the following configurations:

- Setting the database `ARCHIVE_LAG_TARGET` parameter to a value outside the 60–7200 range
- Disabling the DB instance log mode (`NOARCHIVELOG`)
- Turning off the Amazon Elastic Block Store (Amazon EBS) optimized attribute of the EC2 instance
- Modifying the original EBS volumes attached to the EC2 instance
- Adding new EBS volumes or changing the volume type from gp2 to gp3
- Changing the extension format for the `LOG_ARCHIVE_FORMAT` parameter (requires `*.arc`)
- Multiplexing or changing the control file location and name (it has to be `/rdsdbdata/db/*DBNAME*/controlfile/control-01.ctl`)

For additional information about these and other unsupported configurations, see the Amazon RDS documentation.

Product versions

For Oracle Database versions and instance classes supported by Amazon RDS Custom, see Requirements and limitations for Amazon RDS Custom for Oracle.

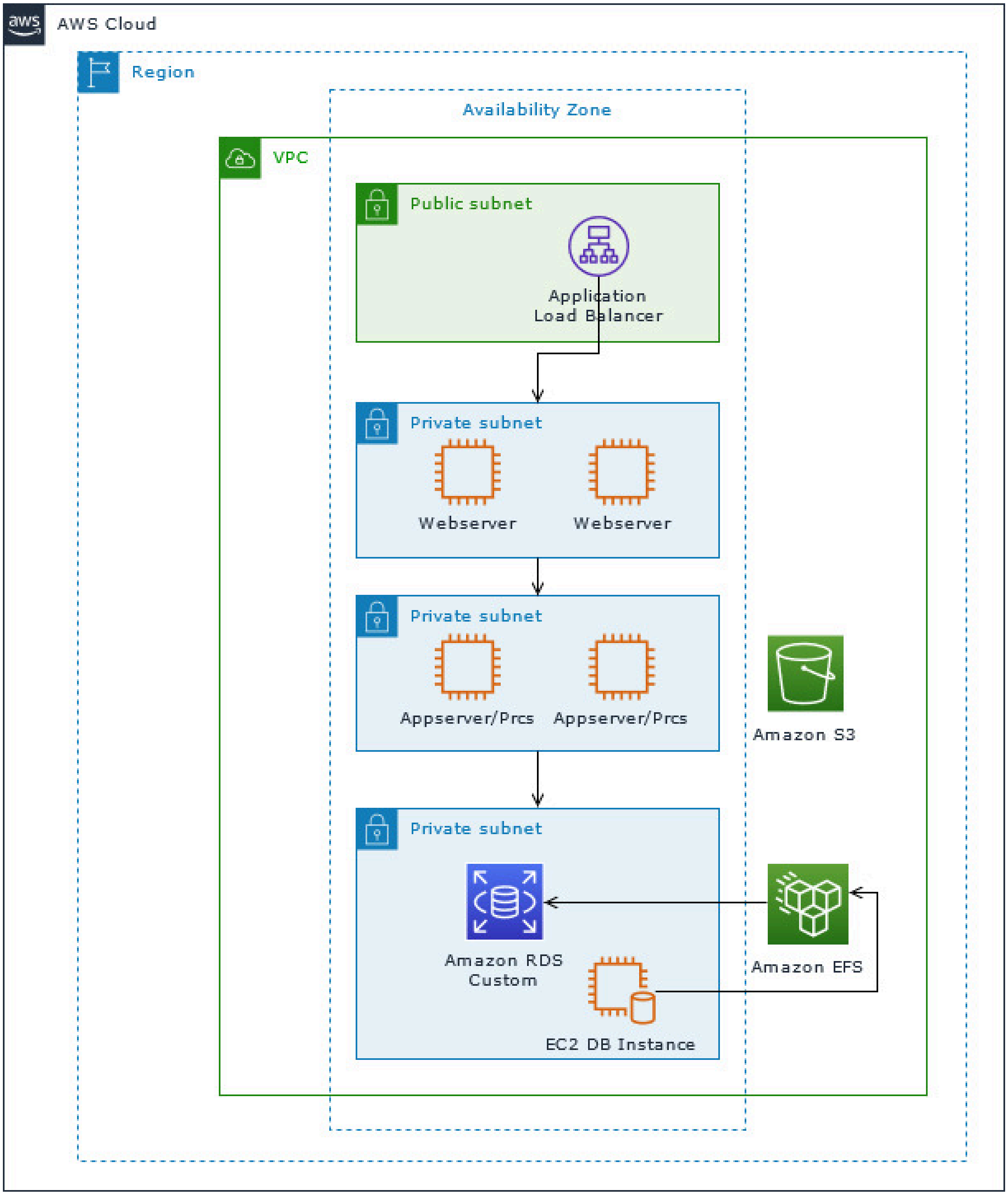
Architecture

Target technology stack

- Application Load Balancer
- Amazon EFS
- Amazon RDS Custom for Oracle
- AWS Secrets Manager
- Amazon Simple Storage Service (Amazon S3)

Target architecture

The following architecture diagram represents a PeopleSoft system running in a single Availability Zone on AWS. The application tier is accessed through an Application Load Balancer. Both the application and the databases are in private subnets, and the Amazon RDS Custom and Amazon EC2 database instance use an Amazon EFS shared file system to store and access the RMAN backup files. Amazon S3 is used for creating the custom RDS Oracle engine and for storing the redo logs metadata.



Tools

Tools

AWS services

- Amazon RDS Custom for Oracle is a managed database service for legacy, custom, and packaged applications that require access to the underlying operating system and database environment. It automates database administration tasks, such as backups and high availability.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud. This pattern uses an Amazon EFS shared file system to store and access the RMAN backup files.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. In this pattern, you retrieve the database user passwords from Secrets Manager to create the `RDSADMIN` and `ADMIN` users and to change the `sys` and `system` passwords.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones. This pattern uses an Application Load Balancer.

Other tools

- Oracle Recovery Manager (RMAN) provides backup and recovery support for Oracle databases. This pattern uses RMAN to perform a hot backup of the source Oracle database on Amazon EC2 that is restored on Amazon RDS Custom.

Best practices

- For database initialization parameters, customize the standard pfile that's provided by the Amazon RDS Custom DB instance for PeopleSoft instead of using the spfile from the Oracle source database. This is because white spaces and comments cause issues when creating read replicas in Amazon RDS Custom. For more information about database initialization parameters, see Oracle Support Note 1100831.1 (requires an Oracle Support account).

- Amazon RDS Custom uses Oracle automatic memory management by default. If you want to use the Hugemem kernel, you can configure Amazon RDS Custom to use automatic shared memory management instead.
- Leave the `memory_max_target` parameter enabled by default. The framework uses this in the background to create read replicas.
- Enable Oracle Flashback Database. This feature is useful when reinstating the standby in failover (not switchover) testing scenarios.

Epics

Set up the DB instance and file system

Task	Description	Skills required
Create the DB instance.	In the Amazon RDS console, create an Amazon RDS Custom for Oracle DB instance with a DB name called FSDMO92 (or your source database name). For instructions, see [Working with Amazon RDS Custom](#) in the AWS documentation and the [Amazon RDS Custom for Oracle – New Control Capabilities in Database Environment](#) blog post. This ensures that the database name is set to the same name as the source database. (If kept blank, the EC2 instance and database name will be set to `ORCL`.)	DBA

Perform an RMAN full backup of the source Amazon EC2 database

Task	Description	Skills required
Create a backup script.	Create an RMAN backup script to back up the database to the Amazon EFS file system that you mounted (`/efs` in the following example). You can use the example code or run one of your existing RMAN scripts. ```#!/bin/bash Dt=`date +'%Y%m%d-%H%M'` BACKUP_LOG="rman-${ORACLE_SID}-$Dt" export TAGDATE=`date +%Y%m%d%H%M`; LOGPATH=/u01/scripts/logs rman target / >> $LOGPATH/rman-${ORACLE_SID}-$Dt << EOF SQL "ALTER SYSTEM SWITCH LOGFILE"; SQL "ALTER SESSION SET NLS_DATE_FORMAT="DD.MM.YYYY HH24:MI:SS""; RUN { ALLOCATE CHANNEL ch11 TYPE DISK MAXPIECESIZE 5G; ALLOCATE CHANNEL ch12 TYPE DISK MAXPIECESIZE 5G; BACKUP AS COMPRESSED BACKUPSET FULL DATABASE```	DBA

Task	Description	Skills required

	``
	 FORMAT '/efs/rma
	n_backup/FSCM/%d_%T_
	%s_%p_FULL' ;
	  SQL "ALTER SYSTEM
	 ARCHIVE LOG CURRENT";
	  BACKUP FORMAT '/efs/
	rman_backup/FSCM/%d_
	%T_%s_%p_ARCHIVE'
	 ARCHIVELOG ALL DELETE
	 ALL INPUT ;
	  BACKUP CURRENT
	 CONTROLFILE FORMAT '/
	efs/rman_backup/FSCM/
	%d_%T_%s_%p_CONTROL';
	}
	EXIT;
	EOF
	``	

Run the backup script.	To run the RMAN backup script, log in as the Oracle Home User, and run the script. ``` $ chmod a+x rman_back up.sh $ ./rman_backup.sh & ```	DBA

Task	Description	Skills required

Check for errors and note the name of the backup file.	Check the RMAN log file for errors. If everything looks fine, list the backup of the control file by running the following command.
	``RMAN> list backup of controlfile;
	
	using target database control file instead of recovery catalog```
	Note the name of the output file.
	``List of Backup Sets
	===================
	
	BS Key Type LV Size Device Type Elapsed Time Completion Time
	------- ---- -- ---------- ----------- ------------ ---------------
	12 Full 21.58M DISK 00:00:01 13-JUL-22
	        BP Key: 12 Status: AVAILABLE Compressed: NO Tag: TAG20220713T150155
	        Piece Name: /efs/rman_backup/FSCM/FSDMO92_20220713_12_1_CONTROL
	  Control File Included: Ckp SCN: 165591599```
		DBA

Task	Description	Skills required
	`85898 Ckp time: 13-JUL-22` You will use the backup control file `/efs/rman_backup/FSCM/FSDMO92_20220713_12_1_CONTROL` when you restore the database on Amazon RDS Custom.	

Shut down the source application tier

Task	Description	Skills required

Shut down the application.	To shut down the source application tier, use the `psadmin` utility or the `psadmin` command line utility.
	
	1. To shut down the webserver, run the following command.
	`psadmin -w shutdown -d "webserver domain name"`
	2. To shut down the application server, run the following command.
	`psadmin -c shutdown -d "application server domain name"`
		DBA, PeopleSoft Administrator

Task	Description	Skills required
	3. To shut down the process scheduler, run the following command. `psadmin -p stop -d "process scheduler domain name"`	

Configure the target Amazon RDS Custom database

Task	Description	Skills required
Install the nfs-utils rpm package.	To install the `nfs-utils rpm` package, run the following command. `$ yum install -y nfs-utils`	DBA
Mount the EFS storage.	Get the Amazon EFS mount command from the Amazon EFS console page. Mount the EFS file system on the Amazon RDS instance by using a Network File System (NFS) client. `sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-xxxxxxxxxx.efs.eu-west-1.amazonaws.com:/ /efs`	DBA

Task	Description	Skills required
	```sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-xxxxxxxxxx.efs.eu-west-1.amazonaws.com:/ /efs```	

Drop the starter database and create the directories to store the database files

Task	Description	Skills required
Pause automation mode.	You have to pause automation mode on your Amazon RDS Custom DB instance before you proceed with the next steps, to make sure that automation doesn't interfere with the RMAN restore activity. You can pause the automation by using the AWS console or the AWS Command Line Interface (AWS CLI) command (make sure that you have configured the AWS CLI first). ```aws rds modify-db-instance \ --db-instance-identifier peoplesoft-fscm-92 \ --automation-mode all-paused \```	DBA

Task	Description	Skills required
	``` --resume-full-au tomation-mode-minute 360 \ --region eu-west-1 ``` When you specify the duration of the pause, make sure that you leave enough time for the RMAN restore. This depends on the size of the source database, so modify the 360 value accordingly. Also, make sure that the total time of the paused automation does not overlap with the backup or maintenance window of the database.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create and modify the parameter file for PeopleSoft</td><td>To create and modify the pfile for PeopleSoft, use the standard pfile created with the Amazon RDS Custom DB instance. Add the parameters you need for PeopleSoft.

1. Switch to <code>rds user rdsdb</code> by running the following command.<pre><code>$ sudo su - rdsdb</code></pre>2. Log in to SQL*Plus on the starter database, and create the pfile by running the following command.<pre><code>SQL> create pfile
 from spfile;</code></pre>This creates the pfile in <code>$ORACLE_HOME/dbs</code> .
3. Make a backup of this pfile.
4. Edit the pfile to add or update PeopleSof t parameters.<pre><code>*._gby_hash_aggreg
ation_enabled=false

*._unnest_subqu
ery=false

*.nls_language='
AMERICAN'</code></pre></td><td>DBA</td></tr>
</table>

Task	Description	Skills required
	``` *.nls_length_sem antics='CHAR' *.nls_territ ory='AMERICA' *.open_cursors=1000 *.db_files=1200 *.undo_tablespace=' UNDOTBS1' ``` PeopleSoft related parameters can be found in Oracle Support Note 1100831.1. 5. Remove the spfile reference from the pfile. ``` *.spfile='/rdsdbbi n/oracle/dbs/spfil eFSDM092.ora' ```	

Drop the starter database.	To drop the existing Amazon RDS Custom database, use the following code.
	``
	$ sqlplus / as sysdba
	SQL> shutdown immediate
	;
	SQL> startup mount
	 exclusive restrict;
	SQL> drop database;
	SQL> exit
	``	

Task	Description	Skills required
Restore the Amazon RDS Custom database from the backup.	Restore the database by using the following script. The script will first restore the control file and then restore the entire database from the backup pieces stored on the EFS mount. ```#!/bin/bash Dt=`date +'%Y%m%d-%H%M'` BACKUP_LOG="rman-${ORACLE_SID}-$Dt" export TAGDATE=`date +%Y%m%d%H%M`; LOGPATH=/rdsdbdata/scripts/logs rman target / >> $LOGPATH/rman-${ORACLE_SID}-$Dt << EOF restore controlfile from "/efs/rman_backup/FSCM/FSDM092_20220713_12_1_CONTROL"; alter database mount; run { set newname for database to '/rdsdbdata/db/FSDM092_A/datafile/%f_%b'; SET NEWNAME FOR TEMPFILE 1 TO '/rdsdbdata/db/FSDM092_A/datafile/%f_%b'; RESTORE DATABASE; SWITCH DATAFILE ALL; SWITCH TEMPFILE ALL; RECOVER DATABASE;```	DBA

Task	Description	Skills required

	``
	}
	EOF
	sqlplus / as sysdba
	 >> $LOGPATH/rman-${OR
	ACLE_SID}-$Dt<<-EOF
	ALTER DATABASE RENAME
	 FILE '/u01/psoft/db/
	oradata/FSDMO92/redo0
	1.log' TO '/rdsdbda
	ta/db/FSDMO92_A/on
	linelog/redo01.log';
	ALTER DATABASE RENAME
	 FILE '/u01/psoft/db/
	oradata/FSDMO92/redo0
	2.log' TO '/rdsdbda
	ta/db/FSDMO92_A/on
	linelog/redo02.log';
	ALTER DATABASE RENAME
	 FILE '/u01/psoft/db/
	oradata/FSDMO92/redo0
	3.log' TO '/rdsdbda
	ta/db/FSDMO92_A/on
	linelog/redo03.log';
	alter database clear
	 unarchived logfile
	 group 1;
	alter database clear
	 unarchived logfile
	 group 2;
	alter database clear
	 unarchived logfile
	 group 3;
	alter database open
	 resetlogs;
	EXIT
	EOF
	``	

Retrieve passwords from Secrets Manager, create users, and change passwords

Task	Description	Skills required
Retrieve the password from Secrets Manager.	You can perform this step by using the AWS console or the AWS CLI. The following steps show instructions for the console. 1. Sign in to the AWS Management Console and open the Amazon RDS console. 2. In the navigation pane, choose **Databases**, and then select the Amazon RDS database. 3. Choose the **Configuration** tab, and note the resource ID for the instance. It will be in the format `db-<ID>` (for example, `db-73GJNHLGDNZNDOXNWXSECUW6LE`). 4. Open the Secrets Manager console. 5. Choose the secret that has the same name as `do-not-delete-custom-<resource_id>` , where `resource-id` refers to the resource ID that you noted in step 3. 6. Choose **Retrieve secret value**.	DBA

Task	Description	Skills required
	This password will be same for the `sys`, `system`, `rdsadmin`, and `admin` users.	

Task	Description	Skills required

Create the RDSADMIN user.	`RDSADMIN` is the database user for monitoring and orchestrating the Amazon RDS Custom DB instance. Because the starter database was dropped and the target database was restored from the source using RMAN, you must recreate this user after the restore operation to make sure that Amazon RDS Custom monitoring works as expected. You also must create a separate profile and tablespace for the `RDSADMIN` user.
	
	1. Enter the following commands at a SQL prompt.
	
	`SQL> set echo on feedback on serverout on`
	`SQL> @?/rdbms/admin/utlpwdmg.sql`
	`SQL> ALTER PROFILE DEFAULT`
	`LIMIT`
	`FAILED_LOGIN_ATTEMPTS UNLIMITED`
	`PASSWORD_LIFE_TIME UNLIMITED`
	`PASSWORD_VERIFY_FUNCTION NULL;`
		DBA

Task	Description	Skills required
	2. Create the profile `RDSADMIN`. ```SQL> set echo on feedback on serverout on SQL> alter session set "_oracle_script"=true; SQL> CREATE PROFILE RDSADMIN LIMIT COMPOSITE_LIMIT UNLIMITED SESSIONS_PER_USER UNLIMITED CPU_PER_SESSION UNLIMITED CPU_PER_CALL UNLIMITED LOGICAL_READS_PER_SESSION UNLIMITED LOGICAL_READS_PER_CALL UNLIMITED IDLE_TIME UNLIMITED CONNECT_TIME UNLIMITED PRIVATE_SGA UNLIMITED FAILED_LOGIN_ATTEMPTS 10 PASSWORD_LIFE_TIME UNLIMITED PASSWORD_REUSE_TIME UNLIMITED PASSWORD_REUSE_MAX UNLIMITED PASSWORD_VERIFY_FUNCTION NULL PASSWORD_LOCK_TIME 86400/86400```	

Task	Description	Skills required

	`PASSWORD_GRACE_TIME 604800/86400;`
	3. Create the `RDSADMIN` tablespace.
	`SQL> CREATE BIGFILE TABLESPACE rdsadmin '/rdsdbdata/db/FSDMO92_A/datafile/rdsadmin.dbf' DATAFILE SIZE 7M AUTOEXTEND ON NEXT 1m LOGGING ONLINE PERMANENT BLOCKSIZE 8192 EXTENT MANAGEMENT LOCAL AUTOALLOCATE DEFAULT NOCOMPRESS SEGMENT SPACE MANAGEMENT AUTO;`
	4. Create the `RDSADMIN` user. Replace the `RDSADMIN` password with the password you obtained earlier from Secrets Manager.
	`SQL> CREATE USER rdsadmin IDENTIFIED BY xxxxxxxxxx DEFAULT TABLESPACE rdsadmin TEMPORARY TABLESPACE TEMP profile rdsadmin ;`
	5. Grant privileges to RDSADMIN.	

Task	Description	Skills required

	<pre>SQL> GRANT "CONNECT"
	 TO RDSADMIN WITH
	 ADMIN OPTION;
	SQL> GRANT "RESOURCE
	" TO RDSADMIN WITH
	 ADMIN OPTION;
	SQL> GRANT "DBA" TO
	 RDSADMIN;
	SQL> GRANT "SELECT_C
	ATALOG_ROLE" TO
	 RDSADMIN WITH ADMIN
	 OPTION;
	SQL> GRANT ALTER
	 SYSTEM TO RDSADMIN;
	SQL> GRANT UNLIMITED
	 TABLESPACE TO
	 RDSADMIN;
	SQL> GRANT SELECT ANY
	 TABLE TO RDSADMIN;
	SQL> GRANT ALTER
	 DATABASE TO
	 RDSADMIN;
	SQL> GRANT ADMINISTER
	 DATABASE TRIGGER TO
	 RDSADMIN;
	SQL> GRANT ANY OBJECT
	 PRIVILEGE TO RDSADMIN
	 WITH ADMIN OPTION;
	SQL> GRANT INHERIT
	 ANY PRIVILEGES TO
	 RDSADMIN;
	SQL> ALTER USER
	 RDSADMIN DEFAULT ROLE
	 ALL;</pre>6. Set the SYS, SYSTEM, and DBSNMP user profiles to RDSADMIN.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>SQL> set echo on
 feedback on serverout
 on
SQL> alter user SYS
 profile RDSADMIN;
SQL> alter user SYSTEM
 profile RDSADMIN;
SQL> alter user DBSNMP
 profile RDSADMIN;</code></pre></td><td></td></tr>
<tr><td>Create the master user.</td><td>Because the starter database was dropped and the target database was restored from the source by using RMAN, you must recreate the master user. In this example, the master user name is <code>admin</code>.<pre><code>SQL> create user
 admin identified by
 <password>;
SQL> grant dba to admin</code></pre></td><td>DBA</td></tr>
</table>

Task	Description	Skills required

Change the system passwords .	Change the system passwords by using the password you retrieved from Secrets Manager.
	`SQL> alter user sys identified by xxxxxxxxxxx;`
	`SQL> alter user system identified by xxxxxxxxxx;`
	If you don't change these passwords, Amazon RDS Custom displays the error message, "The database monitoring user or user credentials have changed."
		DBA

Configure the TNS entries for Amazon RDS Custom and PeopleSoft

Task	Description	Skills required
Configure the tnsnames file.	To connect to the database from the application tier, configure the `tnsnames.ora` file so you can connect to the database from the application tier. In the following example, you can see that there is a soft link to the `tnsnames.ora` file, but the file is empty by default.	DBA

Task	Description	Skills required

	<pre>$ cd /rdsdbbin/oracle/n
	etwork/admin
	$ ls -ltr
	-rw-r--r-- 1 rdsdb
	 database 1536 Feb 14
	 2018 shrept.lst
	lrwxrwxrwx 1 rdsdb
	 database 30 Apr 5
	 13:19 listener.ora -
	> /rdsdbdata/config/
	listener.ora
	lrwxrwxrwx 1 rdsdb
	 database 28 Apr 5
	 13:19 sqlnet.ora -
	> /rdsdbdata/config/
	sqlnet.ora
	lrwxrwxrwx 1 rdsdb
	 database 30 Apr 5
	 13:19 tnsnames.ora -
	> /rdsdbdata/config/
	tnsnames.ora</pre>
	1. Create the `tsnames.ora` entry. Because of the way Amazon RDS automation parses the files, you have to make sure that the entry doesn't contain any white spaces, comments, or extra lines. Otherwise, you might run into issues when using some of the APIs, such [as create-db-instance-read-replica](#).
	2. Replace the port, host, and SID in accordance with your PeopleSoft	

Task	Description	Skills required

	database requirements. Use the following code as an example.
	
	`$ vi tnsnames.ora`
	
	`FSDMO92=(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = x.x.x.x)(PORT = 1521))) (CONNECT_DATA = (SERVER = DEDICATED) (SID = FSDMO92)))`
	
	3. To confirm that the PeopleSoft database can be reached, run the following command.
	
	`$ tnsping FSDMO92`
	
	`TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 14-JUL-2022 10:16:45`
	
	`Copyright (c) 1997, 2021, Oracle.  All rights reserved.`
	
	`Used parameter files: /rdsdbbin/oracle/network/admin/sqlnet.ora`
	
	`Used TNSNAMES adapter to resolve the alias`	

Task	Description	Skills required
	```Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = x.x.x.x)(PORT = 1521))) (CONNECT_DATA = (SERVER = DEDICATED) (SID = FSDM092))) OK (0 msec)```	

Create the spfile softlink

Task	Description	Skills required
Create the spfile softlink.	1. To create spfile in the location `/rdsdbdata/admin/FSDM092/pfile` , run the following command. ```SQL> create spfile='/rdsdbdata/admin/FSDM092/pfile/spfileFSDM092.ora' from pfile;``` 2. Navigate to `$ORACLE_HOME/dbs` , and create a soft link for the spfile. ```ln -s '/rdsdbdata/admin/FSDM092/pfile/spfileFSDM092.ora' spfileFSDM092.ora```	DBA

Task	Description	Skills required
	3. After this file is created, you can shut down and start the database by using the spfile.	

Perform post-migration steps

Task	Description	Skills required
Validate the schema, connections, and maintenan ce tasks.	To finalize the migration, perform the following tasks. • Gather schema statistics. • Ensure that the PeopleSoft application tier can connect to the new Amazon RDS Custom database. • Set up your backup and maintenance schedules.	DBA

Related resources

- Working with Amazon RDS Custom
- Amazon RDS Custom for Oracle – New Control Capabilities in Database Environment (blog post)
- Integrate Amazon RDS Custom for Oracle with Amazon EFS (blog post)
- Configuring Amazon RDS as an Oracle PeopleSoft Database (AWS whitepaper)

Migrate Oracle ROWID functionality to PostgreSQL on AWS

Rakesh Raghav and Ramesh Pathuri, Amazon Web Services

Summary

This pattern describes options for migrating the `ROWID` pseudocolumn functionality in Oracle Database to a PostgreSQL database in Amazon Relational Database Service (Amazon RDS) for PostgreSQL, Amazon Aurora PostgreSQL-Compatible Edition, or Amazon Elastic Compute Cloud (Amazon EC2).

In an Oracle database, the `ROWID` pseudocolumn is a physical address of a row in a table. This pseudocolumn is used to uniquely identify a row even if the primary key isn't present on a table. PostgreSQL has a similar pseudocolumn called `ctid`, but it cannot be used as a `ROWID`. As explained in the PostgreSQL documentation, `ctid` might change if it's updated or after every `VACUUM` process.

There are three ways you can create the `ROWID` pseudocolumn functionality in PostgreSQL:

- Use a primary key column instead of `ROWID` to identify a row in a table.
- Use a logical primary/unique key (which might be a composite key) in the table.
- Add a column with auto-generated values and make it a primary/unique key to mimic `ROWID`.

This pattern walks you through all three implementations and describes the advantages and disadvantages of each option.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Procedural Language/PostgreSQL (PL/pgSQL) coding expertise
- Source Oracle Database
- An Amazon RDS for PostgreSQL or Aurora PostgreSQL-Compatible cluster, or an EC2 instance to host the PostgreSQL database

Limitations

- This pattern provides workarounds for the `ROWID` functionality. PostgreSQL doesn't provide an equivalent to `ROWID` in Oracle Database.

Product versions

- PostgreSQL 11.9 or later

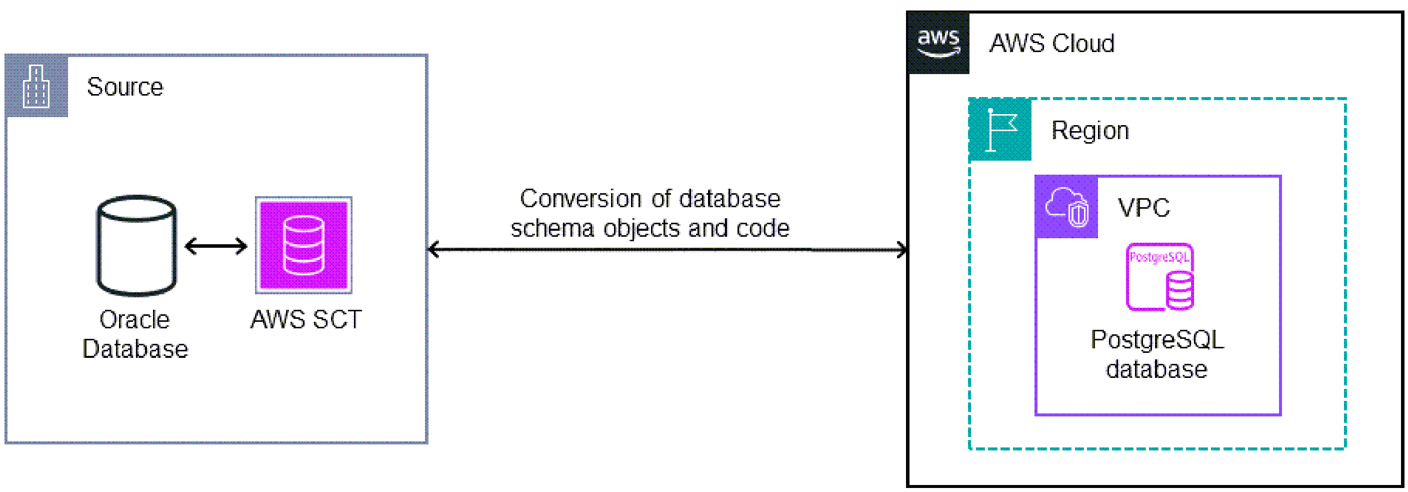
Architecture

Source technology stack

- Oracle Database

Target technology stack

- Aurora PostgreSQL-Compatible, Amazon RDS for PostgreSQL, or an EC2 instance with a PostgreSQL database



Implementation options

There are three options to work around the lack of `ROWID` support in PostgreSQL, depending on whether your table has a primary key or unique index, a logical primary key, or an identity attribute. Your choice depends on your project timelines, your current migration phase, and dependencies on application and database code.

Option	Description	Advantages	Disadvantages
Primary key or unique index	If your Oracle table has a primary key, you can use the attributes of this key to uniquely identify a row.	• No dependenc y on proprietary database features. • Minimal impact on performance, because primary key fields are indexed.	• Requires changes to application and database code that relies on `ROWID` to switch to primary key fields.
Logical primary/u nique key	If your Oracle table has a logical primary key, you can use the attributes of this key to uniquely identify a row. A logical primary key consists of an attribute or a set of attributes that can uniquely identify a row, but it isn't enforced on the database through a constraint.	• No dependenc y on proprietary database features.	• Requires changes to application and database code that relies on `ROWID` to switch to primary key fields. • Significant impact on performance if the attribute s of the logical primary key aren't indexed. However, you can add a unique index to prevent performan ce issues.
Identity attribute	if your Oracle table doesn't have a primary key, you can create an additional field as `GENERATED ALWAYS AS IDENTITY`. This attribute generates	• No dependenc y on proprietary database features. • PostgreSQL database populates the attribute and maintains its uniqueness.	• Requires changes to application and database code that relies on `ROWID` to switch to identity attribute. • Significant impact on performance

	a unique value whenever data is inserted into the table, so it can be used to uniquely identify a row for Data Manipulation Language (DML) operations.		if the additional field isn't indexed. However, you can add an index to prevent performan ce issues.

Tools

- [Amazon Relational Database Service (Amazon RDS) for PostgreSQL] helps you set up, operate, and scale a PostgreSQL relational database in the AWS Cloud.
- [Amazon Aurora PostgreSQL-Compatible Edition] is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- [AWS Command Line Interface (AWS CLI)] is an open-source tool that helps you interact with AWS services through commands in your command-line shell. In this pattern, you can use the AWS CLI to run SQL commands through **pgAdmin**.
- [pgAdmin] is an open-source management tool for PostgreSQL. It provides a graphical interface that helps you create, maintain, and use database objects.
- [AWS Schema Conversion Tool (AWS SCT)] supports heterogeneous database migrations by automatically converting the source database schema and a majority of the custom code to a format that's compatible with the target database.

Epics

Identify the source tables

Task	Description	Skills required
Identify Oracle tables that use the `ROWID` attribute.	Use the AWS Schema Conversion Tool (AWS SCT) to identify Oracle tables that have `ROWID` functionality. For	DBA or developer

Task	Description	Skills required
	more information, see the AWS SCT documentation. —or— In Oracle, use the DBA_TAB_COLUMNS view to identify tables that have a ROWID attribute. These fields might be used to store alphanumeric 10-byte characters. Determine the usage and convert these to a VARCHAR field if appropriate.	
Identify code that references these tables.	Use AWS SCT to generate a migration assessment report to identify procedures affected by ROWID. For more information, see the AWS SCT documentation. —or— In the source Oracle database, use the text field of the dba_source table to identify objects that use ROWID functionality.	DBA or developer

Determine primary key usage

Task	Description	Skills required
Identify tables that don't have primary keys.	In the source Oracle database, use DBA_CONSTRAINTS to	DBA or developer

Task	Description	Skills required

	identify tables that don't have primary keys. This informati on will help you determine the strategy for each table. For example:
	`select dt.*`
	`from dba_tables dt`
	`where not exists (select 1`
	`from all_constraints ct`
	`where ct.owner = Dt.owner`
	`and ct.table_name = Dt.table_name`
	`and ct.constraint_type = 'P'`
	`)`
	`and dt.owner = '{schema}'`	

Identify and apply the solution

Task	Description	Skills required
Apply changes for tables that have a defined or logical primary key.	Make the application and database code changes shown in the [Additional informati on](#) section to use a unique primary key or a logical primary key to identify a row in your table.	DBA or developer

Task	Description	Skills required
Add an additional field to tables that don't have a defined or logical primary key.	Add an attribute of type `GENERATED ALWAYS AS IDENTITY`. Make the application and database code changes shown in the Additional information section.	DBA or developer
Add an index if necessary.	Add an index to the additiona l field or logical primary key to improve SQL performance.	DBA or developer

Related resources

- PostgreSQL CTID (PostgreSQL documentation)
- Generated Columns (PostgreSQL documentation)
- ROWID Pseudocolumn (Oracle documentation)

Additional information

The following sections provide Oracle and PostgreSQL code examples to illustrate the three approaches.

Scenario 1: Using a primary unique key

In the following examples, you create the table `testrowid_s1` with `emp_id` as the primary key.

Oracle code:

```
create table testrowid_s1 (emp_id integer, name varchar2(10), CONSTRAINT testrowid_pk
 PRIMARY KEY (emp_id));
INSERT INTO testrowid_s1(emp_id,name) values (1,'empname1');
INSERT INTO testrowid_s1(emp_id,name) values (2,'empname2');
INSERT INTO testrowid_s1(emp_id,name) values (3,'empname3');
INSERT INTO testrowid_s1(emp_id,name) values (4,'empname4');
commit;
```

```
SELECT rowid,emp_id,name FROM testrowid_s1;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3pAAAAAAAMOAAA          1 empname1
AAAF3pAAAAAAAMOAAB          2 empname2
AAAF3pAAAAAAAMOAAC          3 empname3
AAAF3pAAAAAAAMOAAD          4 empname4

UPDATE testrowid_s1 SET name = 'Ramesh' WHERE rowid = 'AAAF3pAAAAAAAMOAAB' ;
commit;

SELECT rowid,emp_id,name FROM testrowid_s1;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3pAAAAAAAMOAAA          1 empname1
AAAF3pAAAAAAAMOAAB          2 Ramesh
AAAF3pAAAAAAAMOAAC          3 empname3
AAAF3pAAAAAAAMOAAD          4 empname4
```

PostgreSQL code:

```
CREATE TABLE public.testrowid_s1
(
    emp_id integer,
    name character varying,
    primary key (emp_id)
);

insert into public.testrowid_s1 (emp_id,name) values
(1,'empname1'),(2,'empname2'),(3,'empname3'),(4,'empname4');

select emp_id,name from testrowid_s1;
 emp_id |   name
--------+----------
      1 | empname1
      2 | empname2
      3 | empname3
      4 | empname4

update testrowid_s1 set name = 'Ramesh' where emp_id = 2 ;

select emp_id,name from testrowid_s1;
 emp_id |   name
```

```
--------+----------
       1 | empname1
       3 | empname3
       4 | empname4
       2 | Ramesh
```

Scenario 2: Using a logical primary key

In the following examples, you create the table `testrowid_s2` with `emp_id` as the logical primary key.

Oracle code:

```
create table testrowid_s2 (emp_id integer, name varchar2(10) );
INSERT INTO testrowid_s2(emp_id,name) values (1,'empname1');
INSERT INTO testrowid_s2(emp_id,name) values (2,'empname2');
INSERT INTO testrowid_s2(emp_id,name) values (3,'empname3');
INSERT INTO testrowid_s2(emp_id,name) values (4,'empname4');
commit;

SELECT rowid,emp_id,name FROM testrowid_s2;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3rAAAAAAAMeAAA          1 empname1
AAAF3rAAAAAAAMeAAB          2 empname2
AAAF3rAAAAAAAMeAAC          3 empname3
AAAF3rAAAAAAAMeAAD          4 empname4

UPDATE testrowid_s2 SET name = 'Ramesh' WHERE rowid = 'AAAF3rAAAAAAAMeAAB' ;
commit;

SELECT rowid,emp_id,name FROM testrowid_s2;
ROWID                  EMP_ID NAME
------------------ ---------- ----------
AAAF3rAAAAAAAMeAAA          1 empname1
AAAF3rAAAAAAAMeAAB          2 Ramesh
AAAF3rAAAAAAAMeAAC          3 empname3
AAAF3rAAAAAAAMeAAD          4 empname4
```

PostgreSQL code:

```
CREATE TABLE public.testrowid_s2
(
```

```
    emp_id integer,
    name character varying
);

insert into public.testrowid_s2 (emp_id,name) values
(1,'empname1'),(2,'empname2'),(3,'empname3'),(4,'empname4');

select emp_id,name from testrowid_s2;
 emp_id |   name
--------+----------
      1 | empname1
      2 | empname2
      3 | empname3
      4 | empname4

update testrowid_s2 set name = 'Ramesh' where emp_id = 2 ;

select emp_id,name from testrowid_s2;
 emp_id |   name
--------+----------
      1 | empname1
      3 | empname3
      4 | empname4
      2 | Ramesh
```

Scenario 3: Using an identity attribute

In the following examples, you create the table `testrowid_s3` with no primary key and by using an identity attribute.

Oracle code:

```
create table testrowid_s3 (name varchar2(10));
INSERT INTO testrowid_s3(name) values ('empname1');
INSERT INTO testrowid_s3(name) values ('empname2');
INSERT INTO testrowid_s3(name) values ('empname3');
INSERT INTO testrowid_s3(name) values ('empname4');
commit;

SELECT rowid,name FROM testrowid_s3;
ROWID              NAME
------------------ ----------
AAAF3sAAAAAAAMmAAA empname1
```

```
AAAF3sAAAAAAAMmAAB empname2
AAAF3sAAAAAAAMmAAC empname3
AAAF3sAAAAAAAMmAAD empname4

UPDATE testrowid_s3 SET name = 'Ramesh' WHERE rowid = 'AAAF3sAAAAAAAMmAAB' ;
commit;

SELECT rowid,name FROM testrowid_s3;
ROWID              NAME
------------------ ----------
AAAF3sAAAAAAAMmAAA empname1
AAAF3sAAAAAAAMmAAB Ramesh
AAAF3sAAAAAAAMmAAC empname3
AAAF3sAAAAAAAMmAAD empname4
```

PostgreSQL code:

```
CREATE TABLE public.testrowid_s3
(
    rowid_seq bigint generated always as identity,
    name character varying
);

insert into public.testrowid_s3 (name) values
('empname1'),('empname2'),('empname3'),('empname4');

select rowid_seq,name from testrowid_s3;
 rowid_seq |   name
-----------+----------
         1 | empname1
         2 | empname2
         3 | empname3
         4 | empname4

update testrowid_s3 set name = 'Ramesh' where rowid_seq = 2 ;

select rowid_seq,name from testrowid_s3;
 rowid_seq |   name
-----------+----------
         1 | empname1
         3 | empname3
         4 | empname4
         2 | Ramesh
```

Migrate Oracle Database error codes to an Amazon Aurora PostgreSQL-Compatible database

Sai Parthasaradhi and Veeranjaneyulu Grandhi, Amazon Web Services

Summary

This pattern shows how to migrate Oracle Database error codes to an [Amazon Aurora PostgreSQL-Compatible Edition](https://aws.amazon.com/rds/aurora/postgresql-features/) database by using a predefined metadata table.

Oracle Database error codes don't always have a corresponding PostgreSQL error code. This difference in error codes can make it difficult to configure the processing logic of the procedures or functions in the target PostgreSQL architecture.

You can simplify the process by storing the source and target database error codes that are meaningful to your PL/pgSQL program in a metadata table. Then, configure the table to flag valid Oracle Database error codes and map them to their PostgreSQL equivalents before continuing with the remaining process logic. If the Oracle Database error code isn't in the metadata table, the process exits with the exception. Then, you can manually review the error details and add the new error code to the table if your program requires it.

By using this configuration, your Amazon Aurora PostgreSQL-Compatible database can handle errors in the same way that your source Oracle database does.

> **Note**
>
> Configuring a PostgreSQL database to handle Oracle Database error codes correctly usually requires changes to the database and application code.

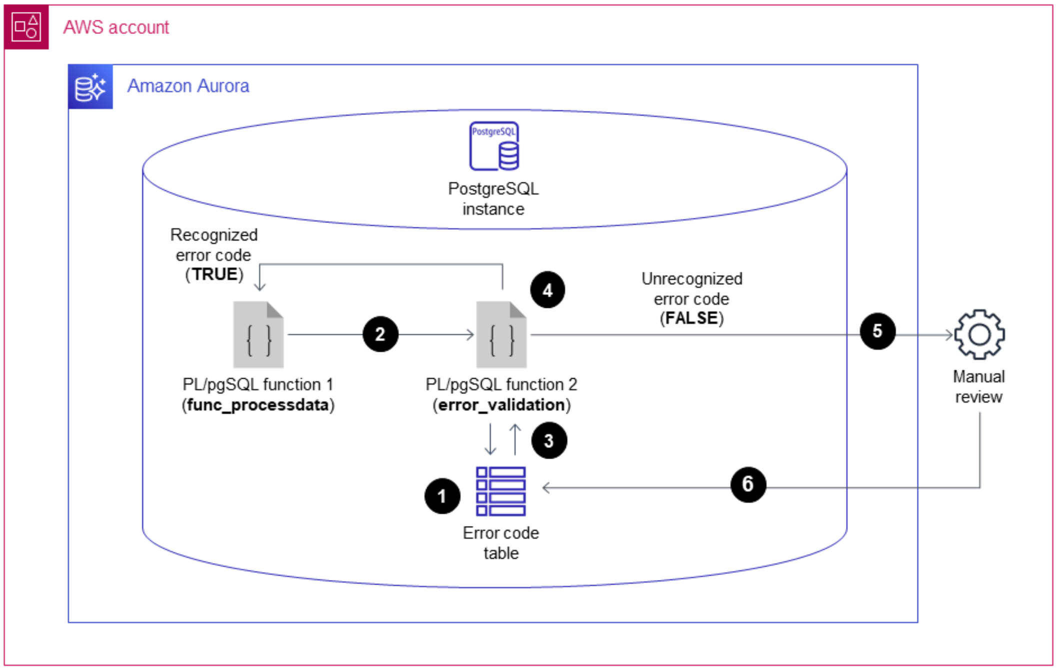
Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Oracle Database with instance and listener services up and running
- An Amazon Aurora PostgreSQL-Compatible cluster that's up and running
- Familiarity with Oracle Database
- Familiarity with PostgreSQL databases

Architecture

The following diagram shows an example Amazon Aurora PostgreSQL-Compatible database workflow for data error code validation and handling:



The diagram shows the following workflow:

1. A table holds Oracle Database error codes and classifications and their equivalent PostgreSQL error codes and classifications. The table includes a **valid_error** column that classifies if specific, predefined error codes are valid or not.
2. When a PL/pgSQL function (**func_processdata**) throws an exception, it invokes a second PL/pgSQL function (**error_validation**).
3. The **error_validation** function accepts the Oracle Database error code as an input argument. Then, the function checks the incoming error code against the table to see if the error is included in the table.
4. If the Oracle Database error code is included in the table, then the **error_validation** function returns a **TRUE** value and the process logic continues. If the error code isn't included in the table, then the function returns a **FALSE** value, and the process logic exits with an exception.
5. When the function returns a **FALSE** value, then the error details are manually reviewed by the application's functional lead to determine its validity.

6. The new error code is then either manually added to the table or not. If the error code is valid and added to the table, then the **error_validation** function returns a **TRUE** value the next time the exception occurs. If the error code isn't valid, and the process must fail when the exception occurs, then the error code isn't added to the table.

Technology stack

- Amazon Aurora PostgreSQL
- pgAdmin
- Oracle SQL Developer

Tools

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- pgAdmin is an open-source administration and development tool for PostgreSQL. It provides a graphical interface that simplifies the creation, maintenance, and use of database objects.
- Oracle SQL Developer is a free, integrated development environment that simplifies the development and management of Oracle Database in both traditional and cloud deployments.

Epics

Migrate Oracle Database error codes to your Amazon Aurora PostgreSQL-Compatible database

Task	Description	Skills required

Create a table in the Amazon Aurora PostgreSQL-Compatible database.	Run the following PostgreSQL CREATE TABLE command:
	`(`
	`source_error_code numeric NOT NULL,`
	`target_error_code character varying NOT NULL,`
		PostgreSQL Developer, Oracle, RDS/Aurora for PostgreSQL

Task	Description	Skills required
	``` valid_error character varying(1) NOT NULL); ```	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Add PostgreSQL error codes and their corresponding Oracle Database error codes to the table.</td><td>Run the PostgreSQL INSERT command to add the required error code values to the error_codes table.

The PostgreSQL error codes must use the character varying data type (SQLSTATE value). The Oracle error codes must use the numeric data type (SQLCODE value).

Example Insert statements:

<pre>insert into error_codes values (-1817,'22007','Y');
insert into error_codes values (-1816,'22007','Y');
insert into error_codes values (-3114,'08006','N');</pre>
Note

If you’re catching Oracle-specific Java database connectivity (JDBC) exceptions, you must replace those exceptions with either generic cross-database exceptions or switch</td><td>PostgreSQL Developer, Oracle, RDS/Aurora for PostgreSQL</td></tr>
</table>

Task	Description	Skills required
	to PostgreSQL-specific exceptions.	
Create a PL/pgSQL function to validate error codes.	Create a PL/pgSQL function by running the PostgreSQL CREATE FUNCTION command. Make sure that the function does the following: • Accepts the Oracle error codes thrown by a program. • Checks if error codes are present in the **error_codes** table. • Returns **TRUE** or **FALSE** value, based on if the error code is present in the metadata table or not.	PostgreSQL Developer, Oracle, RDS/Aurora for PostgreSQL

Task	Description	Skills required
Manually review new error codes as they're recorded by the PL/pgSQL function.	Manually review the new error codes. If a new error code is valid for your use case, add it to the **error_codes** table by running the PostgreSQL **INSERT** command. -or- If a new error code isn't valid for your use case, don't add it to the table. The process logic will continue to fail and exit with exception when the error occurs.	PostgreSQL Developer , Oracle, RDS/Aurora for PostgreSQL

Related resources

Appendix A. PostgreSQL Error Codes (PostgreSQL documentation)

Database error messages (Oracle Database documentation)

Migrate Redis workloads to Redis Enterprise Cloud on AWS

Antony Prasad Thevaraj, Amazon Web Services

Srinivas Pendyala, Redis

Summary

This pattern discusses the high-level process for migrating Redis workloads to Redis Enterprise Cloud on Amazon Web Services (AWS). It describes the migration steps, provides information about the selection of tools available, and discusses the advantages, disadvantages, and steps for using each tool. Optionally, if you require additional help in migrating workloads from Redis, you can engage Redis Professional Services.

If you run Redis OSS or Redis Enterprise Software on premises, you're familiar with the significant administrative overhead and operational complexity of maintaining your Redis databases in your data center. By migrating your workloads to the cloud, you can significantly reduce this operational burden and take advantage of Redis Enterprise Cloud, which is a fully hosted database as a service (DBaaS) offering from Redis. This migration helps increase your business agility, improves application reliability, and reduces overall costs while you gain access to the newest Redis Enterprise Cloud on AWS features such as 99.999% availability, architectural simplicity, and scale.

There are potential applications for Redis Enterprise Cloud in the financial services, retail, healthcare, and gaming sectors, as well as in use cases that require solutions for fraud detection, real-time inventory, claims processing, and session management. You can use Redis Enterprise Cloud to connect to your AWS resources—for example, to an application server that is running on Amazon Elastic Compute Cloud (Amazon EC2) instances, or to a microservice that is deployed as an AWS Lambda service.

Prerequisites and limitations

Assumptions

- You are currently operating an on-premises database system that you want to migrate to the cloud.
- You have identified the migration requirements for your workloads, including:
  - Data consistency requirements
  - Infrastructure and system environment requirements

  - Data mapping and transformation requirements
  - Functional testing requirements
  - Performance testing requirements
  - Validation requirements
  - Defined cutover strategy
- You have assessed timelines and cost estimates required for the migration.
- Your requirements take into consideration the scope of the work and the systems and databases you have identified to be part of the migration.
- You have identified the stakeholders along with their roles and responsibilities in a responsible, accountable, consulted, informed (RACI) matrix.
- You have received the necessary agreement and approvals from all stakeholders.

Cost

Depending on the technical specifications of your existing source database (for example, memory sizing, throughput, and total data size), a Redis solutions architect can size the target system on Redis Enterprise Cloud. For general pricing information, see Redis Pricing on the Redis website.

People and skills

The migration process involves the following roles and responsibilities.

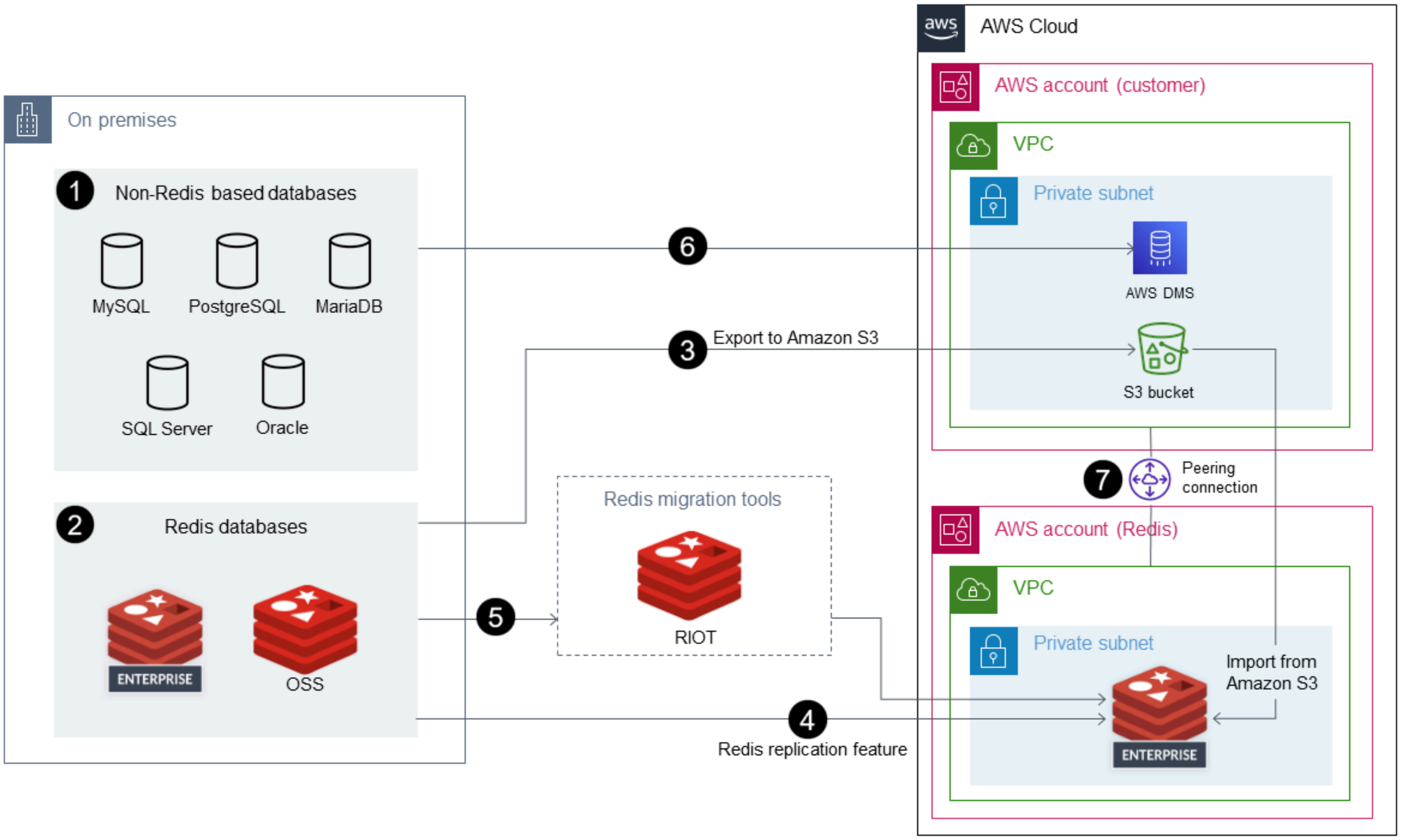
Role	Description	Skills required
Migration solutions architect	A technical architect who has expertise in defining, planning, and implementing migration strategies	Technical and application-level understanding of source and target systems; experienc e with migrating workloads to the cloud
Data architect	A technical architect who has broad experience in defining, implementing, and deliverin g data solutions for a wide variety of databases	Data modeling for structured and unstructured data, deep understanding and experienc e in implementing databases for an enterprise

Redis solutions architect	A technical architect who can help architect an optimally sized Redis cluster for the appropriate use case	Expertise in architecting and deploying Redis solutions for a wide variety of use cases
Cloud solutions architect	A technical architect who has a deeper understanding of cloud solutions, especially on AWS	Expertise in architecting solutions for the cloud; workload migration and application modernization experience
Enterprise architect	A technical architect who has a complete understanding of the technical landscape at your organization, who has a shared vision for the future roadmap, and who practices and establishes standardized architectural best practices across all teams in your organization	Software architecture certifications such as TOGAF, foundational software engineering skills, and solutions architecture and enterprise architecture expertise
IT or DevOps engineer	An engineer who is responsib le for creating and maintaini ng the infrastructure, including monitoring the infrastructure for issues, performing maintenance tasks, and making updates as needed.	Strong understanding of various technologies, including operating systems, networking, and cloud computing; familiarity with programming languages such as Python, Bash, and Ruby, as well as tools such as Docker, Kubernetes, and Ansible

Architecture

Migration options

The following diagram shows options for migrating your on-premises (Redis-based or other) data sources to AWS. It shows several migration tools that you can choose from, such as exporting Redis Database (RDB) files to Amazon Simple Storage Service (Amazon S3), using the Redis replication feature, or using AWS DMS.

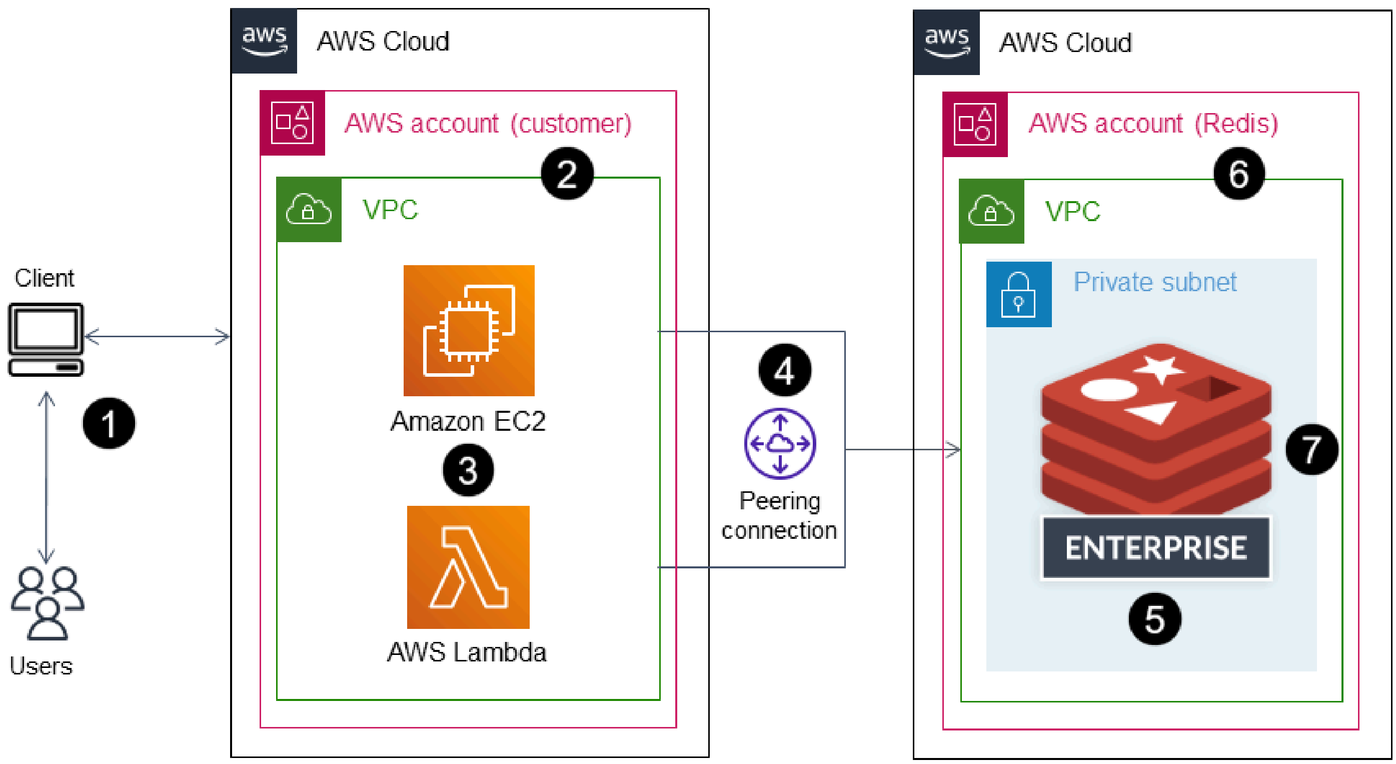


1. On-premises data sources: Databases that aren't based on Redis, such as MySQL, PostgreSQL, Oracle, SQL Server, or MariaDB.
2. On-premises data sources: Redis protocol-based databases such as Redis OSS and Redis Enterprise Software.
3. The simplest way to migrate data from Redis-based databases is to export RDB files and import them into the target Redis Enterprise Cloud on AWS.
4. Alternatively, you can migrate the data from source to target by using the replication feature (`ReplicaOf`) in Redis.
5. If your data migration requirements include transformation of data, you can employ Redis Input/Output Tools (RIOT) to migrate the data.
6. Alternatively, you can use AWS Data Migration Service (AWS DMS) to migrate the data from SQL-based databases.

7. You must use virtual private cloud (VPC) peering for AWS DMS to migrate the data successfully into the target Redis Enterprise Cloud on AWS.

Target architecture

The following diagram shows a typical deployment architecture for Redis Enterprise Cloud on AWS and illustrates how it can be used with key AWS services.



1. You can connect to the business applications that are backed by Redis Enterprise Cloud on AWS.
2. You can run business applications in your own AWS account, in a VPC within that account.
3. You can use Redis Enterprise Cloud database endpoints to connect to your applications. Examples include an application server running on EC2 instances, a microservice deployed as an AWS Lambda service, an Amazon Elastic Container Service (Amazon ECS) application, or an Amazon Elastic Kubernetes Service (Amazon EKS) application.
4. Business applications running in your VPC require a VPC peer connection to the Redis Enterprise Cloud VPC. This enables the business applications to connect securely over private endpoints.
5. Redis Enterprise Cloud on AWS is an in-memory NoSQL database platform deployed as a DBaaS on AWS and is fully managed by Redis.

6. Redis Enterprise Cloud is deployed within a VPC in a standard AWS account that is created by Redis.
7. For security reasons, Redis Enterprise Cloud is deployed in a private subnet that can be accessed at both private and public endpoints. We recommend that you connect your client applications to Redis on private endpoints. If you plan to use a public endpoint, we strongly recommend that you enable TLS to encrypt the data between your client applications and Redis Enterprise Cloud.

The Redis migration methodology aligns with the AWS migration methodology, which is illustrated in Mobilize your organization to accelerate large-scale migrations on the AWS Prescriptive Guidance website.

Automation and scale

The environment setup tasks for the migration can be automated through AWS Landing Zone and infrastructure as code (IaC) templates for automation and scale. These are discussed in the Epics section of this pattern.

Tools

Based on your data migration requirements, you can choose from a selection of technological options to migrate your data to Redis Enterprise Cloud on AWS. The following table describes and compares these tools.

Tool	Description	Advantages	Disadvantages
RDB export and import	You export the data from the source (for example, Redis OSS or Redis Enterprise Software) database in the form of RDB files. If your database is provided through a Redis OSS Cluster, you export each master shard to an RDB.	• Simple. • Works with any Redis-based solution that can export data in RDB format as a source (includin g Redis OSS and Redis Enterprise Software). • Achieves data consistency with a simple process.	• Doesn't address data transform ation requirements or support logical database merges. • Time-consuming for larger datasets. • No delta migration support can lead to longer downtime.

You then import all the RDB files in one step. If your source database is based on an OSS Cluster but your target database isn't using the OSS Cluster API, you have to change your application source code to use a standard Redis client library.

Data transformation requirements or logical database merges require a more complex process, which is explained under *Logical database merge* later in this table.

[Redis replication feature](#) (active-passive)	You can continuously replicate data from a Redis OSS, Enterprise Software, or Enterprise Cloud database to a Redis Enterprise Cloud database. After the initial synchronization, the Redis replication feature (`ReplicaOf`) performs a delta migration, which means that there's nearly no observed application downtime. The Redis replication feature is intended to be used in an active-passive way. The target is assumed to be passive and gets fully resynchronized (flushed and synchronized from the source database). Therefore, switching between the source and the target is somewhat more complicated. It's possible to replicate from a Redis OSS Cluster to a	• Supports continuous replication (initial data load followed by deltas). • Nearly no downtime (depends on replication lag). • Achieves data consistency.	• Only one site is intended to be active, so switching between sites is more complicated. • Supports a maximum of 32 master shards when you migrate from an OSS Cluster.

	standard clustered Redis Enterprise Cloud database by specifying all the master shards of the OSS Cluster as sources. However, the Redis replicati on feature allows a maximum of 32 source databases.		
AWS DMS	You can use AWS DMS to migrate data from any supported source database to a target Redis data store with minimal downtime. For more information, see Using Redis as a target for AWS DMS in the AWS DMS documentation.	• Supports the migration of both NoSQL and SQL data sources. • Works well with other AWS services. • Supports live migration and change data capture (CDC) use cases.	• Redis key-value s cannot contain special characters such as %. • Doesn't support the migration of data that has special characters in rows or in field names. • Doesn't support full large binary object (LOB) mode.

Logical database merge	Special database merge requirements might require a custom data migration solution. For example, you might have four logical databases (`SELECT 0..3`) in Redis OSS, but you might want to use a single database endpoint instead of moving the data to multiple Redis Enterprise Cloud databases. Redis Enterprise doesn’t support selectable logical databases, so you would have to transform the source database’s physical data model. For example, you could map each database index to a prefix (`0` to `usr`, `1` to `cmp`, and so on), and then use a migration script or an extract, transform, and load (ETL) tool to output an RDB file, which you can then import into the target database.	• Granular control on shaping the data during migration to the target system by using custom scripts.	• If you decide not to complete the migration, rollback can be very challenging, especially if newer data has to be rolled back to source systems. • Cost to build can be high if the goal is to build a one-off solution for a one-time migration. • Maintenance costs for code, infrastructure, development time, and other areas can be high if migration requirements change frequently.

In addition, you can use the following tools and services from AWS.

Assessment and discovery tools:

- AWS Application Discovery Service
- Migration Evaluator

Application and server migration tools:

- AWS Application Migration Service

Database migration tools:

- AWS Schema Conversion Tool (AWS SCT)
- AWS Database Migration Service (AWS DMS)

Data migration tools:

- AWS Storage Gateway
- AWS DataSync
- AWS Direct Connect
- AWS Snowball
- Amazon Data Firehose

Migration management:

- AWS Migration Hub

AWS Partner solutions:

- AWS Migration Competency Partners

Epics

Complete discovery and assessment tasks

Task	Description	Skills required
Identify workloads.	Identify the suitable candidate workloads that you want to migrate. Consider the following before you choose a workload for migration: • What is the business value in migrating or not migrating this workload? • Is there a contingency plan if this workload doesn't successfully migrate to the target system? Ideally, choose a workload that has maximum business impact with minimum risks involved. Keep the overall process iterative and migrate in small increments.	Data architect, Business champions, Migration project sponsors
Identify data sources and requirements; design data model.	Redis runs a workshop to accelerate discovery and to define migration planning for the project. As a part of this workshop, Redis teams identify the data sources and source data model requireme nts, and analyze how these can be remodeled in Redis Enterprise Cloud.	Redis solutions architect

Task	Description	Skills required
	The Redis migration team (Professional Services) performs a detailed data model design exercise with your organization. As a part of this exercise, the Redis team: • Identifies target Redis data structures. • Defines the data mapping strategy. • Documents the migration approach and recommend ations. • Reviews and finalizes the data model with the stakeholders.	

Task	Description	Skills required
Identify the characteristics of the source database.	Identify the Redis product that is used in the source and target environments. For example: • Is the source database an OSS Cluster database, a standalone Redis database, or a Redis Enterprise database? • Will the target database be a Redis Enterprise standard database or an OSS Cluster-compatible database? • What are the implications concerning the application source code?	Data architect
Gather current system SLA and other sizing metrics.	Determine the current service-level agreements (SLAs) expressed in terms of throughput (operations per second), latency, overall memory size per database, and high availability (HA) requirements.	Data architect

Task	Description	Skills required
Identify the characteristics of the target system.	Determine the answers to these questions: • How much data has to be migrated? • How long does it take to migrate the given amount of data? • What are the downtime requirements for the migration? Is it acceptable for your service or applicati on to be unavailable for a specific period? If so, for how long? • How consistent should the migrated data be? Can the target database be in a slightly inconsistent (outdated) state? • Does data have to be transformed before it is loaded to the target database? (For example, you might want to convert selectable DB indexes to prefixes before migration.) • Is the source database reachable from the target database's host (for example, from a peer VPC or from a public endpoint using encryption)?	Data architect, Redis solutions architect (optional)

Task	Description	Skills required
	• Complete a data sizing and Redis Cluster sizing exercise with a Redis technical architect. • Identify networking requirements, infrastru cture requirements, software versions, and software licensing, and procure any components before the migration. • Are there any security concerns associated with the transfer of this data?	
Identify dependencies.	Identify the upstream and downstream dependenc ies of the current system to be migrated. Make sure that the migration work is in alignment with other dependent system migration s. For example, if you're planning to migrate other business applications from on premises to the AWS Cloud, identify these applications and align them based on project goals, timelines, and stakeholders.	Data architect, enterprise architect

Task	Description	Skills required
Identify migration tools.	Depending on your data migration requirements (such as source data or downtime requirements), you can use any of the tools described previously in the Tools section. In addition, you can use: • Bidirectional (active-active) replication by using CRDB deployment. • Custom export/import scripts (for example, by using `DUMP/RESTORE` commands). • Additional export/import tools and helper tools such as RIOT, ECstats2, or ETL tools. • IaC tools such as Terraform or AWS CloudFormation templates.	Migration solutions architect, Redis solutions architect
Create a contingency plan.	Establish a contingency plan to roll back, in case you encounter problems during migration.	Project management, Technical teams, including architect

Complete security and compliance tasks

Task	Description	Skills required
Secure the Redis administration console.	To secure the administration console, follow the instructions in the Redis documentation.	IT infrastructure administrator
Secure the Redis database.	See the following pages in the Redis documentation to: • Define role-based access control. • Define network security. • Enable TLS.	
Secure Redis Cloud APIs.	When you enable the API, you can manage the API keys for all owners of your Redis Cloud account. For an overview of the security features of the API, see the API authentication documentation on the Redis website.	IT infrastructure administrator

Set up the new environment

Task	Description	Skills required
Set up a new environment on AWS.	This task includes: • AWS Landing Zone setup activities. The landing zone supports: • Multi-account deployments	IT or DevOps engineer

Task	Description	Skills required
	- Minimum security baseline - Automated way to provision new accounts with a security baseline and ISV prerequisites (networking, security configuration, and so on) - Notifications, centralized logging, and monitoring - ISV software configuration activities. This includes configurations that need to be included in the migration, such as product and workload settings and changes. - IaC activities such as configuring or customizi ng AWS CloudFormation or Terraform templates.	

Task	Description	Skills required
Deploy the migration architecture.	1. Set up Redis Enterprise Cloud on AWS. 2. Install migration tools such as RIOT or AWS DMS. See the Tools section for a list of available tools. 3. Establish connectivity between the application, migration, and database layers. 4. Create a sample workload that can flow through each layer and migrate a small set of sample data. You are now ready to run the actual data migration pipelines and test them.	IT or DevOps engineer

Set up networking

Task	Description	Skills required
Establish connectivity.	Establish connectivity between the on-premises infrastructure and AWS Cloud resources. Use security groups, AWS Direct Connect, and other resources to achieve this functionality. For more information, see Connect Your Data Center to AWS on the AWS website.	IT or DevOps engineer

Task	Description	Skills required
Set up VPC peering.	Establish VPC peering between the VPCs that run business applications (or the EC2 instances that run migration tools or the AWS DMS replication server) and the VPC that runs Redis Enterprise Cloud. For instructions, see Get started with Amazon VPC in the Amazon VPC documentation, and Enable VPC peering in the Redis documentation.	IT or DevOps engineer

Migrate data

Task	Description	Skills required
Choose a data migration tool.	Review the table in the Tools section to see descriptions, advantages, and disadvantages of these tools: • RDS export and import • Redis replication feature (`ReplicaOf`) • AWS DMS • Logical database merge The following rows describe the data migration tasks associated with each tool.	Migration solutions architect

Task	Description	Skills required
Option 1: Use RDB export and import.	1. Disconnect the source: Stop the traffic on the source database (for example, by disconnecting business applications). 2. Export: Export the source database's data as an RDB file. 3. Stage: Upload the data to a location that is accessibl e to the Redis Enterprise Cloud instances on AWS (for example, you can upload them to an S3 bucket or FTP server). 4. Import: Import the RDB files (by listing them all in one import step) to your Redis Enterprise Cloud target database. 5. Cut over: Move to the target database (for example, by connecting your application connect to it). For more information, see the [Redis documentation].	Migration solutions architect, Redis solutions architect

Task	Description	Skills required
Option 2: Use the Redis replication feature (active-p assive).	1. Connect database: Establish a `ReplicaOf` link between the source and target databases. 2. Run an initial sync: Wait until the initial synchroni zation between the source and target databases is complete. 3. Disconnect the source: Stop the traffic on the source database (for example, by disconnecting the application). 4. Run delta replication: Wait until the delta is replicated on the target database. 5. Cut over: Move to the target database (for example, by connecting your application to it). 6. Delete: Remove the `ReplicaOf` link between the source and target databases. For more information, see the Redis documentation.	Migration solutions architect, Redis solutions architect

Task	Description	Skills required
Option 3: Use AWS DMS.	1. Set up an AWS DMS replication instance: This instance performs all migration processes. For instructions: Working with an AWS DMS replication instance in the AWS DMS documentation. 2. Define the source database: Define the source endpoint. Test the connectivity between the source endpoint and the AWS DMS replication server. For instructions: Creating source and target endpoints in the AWS DMS documentation. 3. Set up the target database: Set up Redis Enterprise Cloud on AWS and set up the database to migrate to. 4. Define the target database: Define the target endpoint. Make sure that VPC peering is established between the VPC where AWS DMS is running and the VPC that hosts Redis Enterprise Cloud on AWS. Test the connectivity between the AWS DMS replication server and the target database.	Migration solutions architect, Redis solutions architect

Task	Description	Skills required
	5. Create an AWS DMS task: Create a task or a set of tasks to define the tables and replication processes you want to use to migrate the data. For instructions: [Working with AWS DMS tasks](#) in the AWS DMS documentation. 6. Migrate: Migrate the data by running the AWS DMS task. 7. Cut over: Move to the target database (for example, by connecting your application to it).	
Option 4: Use logical database merge.	This option involves using a migration script or ETL tool that can transform the source database's physical data model and generating an RDB file. Redis Professional Services can help with this step, if needed.	Migration solutions architect, Redis solutions architect

Migrate your application

Task	Description	Skills required
Align project management timelines and goals.	Align the migration project goals, milestones, and timelines of the application	Project management

Task	Description	Skills required
	layer with that of the Redis data migration project.	
Align testing activities.	After the application layer is migrated and modernized in the AWS Cloud, point the application layer to the newly migrated Redis Enterprise Cloud on AWS for testing.	Testing

Test

Task	Description	Skills required
Implement test plans.	Run the data migration routines and the scripts that were developed during the implementation phase in a testing environment, per test requirements, at your site.	Testing
Test data quality.	Test data quality after you migrate the data.	Testing
Test functionality.	Test data queries and the application layer to ensure that the application is performing at the same level as in the source system.	Testing

Cut over

Task	Description	Skills required
Make the cutover decision.	After all application-level and database-level testing is complete, the executive leadership team and stakeholders make the final decision regarding whether to cut over to the new environment on AWS based on the final results confirmed by the testing teams.	Project management, Business champions
Cut over to the AWS Cloud.	When you have confirmed that everything is in place, point the application layer to the newly migrated data and point clients to the new application layer that is running based on the new Redis Enterprise Cloud system on AWS.	IT or DevOps engineer, Data architect, Migration solutions architect, Redis solutions architect

Related resources

Redis resources

- Redis Enterprise Cloud documentation
- RIOT tool (GitHub repository)
- Terraform Provider (download)

AWS resources

- Demo migrations

- AWS Partner Solutions
- Documentation
- Blog posts
- White papers
- Tutorials and videos
- AWS cloud migration
- AWS Prescriptive Guidance

Additional information

For standard security requirements for migrating Redis workloads to the AWS Cloud, see the Best Practices for Security, Identity, and Compliance on the AWS website, and the Redis Trust Center on the Redis website.

Migrate SAP ASE on Amazon EC2 to Amazon Aurora PostgreSQL-Compatible using AWS SCT and AWS DMS

Amit Kumar and Ankit Gupta, Amazon Web Services

Summary

This pattern describes how to migrate an SAP Adaptive Server Enterprise (SAP ASE) database that is hosted on an Amazon Elastic Compute Cloud (Amazon EC2) instance to Amazon Aurora PostgreSQL-Compatible Edition by using AWS Schema Conversion Tool (AWS SCT) and AWS Database Migration Service (AWS DMS). The pattern focuses on both data definition language (DDL) conversions for stored objects and data migration.

Aurora PostgreSQL-Compatible supports online transaction processing (OLTP) workloads. This managed service provides configurations that automatically scale on demand. It can automatically start up, shut down, scale up, or scale down your database based on your application's needs. You can run your database in the cloud without managing any database instances. Aurora PostgreSQL-Compatible provides a cost-effective option for infrequent, intermittent, or unpredictable workloads.

The migration process consists of two main phases:

- Converting the database schema by using AWS SCT
- Migrating the data by using AWS DMS

Detailed instructions for both phases are provided in the *Epics* section. For information about troubleshooting issues that are specific to using AWS DMS with SAP ASE databases, see Troubleshooting issues with SAP ASE in the AWS DMS documentation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source SAP ASE database on an EC2 instance with server, database, and listener services up and running
- A target Aurora PostgreSQL-Compatible database

Limitations

- The port number for connections must be 5432.
- The huge_pages feature is on by default but can be modified.
- Point-in-time recovery (PITR) granularity is 5 minutes.
- Cross-Region replication is currently not available.
- The maximum storage size for an Aurora database is 128 TiB.
- You can create up to 15 read replicas.
- The table size limit is constrained only by the size of the Aurora cluster volume, so the maximum table size for an Aurora PostgreSQL-Compatible DB cluster is 32 TiB. We recommend that you follow best practices for table design, such as partitioning large tables.

Product versions

- Source database: AWS DMS currently supports SAP ASE 15, 15.5, 15.7, and 16.x. See the AWS DMS User Guide for the latest information about SAP ASE version support.
- Target database: PostgreSQL 9.4 and later (for version 9.x), 10.x, 11.x, 12.x, 13.x, and 14.x. See the AWS DMS User Guide for the latest supported PostgreSQL versions.
- Amazon Aurora 1.x or later. For the latest information, see Aurora PostgreSQL-Compatible releases and engine versions in the Aurora documentation.

Architecture

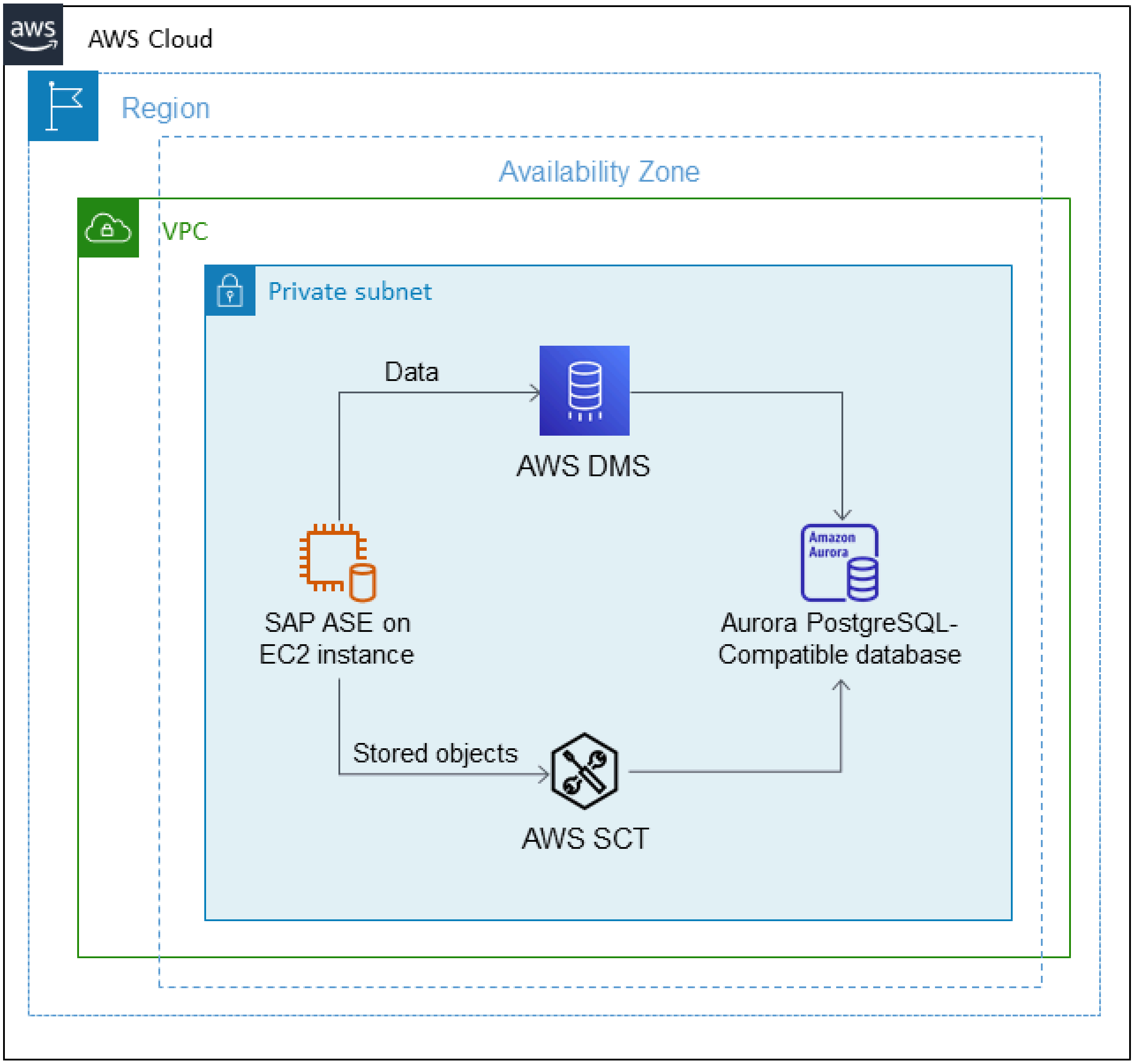
Source technology stack

- SAP ASE database running on Amazon EC2

Target technology stack

- Aurora PostgreSQL-Compatible database

Migration architecture



Tools

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- AWS Schema Conversion Tool (AWS SCT) supports heterogeneous database migrations by automatically converting the source database schema and most of the custom code to a format that's compatible with the target database.
- AWS DMS supports several different source and target databases. For more information, see Sources for Data Migration and Targets for Data Migration in the AWS DMS documentation. For

the most comprehensive version and feature support, we recommend that you use the latest version of AWS DMS.

Epics

Set up the environment

Task	Description	Skills required
Configure network access in the source EC2 instance.	Set up security groups in the EC2 instance that hosts your source SAP ASE database. For instructions, see Amazon EC2 security groups for Linux instances in the Amazon EC2 documentation.	Systems administrator
Create your target Aurora PostgreSQL-Compatible DB cluster.	Install, configure, and launch an Aurora PostgreSQL-Compatible cluster for your target database. For more information, see Creating an Amazon Aurora DB cluster in the Aurora documentation.	DBA
Set up authorization for the target DB cluster.	Set up security groups and firewalls for the target database. For instructions, see Creating an Amazon Aurora DB cluster in the Aurora documentation.	DBA, Systems administrator

Convert your database schema with AWS SCT

Task	Description	Skills required
Launch AWS SCT.	Launch AWS SCT by following the instructions in the [AWS SCT documentation](). AWS SCT provides a project-based user interface to automatically convert the database schema of your SAP ASE source database into a format that's compatible with your target Aurora PostgreSQL-Compatible DB instance.	DBA
Create AWS SCT endpoints.	Create endpoints for the source SAP ASE and target PostgreSQL databases. For instructions, see the [AWS SCT documentation]().	DBA
Create an assessment report.	Create a database migration assessment report to evaluate the migration and detect any incompatible objects and functions. For instructions, see the [AWS SCT documentation]().	DBA
Convert the schema.	Convert the database schema by following the instructions in the [AWS SCT documentation]().	DBA

Task	Description	Skills required
Validate database objects.	If AWS SCT cannot convert a database object, it will identify its name and other details. You must convert these objects manually. To identify these mismatches, follow the instructions in the AWS blog post [Validate database objects after migrating from SAP ASE to Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL]().	DBA

Analyze the AWS DMS migration

Task	Description	Skills required
Validate the source and target database versions.	Check the SAP ASE database versions for compatibility with AWS DMS. For more information, see Sources for AWS DMS and Targets for AWS DMS in the AWS DMS documentation.	DBA
Identify the requirements for the storage type and capacity.	Choose the appropriate storage capacity for the target database based on the size of your source database.	DBA, Systems administrator

Task	Description	Skills required
Choose the instance type, capacity, and other features of the replication instance.	Choose the instance type, capacity, storage features, and network features that meet your requirements. For guidance, see [Choosing the right AWS DMS replication instance for your migration] in the AWS DMS documentation.	DBA, Systems administrator
Identify network access security requirements.	Identify the network access security requirements for the source and target databases. Follow the guidance in [Setting up a network for a replication instance] in the AWS DMS documentation.	DBA, Systems administrator

Migrate the data

Task	Description	Skills required
Migrate the data by creating a migration task in AWS DMS.	To migrate data, create a task and follow the instructions in the [AWS DMS documentation.] We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.	DBA
Validate the data.	To validate that your data was migrated accurately from the	DBA

Task	Description	Skills required
	source database to the target database, follow the data validation guidelines provided in the AWS DMS documenta tion.	

Migrate the application

Task	Description	Skills required
Identify the application migration strategy.	Choose one of the seven strategies (7Rs) for migrating applications to the cloud.	DBA, App owner, Systems administrator
Follow the application migration strategy.	Complete the database tasks identified by the application team, including updating DNS connection details for the target database and updating dynamic queries.	DBA, App owner, Systems administrator

Cut over to the target database

Task	Description	Skills required
Switch the application clients over to the new infrastru cture.	Switch the connection from the source database to the target database. For more information, see the Cut over section of the *Migration strategy for relational databases*.	DBA, App owner, Systems administrator

Close the project

Task	Description	Skills required
Shut down the temporary AWS resources.	Terminate all migration tasks, replication instances , endpoints, and other AWS SCT and AWS DMS resources. For more information, see the [AWS DMS documentation]().	DBA, Systems administrator
Review and validate the project documents.	Validate all the steps in the project documentation to ensure that all tasks have been completed successfully.	DBA, App owner, Systems administrator
Close the project.	Close the migration project and provide any feedback.	DBA, App owner, Systems administrator

Related resources

References

- Enable encrypted connections for PostgreSQL DB instances in Amazon RDS (AWS Prescriptive Guidance)
- Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport (AWS Prescriptive Guidance)
- Amazon Aurora pricing
- Best practices with Amazon Aurora PostgreSQL-Compatible Edition (Amazon Aurora documentation)
- AWS SCT documentation
- AWS DMS documentation
- Using an SAP ASE Database as a Source for AWS DMS

Tutorials and videos

- Getting Started with AWS Database Migration Service
- AWS Database Migration Service (video)

Migrate Windows SSL certificates to an Application Load Balancer using ACM

Chandra Sekhar Yaratha and Igor Kovalchuk, Amazon Web Services

Summary

The pattern provides guidance for using AWS Certificate Manager (ACM) to migrate existing Secure Sockets Layer (SSL) certificates from websites that are hosted on on-premises servers or Amazon Elastic Compute Cloud (Amazon EC2) instances on Microsoft Internet Information Services (IIS). The SSL certificates can then be used with Elastic Load Balancing on AWS.

SSL protects your data, affirms your identity, provides better search engine rankings, helps meet Payment Card Industry Data Security Standard (PCI DSS) requirements, and improves customer trust. Developers and IT teams that manage these workloads want their web applications and infrastructure, including the IIS server and Windows Server, to remain compliant with their baseline policies.

This pattern covers manually exporting existing SSL certificates from Microsoft IIS, converting them from Personal Information Exchange (PFX) format to the Private Enhanced Mail (PEM) format that ACM supports, and then importing them into ACM in your AWS account. It also describes how to create an Application Load Balancer for your application and configure the Application Load Balancer to use your imported certificates. HTTPS connections are then terminated on the Application Load Balancer, and you don't need further configuration overhead on the web server. For more information, see [Create an HTTPS listener for your Application Load Balancer](#).

Windows servers use .pfx or .p12 files to contain the public key file (SSL certificate) and its unique private key file. The Certificate Authority (CA) provides you with your public key file. You use your server to generate the associated private key file where the certificate signing request (CSR) was created.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC) on AWS with at least one private and one public subnet in each Availability Zone used by your targets
- IIS version 8.0 or later running on Windows Server 2012 or later
- A web application running on IIS

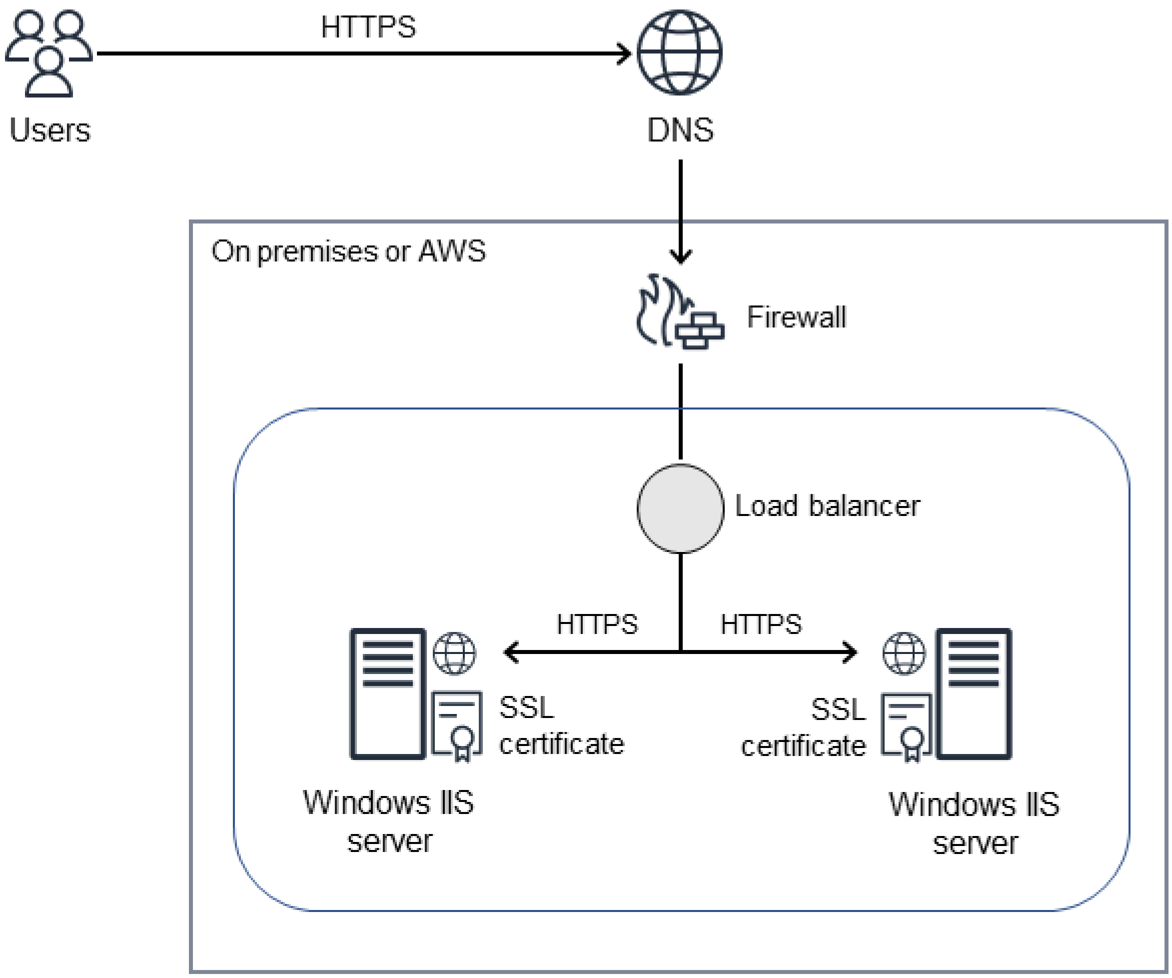
- Administrator access to the IIS server

Architecture

Source technology stack

- IIS web server implementation with SSL to ensure that data is transmitted securely in an encrypted connection (HTTPS)

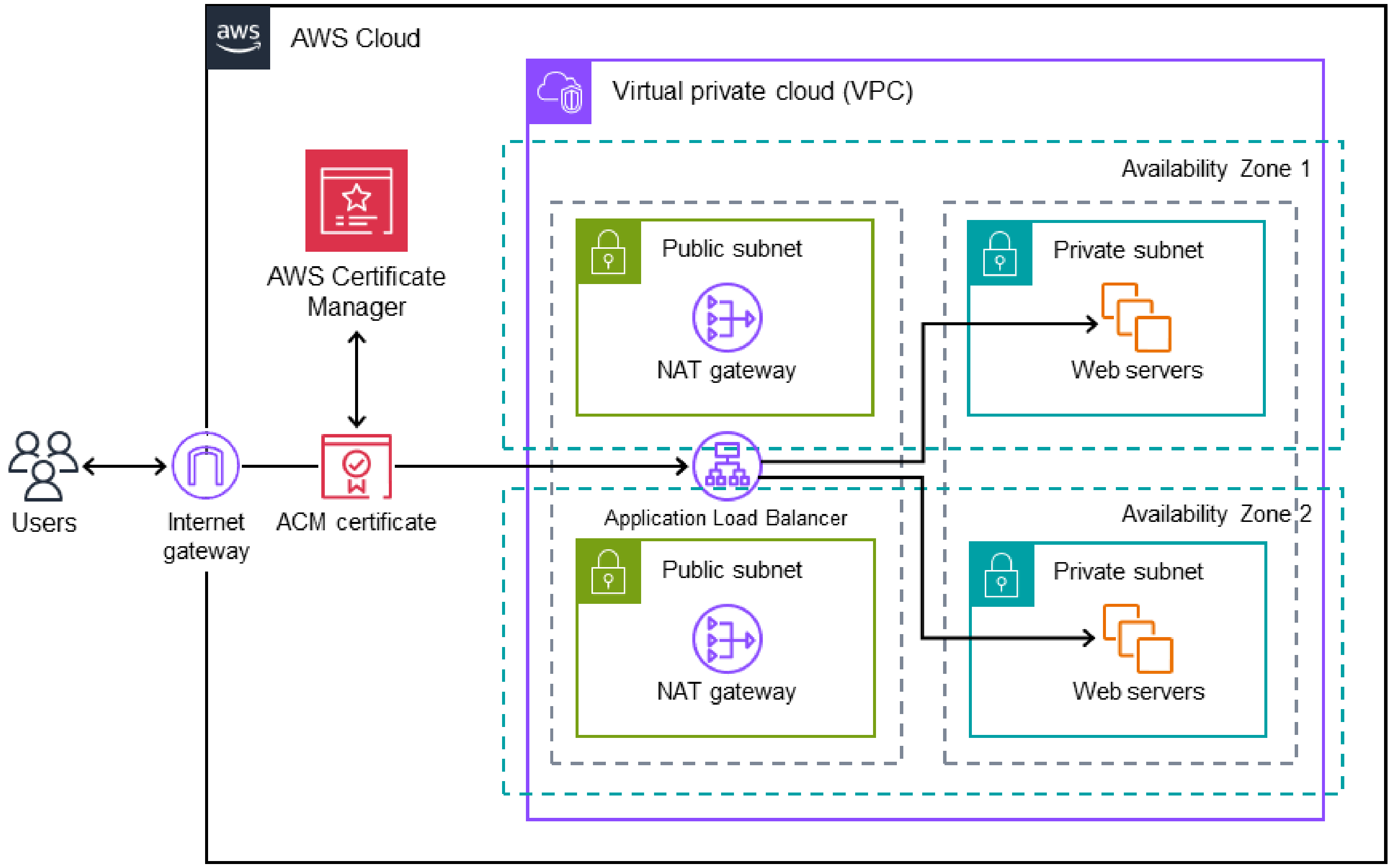
Source architecture



Target technology stack

- ACM certificates in your AWS account
- An Application Load Balancer configured to use imported certificates
- Windows Server instances in the private subnets

Target architecture



Tools

- AWS Certificate Manager (ACM) helps you create, store, and renew public and private SSL/TLS X.509 certificates and keys that protect your AWS websites and applications.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across EC2 instances, containers, and IP addresses in one or more Availability Zones.

Best practices

- Enforce traffic redirects from HTTP to HTTPS.
- Configure security groups for your Application Load Balancer properly to allow inbound traffic only to specific ports.
- Launch your EC2 instances in different Availability Zones to ensure high availability.

- Configure your application's domain to point to the Application Load Balancer's DNS name instead of its IP address.
- Make sure that the Application Load Balancer has application-layer health checks configured.
- Configure the threshold for health checks.
- Use Amazon CloudWatch to monitor the Application Load Balancer.

Epics

Export a .pfx file

Task	Description	Skills required
Export the .pfx file from Windows Server.	To export the SSL certificate as a .pfx file from the on-premises IIS manager in Windows Server: 1. Choose **Start**, **Administrative**, **Internet Information Services (IIS) Manager**. 2. Select the server name, and under **Security**, double-click **Server Certificates**. 3. Choose the certificate that you want to export, and then choose **Export**. 4. In the **Export Certificate** box, choose a location, path, and name for your .pfx file. 5. Specify and confirm a password for your .pfx file.	Systems administrator

Task	Description	Skills required
	Note You need this password when you install the .pfx file. 6. Choose OK. Your .pfx file should now be saved to the location and path you specified.	

Convert the PFX-encoded certificate to PEM format

Task	Description	Skills required
Download and install the OpenSSL toolkit.	1. Download and install [Win32/Win64 OpenSSL](https://slproweb.com/products/Win32OpenSSL.html) from the Shining Light Productions website. 2. Add the location of the OpenSSL binaries to your system PATH variable, so the binaries can be available for command-line use.	Systems administrator
Convert the PFX-encoded certificate to PEM format.	The following steps convert the PFX-encoded, signed certificate file into three files in PEM format:	Systems administrator

Task	Description	Skills required
	• `cert-file.pem` contains the SSL/TLS certificate for the resource. • `privatekey.pem` contains the private key of the certificate with no password protection. • `ca-chain.pem` contains the root certificate of the CA. To convert the PFX encoded certificate: 1. Run Windows PowerShell. 2. Use the following command to extract the private key of the certificate from the PFX file. Enter the certificate password when prompted. `openssl pkcs12 -in <filename>.pfx -nocerts -out withpw-privatekey.pem` The command generates a PEM-encoded private key file named `privatekey.pem` . Enter a passphrase to protect the private key file when prompted.	

Task	Description	Skills required

	3. Run the following command to remove the passphrase. When prompted, provide the passphrase that you created in step 2.
	`openssl rsa -in withpw-privatekey.pem -out privatekey.pem`
	If the command is successful, it displays the message "writing RSA key."
	4. Use the following command to transfer the certificate from the PFX file to a PEM file.
	`openssl pkcs12 -in <file_name>.pfx -clcerts -nokeys -out cert-file.pem`
	This creates a PEM-encoded certificate file named `cert-file.pem` . If the command is successful, it displays the message "MAC verified OK."
	5. Create a CA chain file from the PFX file. The following command creates a CA	

Task	Description	Skills required
	chain file named `ca-chain.pem` . ```openssl pkcs12 -in <file_name>.pfx -cacerts -nokeys -chain -out ca-chain.pem``` If the command is successful, it displays the message "MAC verified OK."	

Import a certificate into ACM

Task	Description	Skills required
Prepare to import the certificate.	On the ACM console, choose **Import a certificate**.	Cloud administrator
Provide the certificate body.	For **Certificate body**, paste the PEM-encoded certificate that you want to import. For more information about the commands and steps described in this and other tasks in this epic, see Importing a certificate in the ACM documentation.	Cloud administrator
Provide the certificate private key.	For **Certificate private key**, paste the PEM-encoded, unencrypted private key	Cloud administrator

Task	Description	Skills required
	that matches the certificate's public key.	
Provide the certificate chain.	For **Certificate chain**, paste the PEM-encoded certificate chain, which is stored in the `CertificateChain.pem` file.	Cloud administrator
Import the certificate.	Choose **Review and import**. Confirm that the informati on about your certificate is correct, and then choose **Import**.	Cloud administrator

Create an Application Load Balancer

Task	Description	Skills required
Create and configure the load balancer and listeners.	Follow the instructions in the Elastic Load Balancing documentation to configure a target group, register targets, and create an Application Load Balancer and listener. Add a second listener (HTTPS) for port 443.	Cloud administrator

Troubleshooting

Issue	Solution
Windows PowerShell doesn't recognize the OpenSSL command even after you add it to the system path.	Check `$env:path` to make sure that it includes the location of the OpenSSL binaries. If it doesn’t, run the following command in PowerShell: `$env:path = $env:path + ";C:\OpenSSL-Win64\bin"`

Related resources

Importing a certificate into ACM

- ACM console
- Certificate and key format for importing
- Importing a certificate
- AWS Certificate Manager User Guide

Creating an Application Load Balancer

- Create an Application Load Balancer
- Application Load Balancer User Guide

Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS

Nisha Gambhir, Amazon Web Services

Summary

This pattern describes how to migrate a .NET Framework or .NET Core web or console application from using the Microsoft Azure Service Bus queue messaging platform to Amazon Simple Queue Service (Amazon SQS).

Applications use messaging services to send data to, and receive data from, other applications. These services help build decoupled, highly scalable microservices, distributed systems, and serverless applications in the cloud.

Azure Service Bus queues are part of a broader Azure messaging infrastructure that supports queuing and publish/subscribe messaging.

Amazon SQS is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. Amazon SQS eliminates the complexity and overhead associated with managing and operating message-oriented middleware, and enables developers to focus on differentiating work. Using Amazon SQS, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A .NET Framework or .NET Core web or console application that uses Azure Service Bus queues (sample code attached)

Product versions

- .NET Framework 3.5 or later, or .NET Core 1.0.1, 2.0.0, or later

Architecture

Source technology stack

- A .NET (Core or Framework) web or console application that uses an Azure Service Bus queue to send messages

Target technology stack

- Amazon SQS

Tools

Tools

- Microsoft Visual Studio

Code

To create an AWS Identity and Access management (IAM) policy for Amazon SQS:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane on the left, choose **Policies**, and then choose **Create policy**.

3. Choose the **JSON** tab, and paste the following code:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
            "sqs:DeleteMessage",
            "sqs:GetQueueUrl",
            "sqs:ChangeMessageVisibility",
            "sqs:SendMessageBatch",
            "sqs:ReceiveMessage",
            "sqs:SendMessage",
            "sqs:GetQueueAttributes",
            "sqs:ListQueueTags",
            "sqs:ListDeadLetterSourceQueues",
            "sqs:DeleteMessageBatch",
```

```
                "sqs:PurgeQueue",
                "sqs:DeleteQueue",
                "sqs:CreateQueue",
                "sqs:ChangeMessageVisibilityBatch",
                "sqs:SetQueueAttributes"
            ],
            "Resource": "arn:aws:sqs:*:<AccountId>:*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "sqs:ListQueues",
            "Resource": "*"
        }
    ]
}
```

4. Choose **Review policy**, type a name, and then choose **Create policy**.

5. Attach the newly created policy to your existing IAM role or create a new role.

Epics

Set up Amazon SQS in AWS

Task	Description	Skills required
Create an IAM policy for Amazon SQS.	Create the IAM policy that will provide access to Amazon SQS. See the Code section for a sample policy.	System engineer
Create an AWS profile.	Create a new profile by running the AWS Tools for PowerShell command Set-AWSCredential. This command stores your access key and secret key in your default credentials file under the profile name you specify. Link the Amazon SQS policy	System engineer

Task	Description	Skills required
	you created earlier with this account. Keep the AWS access key ID and secret access key. These will be required in the next steps.	
Create an SQS queue.	You can create a standard queue or a first in, first out (FIFO) queue. For instructions, see the link in the References section.	System engineer

Revise your .NET application code

Task	Description	Skills required
Install AWS Toolkit for Visual Studio.	This toolkit is an extension for Microsoft Visual Studio and makes it easier for you to build and deploy .NET applications in AWS. For installation and usage instructions, see the link in the References section.	Application developer
Install the AWSSDK.SQS NuGet package.	You can install AWSSDK.SQS by choosing "Manage NuGet Package" in Visual Studio or by running the command "Install-Package AWSSDK.SQ S".	Application developer
Create an AWSCredentials object in your .NET applicati on.	The sample application in the attachment shows how to create a BasicAWSCredential	Application developer

Task	Description	Skills required
	s object, which inherits from AWSCredentials. You can use the access key ID and secret access key from earlier, or let the object pick these from the .aws folder as part of the user profile at run time.	
Create an SQS client object.	Create an SQS client object (AmazonSQSClient) for .NET Framework. This is part of the Amazon.SQS namespace. This object is required instead of IQueueClient, which is part of the Microsoft.Azure.Se rviceBus namespace.	Application developer
Call the SendMessageAsync method to send messages to the SQS queue.	Change the code that sends the message to the queue to use the amazonSqsClient.Se ndMessageAsync method. For details, see the attached code sample.	Application developer
Call the ReceiveMessageAsyn c method to receive messages from the SQS queue.	Change the code that receives the message to use the amazonSqsClient.ReceiveMess ageAsync method. For details, see the attached code sample.	Application developer

Task	Description	Skills required
Call the DeleteMessageAsync method to delete messages from the SQS queue.	To delete messages, change the code from the queueClie nt.CompleteAsync method to the amazonSqsClient.De leteMessageAsync method. For details, see the attached code sample.	Application developer

Related resources

- AWS SDK for .NET Developer Guide
- Messaging Using Amazon SQS
- Creating and Using an Amazon SQS Queue with the AWS SDK for .NET
- Send an Amazon SQS Message
- Receive a Message from an Amazon SQS Queue
- Delete a Message from an Amazon SQS Queue
- AWS Toolkit for Visual Studio

Additional information

This pattern includes two sample applications (see the attachments section):

- **AzureSbTestApp** includes code that uses the Azure Service Bus queue.
- **AmazonSqsTestApp** uses Amazon SQS. This is a console application that uses .NET Core 2.2 and includes examples for sending and receiving messages.

Notes:

- queueClient is an object of IQueueClient, which is part of the Microsoft.Azure.ServiceBus namespace (included in the Microsoft.Azure.ServiceBus NuGet package).
- amazonSqsClient is an object of AmazonSQSClient, which is part of the Amazon.SQS namespace (included in the AWSSDK.SQS NuGet package).

- Depending upon where the code is running, say if its running on EC2, the role needs to have permission to write into the SQS Queue.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Migrate a relational database to MongoDB Atlas on AWS

Battulga Purevragchaa and Igor Alekseev, Amazon Web Services

Babu Srinivasan, MongoDB

Summary

This pattern describes the steps for migrating from a relational database such as SQL Server, MySQL, or PostgreSQL to MongoDB Atlas in the AWS Cloud. It uses MongoDB Relational Migrator to help accelerate the data migration from relational databases to MongoDB Atlas.

The pattern accompanies the guide Migrating to MongoDB Atlas on AWS on the AWS Prescriptive Guidance website. It provides the implementation steps for one of the migration scenarios that are discussed in that guide. For additional migration scenarios, see the following patterns on the AWS Prescriptive Guidance website:

- Migrate a self-hosted MongoDB environment to MongoDB Atlas on AWS
- Stream data from IBM Db2, SAP, Sybase, and other databases to MongoDB Atlas on AWS

The pattern is intended for AWS System Integrator (SI) Partners and AWS users.

Prerequisites and limitations

Prerequisites

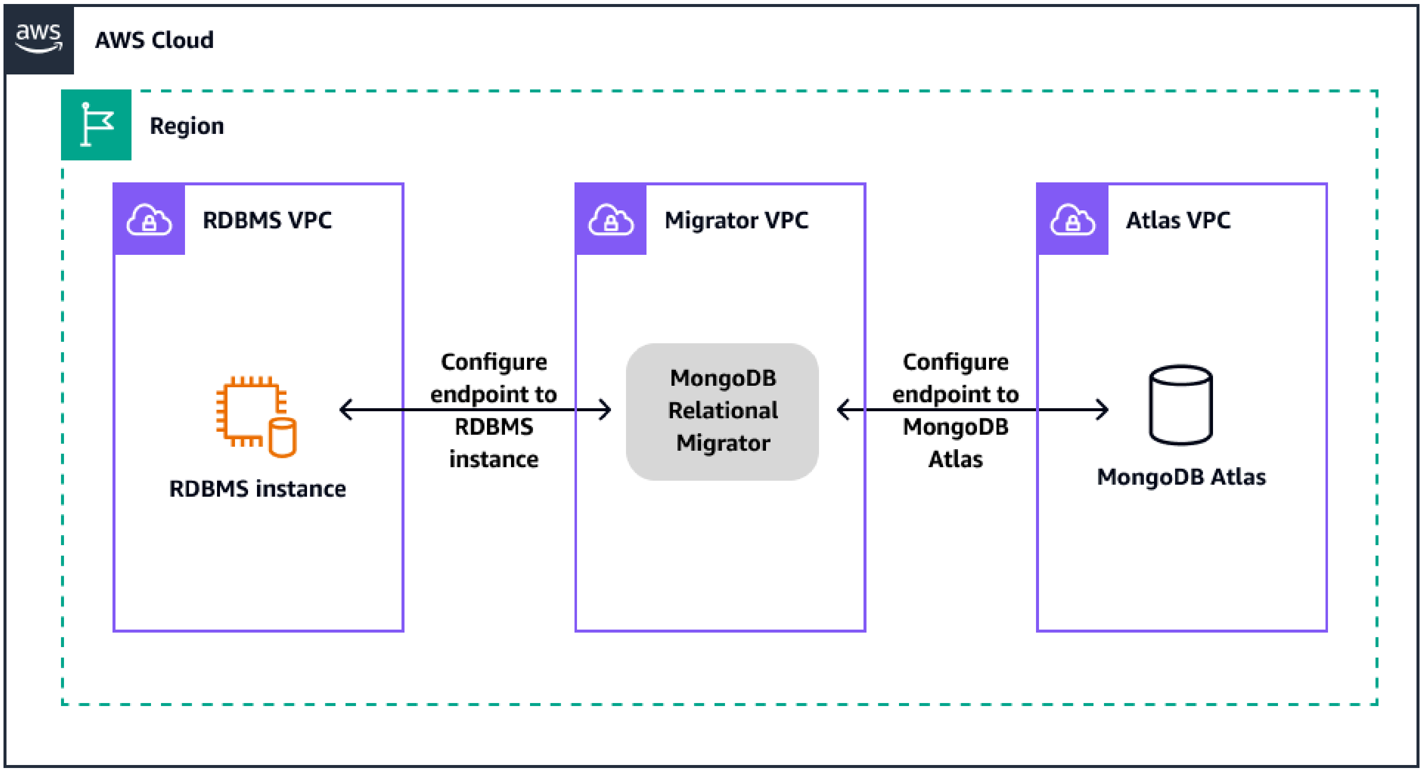
- A source relational database (Oracle Database, SQL Server, PostgreSQL, MySQL, SAP/Sybase ASE, and others) to migrate to MongoDB Atlas.
- Familiarity with relational databases, MongoDB Atlas, and AWS services. This pattern explains some of the migration steps at a high level. Additional details will be added in future versions.

Product versions

- MongoDB version 5.0 or later

Architecture

The following diagram shows the migration from a relational database management system (RDBMS) database to MongoDB Atlas on AWS.



For MongoDB Atlas reference architectures that support different usage scenarios, see [Migrating to MongoDB Atlas on AWS] on the AWS Prescriptive Guidance website.

Tools

- MongoDB Atlas is a fully managed database as a service (DBaaS) for deploying and managing MongoDB databases in the cloud.
- MongoDB Relational Migrator provides a smooth transition of data from traditional relational databases to MongoDB. It helps automate the conversion process and transforms the structured data model of relational databases into the flexible document format provided by MongoDB. The Relational Migrator preserves data integrity and relationships to simplify the migration. Organizations can take advantage of the scalability, performance, and versatility benefits that MongoDB provides while retaining the familiarity of their existing data.

Best practices

For best practices for using MongoDB on AWS, see the posts on the AWS Partner Network Blog.

Epics

Discovery and assessment

Task	Description	Skills required
Determine the parameters and size of the relational database.	Estimate the working set size by using the Relational Migrator recommendations and information from `db.stats()` for the total index space. Assume that a percentage of your data space will be accessed frequently. This task should take approximately one week. For more information and examples for this and the other stories in this epic, see the Related resources section.	App owner, DBA
Estimate network bandwidth requirements.	To estimate your network bandwidth requirements, multiply the average document size by the number of documents served per second. Consider the maximum traffic that any node on your cluster will bear as the basis. To calculate downstream data transfer rates from your cluster to client applications, use the sum of the total documents returned over a period of time. If your applications read from secondary nodes,	DBA

Task	Description	Skills required
	divide this number of total documents by the number of nodes that can serve read operations. To find the average document size for a database, use the `db.stats().avgObjSize` command. This task will typically take one day.	
Select the Atlas tier.	Follow the instructions in the [MongoDB documentation](#) to select the correct Atlas cluster tier.	DBA
Plan for cutover.	Plan for application cutover.	DBA, App owner

Set up a new MongoDB Atlas environment on AWS

Task	Description	Skills required
Create a new MongoDB Atlas cluster on AWS.	In MongoDB Atlas, choose **Build a Cluster**. In the **Create New Cluster** dialog box, select AWS as the cloud provider.	DBA
Select AWS Regions and global cluster configuration.	Select from the list of available AWS Regions for your Atlas cluster. Configure global clusters if required.	DBA
Select the cluster tier.	Select your preferred cluster tier. Your tier selection determines factors such as	DBA

Task	Description	Skills required
	memory, storage, and IOPS specification.	
Configure additional cluster settings.	Configure additional cluster settings such as MongoDB version, backup, and encryption options. For more information about these options, see the Related resources section.	DBA

Configure security and compliance

Task	Description	Skills required
Configure the access list.	To connect to the Atlas cluster, you must add an entry to the project's access list. Atlas uses TLS/SSL to encrypt the connections to the virtual private cloud (VPC) for your database. To set up the access list for the project and for more information about the stories in this epic, see the Related resources section.	DBA
Authenticate and authorize users.	You must create and authenticate the database users who will access the MongoDB Atlas clusters. To access clusters in a project, users must belong to that	DBA

Task	Description	Skills required
	project, and they can belong to multiple projects.	
Create custom roles.	(Optional) Atlas supports creating custom roles in cases where the built-in Atlas database user permissio ns don't cover the set of permissions you want.	DBA
Set up VPC peering.	(Optional) Atlas supports [VPC peering] with other VPCs on AWS.	AWS administrator
Set up an AWS PrivateLink endpoint.	(Optional) You can set up private endpoints on AWS by using AWS PrivateLink. For more information, see the [Amazon VPC documentation].	AWS administrator
Enable two-factor authentic ation.	(Optional) Atlas supports two-factor authentication (2FA) to help users control access to their Atlas accounts.	AWS administrator
Set up user authentication and authorization with LDAP.	(Optional) Atlas supports performing user authentic ation and authorization with Lightweight Directory Access Protocol (LDAP).	DBA

Task	Description	Skills required
Set up unified AWS access.	(Optional) Some Atlas features, including Atlas Data Lake and encryptio n at rest using customer key management, use AWS Identity and Access Management (IAM) roles for authentication.	AWS administrator
Set up encryption at rest using AWS KMS.	(Optional) Atlas supports using AWS Key Management Service (AWS KMS) to encrypt storage engines and cloud provider backups.	AWS administrator
Set up client-side field-level encryption.	(Optional) Atlas supports client-side field level encryption, including automatic encryption of fields.	AWS administrator

Migrate data

Task	Description	Skills required
Add MongoDB Relational Migrator to the access list.	Add the Relational Migrator to the access list for your source database. This helps prepare the source environme nt to connect to the target Atlas cluster.	DBA
Assess the relational database objects.	Launch MongoDB Relational Migrator and connect to your	DBA

Task	Description	Skills required
	relational database. Start the assessment.	
Accept the migration patterns or choose to alter them based on your business needs.	Either accept the database patterns recommended by the Relational Migrator based on the initial assessment and performance parameters, or choose to alter them based on your business requireme nts.	DBA
Launch your target replica set in MongoDB Atlas.	Launch your target replica set in MongoDB Atlas. In Relational Migrator, choose **I'm ready to migrate**.	DBA

Configure operational integration

Task	Description	Skills required
Connect to the MongoDB Atlas cluster.	Make sure that MongoDB Atlas cluster connectivity works as expected.	App owner
Interact with cluster data.	Verify the cluster data.	DBA
Monitor your clusters.	Verify that your clusters are set up correctly.	DBA
Back up and restore cluster data.	Schedule backups with a regular cadence for cluster data.	DBA

Related resources

All of the following links, unless noted otherwise, go to webpages in the MongoDB documentation.

Migration guide

- Migrating to MongoDB Atlas on AWS (AWS Prescriptive Guidance)

Discovery and assessment

- Memory
- Sizing example with Atlas sample data sets
- Sizing example for mobile applications
- Network Traffic
- Cluster Auto-Scaling
- Atlas sizing template

Configuring security and compliance

- Configure IP Access List Entries
- Configure Database Users
- Configure Access to the Atlas UI
- Configure Custom Database Roles
- Configure Database Users
- Set up a Network Peering Connection
- Learn About Private Endpoints in Atlas
- Manage Your Multi-Factor Authentication Options
- Set up User Authentication and Authorization with LDAP
- Atlas Data Lake
- Encryption at Rest using Customer Key Management
- Methods to assume a role (IAM documentation)
- Client-Side Field Level Encryption
- Automatic Encryption
- MongoDB Atlas Security Controls

- MongoDB Trust Center
- Configure Security Features for Clusters

Setting up a new MongoDB Atlas environment on AWS

- Cloud Providers and Regions
- Manage Global Clusters
- Select Cluster Tier
- Configure Additional Settings
- Get Started with Atlas
- Configure Access to the Atlas UI
- Manage Clusters

Migrating data

- Migrate or Import Data

Monitoring clusters

- Monitor Your Clusters

Integrating operations

- Connect to a Cluster
- Interact with Your Data
- Monitor Your Clusters
- Back Up, Restore, and Archive Data

Blog posts

- Modernizing RDBMS Schemas With a MongoDB Document Model

Migrate a self-hosted MongoDB environment to MongoDB Atlas on AWS

Battulga Purevragchaa and Igor Alekseev, Amazon Web Services

Babu Srinivasan, MongoDB

Summary

This pattern describes the steps for migrating from a self-managed MongoDB environment (including MongoDB Community Server, Enterprise Server, Enterprise Advanced, mLab, or any managed MongoDB cluster) to MongoDB Atlas in the AWS Cloud. It uses the Atlas Live Migration Service to help accelerate the data migration from MongoDB to MongoDB Atlas.

The pattern accompanies the guide Migrating to MongoDB Atlas on AWS on the AWS Prescriptive Guidance website. It provides the implementation steps for one of the migration scenarios that are discussed in that guide. For additional migration scenarios, see the following patterns on the AWS Prescriptive Guidance website:

- Migrate a relational database to MongoDB Atlas on AWS
- Stream data from IBM Db2, SAP, Sybase, and other databases to MongoDB Atlas on AWS

The pattern is intended for AWS Systems Integrator (SI) Partners and AWS users.

Prerequisites and limitations

Prerequisites

- A source MongoDB Enterprise Advanced, Community Server, or other self-managed MongoDB environment to migrate to MongoDB Atlas.
- Familiarity with MongoDB, MongoDB Atlas, and AWS services. This pattern explains some of the migration steps at a high level. Additional details will be added in future versions.

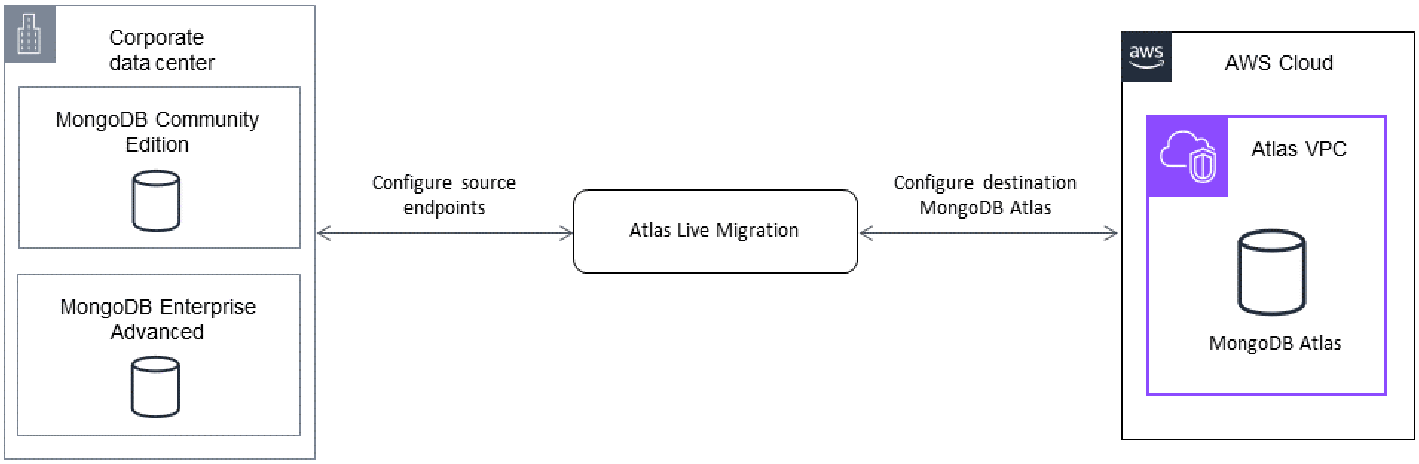
Product versions

- MongoDB version 6.0.13 or later

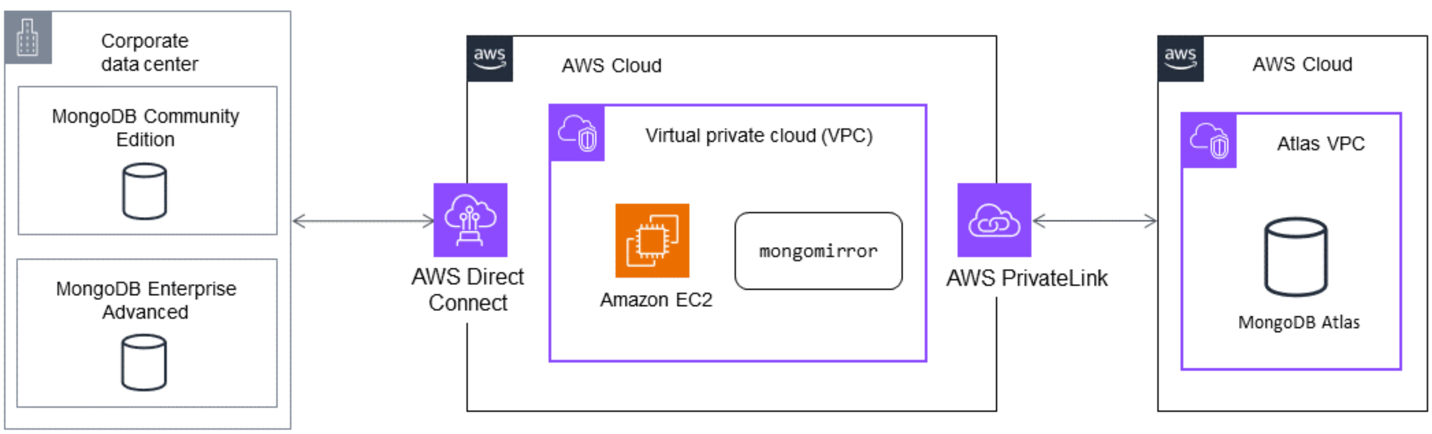
Architecture

The following diagram shows the Atlas Live Migration Service that is used to migrate data from a MongoDB Enterprise Advanced database and a MongoDB Community database to MongoDB

Atlas on AWS. Use this service when you need to migrate large, complex databases to MongoDB Atlas with minimal downtime and continuous data synchronization. This pattern uses Atlas Live Migration Service.



The following diagram shows the MongoDB mirror service (`mongomirror`) that you can also use to migrate data from a MongoDB Enterprise Advanced database and a MongoDB Community database to MongoDB Atlas on AWS over a secured [AWS PrivateLink](https://aws.amazon.com/privatelink/) connection. Use `mongomirror` for ongoing data replication between on-premises MongoDB and MongoDB Atlas. This tool is ideal for disaster recovery or phased migrations but is outside the scope of this pattern.



For more MongoDB Atlas reference architectures that support different usage scenarios, see Migrating to MongoDB Atlas on AWS on the AWS Prescriptive Guidance website.

Tools

- MongoDB Atlas is a fully managed database as a service (DbaaS) for deploying and managing MongoDB databases in the cloud.

- Atlas Live Migration Service is a free MongoDB utility that helps migrate databases to Atlas. This service keeps the source database in sync with the destination database until cutover. When you're ready to cut over, you stop your application instances, point them to the destination Atlas cluster, and restart them. To access this service, choose **Database options** from your MongoDB Atlas cluster.
- mongomirror is a tool for manually migrating data from an existing MongoDB replica set to a MongoDB Atlas replica set. `mongomirror` doesn't require you to shut down your existing replica set or applications, doesn't import user or role data, or copy the configuration database. You can download `mongomirror` from the MongoDB documentation.

Best practices

For best practices for using MongoDB on AWS, see the posts on the AWS Partner Network Blog.

Epics

Discovery and assessment

Task	Description	Skills required
Determine the cluster size.	Estimate the working set size by using the information from `db.stats()` for the total index space. Assume that a percentage of your data space will be accessed frequentl y. Or, you can estimate your memory requirements based on your own assumptio ns. This task should take approximately one week. For more information and examples for this and the other stories in this epic, see the Related resources section.	DBA, App owner
Estimate network bandwidth requirements.	To estimate your network bandwidth requireme	DBA

Task	Description	Skills required
	nts, multiply the average document size by the number of documents served per second. Consider the maximum traffic that any node on your cluster will bear as the basis. To calculate downstream data transfer rates from your cluster to client applications, use the sum of the total documents returned over a period of time. If your applications read from secondary nodes, divide this number of total documents by the number of nodes that can serve read operations. To find the average document size for a database, use the `db.stats().avgObjSize` command. This task will typically take one day.	
Select the Atlas tier.	Follow the instructions in the MongoDB documentation to select the correct Atlas cluster tier.	DBA
Plan for cutover.	Plan for application cutover.	DBA, App owner

Set up a new MongoDB Atlas environment on AWS

Task	Description	Skills required
Create a new MongoDB Atlas cluster on AWS.	Log in to Atlas and open the **Overview** page for your project. Choose the **Create** button to create a cluster. For more information, see the MongoDB documentation.	DBA
Select AWS Regions and global cluster configuration.	Select from the list of available AWS Regions for your Atlas cluster. Configure global clusters if required. For more information, see the MongoDB documentation.	DBA
Select the cluster tier.	Select your preferred cluster tier. Your tier selection determines factors such as memory, storage, and IOPS specification.	DBA
Configure additional cluster settings.	Configure additional cluster settings such as MongoDB version, backup, and encryption options. For more information about these options, see the Related resources section.	DBA

Configure security and compliance

Task	Description	Skills required
Authenticate and authorize users.	You must create and authenticate the database users who will access the MongoDB Atlas clusters. To access clusters in a project, users must belong to that project, and they can belong to multiple projects. Atlas also supports authentication based on AWS Identity and Access Management (IAM). For more information, see the MongoDB documentation.	DBA
Create custom roles.	(Optional) Atlas supports creating custom roles in cases where the built-in Atlas database user permissio ns don't cover the set of permissions you want.	DBA
Set up VPC peering.	(Optional) Atlas supports virtual private cloud (VPC) peering with other VPCs on AWS.	AWS administrator
Set up an AWS PrivateLink endpoint.	(Optional) You can set up private endpoints on AWS by using AWS PrivateLink. For more information, see the Amazon VPC documentation.	AWS administrator
Enable two-factor authentic ation.	(Optional) Atlas supports two-factor authentication	AWS administrator

Task	Description	Skills required
	(2FA) to help users control access to their Atlas accounts.	
Set up user authentication and authorization with LDAP.	(Optional) Atlas supports performing user authentic ation and authorization with Lightweight Directory Access Protocol (LDAP).	AWS administrator
Set up unified AWS access.	(Optional) Some Atlas features, including Atlas Data Lake and encryption at rest using customer key management, use IAM roles for authentication.	AWS administrator
Set up encryption at rest using AWS KMS.	(Optional) Atlas supports using AWS Key Management Service (AWS KMS) to encrypt storage engines and cloud provider backups.	AWS administrator
Set up client-side field-level encryption.	(Optional) Atlas supports client-side field level encryption, including automatic encryption of fields.	AWS administrator

Migrate data

Task	Description	Skills required
Select your target replica set in MongoDB Atlas.	Navigate to the destinati on Atlas cluster and choose the ellipsis (...) button. On	DBA

Task	Description	Skills required
	the cluster list, this button appears below the cluster name. In cluster details, the button appears on the right, next to the **Connect** and **Configuration** buttons. For more information, see the [MongoDB documentation]().	
Add the Atlas Live Migration Service to the access list.	Add the Atlas Live Migration Service to the access list in your AWS source cluster. This helps prepare the source environment to connect to the target Atlas cluster.	DBA
Perform migration with Atlas Live Migration Service.	Choose **Start migration**. When the **Prepare to Cutover** button turns green, perform the cutover. Review Atlas cluster performance metrics. Consider updating the database connection in all application layers to point to the new database.	DBA

Configure operational integration

Task	Description	Skills required
Connect to the MongoDB Atlas cluster.	Make sure that MongoDB Atlas cluster connectivity works as expected.	App owner
Interact with cluster data.	Test the cluster data.	DBA

Task	Description	Skills required
Monitor your clusters.	Verify that your clusters are set up correctly.	DBA
Back up and restore cluster data.	Schedule backups with a regular cadence for cluster data.	DBA

Troubleshooting

Issue	Solution

Error: Could not reach specified source	• Make sure that you added the correct subnet ranges to the IP access list on the source cluster. You can find the four required subnet ranges in the Live Migration modal window.
	• Confirm that the hostname that you specified resolves to a public IP address. At a command prompt, use one of the following commands:
	`nslookup <hostname>`
	`ping <hostname>`
	• Make sure that you are not using a [VPC peering connection](), which isn't compatible with pull live migration. If a VPC peering connection is your only option, use `mongomirror` instead.

Error: Could not resolve hostname	No IP address was found for the given hostname. Confirm that the given hostname is correct and publicly accessible.

Issue	Solution
Any other error	If you encounter any other errors, see Troubleshoot Live Migration (Pull) in the MongoDB documentation.

Related resources

All of the following links, unless noted otherwise, go to webpages in the MongoDB documentation.

Migration guide

- Migrating to MongoDB Atlas on AWS (AWS Prescriptive Guidance)

Legacy migration

- Migration of older versions of MongoDB

Discovery and assessment

- Memory
- Sizing example with Atlas sample data sets
- Sizing example for mobile applications
- Network Traffic
- Cluster Auto-Scaling
- Atlas sizing template

Configuring security and compliance

- Configure IP Access List Entries
- Configure Database Users
- Configure Access to the Atlas UI
- Configure Custom Database Roles
- Configure Database Users
- Set up a Network Peering Connection

- Learn About Private Endpoints in Atlas
- Manage Your Multi-Factor Authentication Options
- Set up User Authentication and Authorization with LDAP
- Atlas Data Lake
- Encryption at Rest using Customer Key Management
- Methods to assume a role (IAM documentation)
- Client-Side Field Level Encryption
- Automatic Encryption
- MongoDB Atlas Security Controls
- MongoDB Trust Center
- Configure Security Features for Clusters

Setting up a new MongoDB Atlas environment on AWS

- Cloud Providers and Regions
- Manage Global Clusters
- Select Cluster Tier
- Configure Additional Settings
- Get Started with Atlas
- Configure Access to the Atlas UI
- Manage Clusters

Migrating data

- Migrate or Import Data

Monitoring clusters

- Monitor Your Clusters

Integrating operations

- Connect to a Cluster

- Interact with Your Data
- Monitor Your Clusters
- Back Up, Restore, and Archive Data

Training

- Live Migration with MongoDB Atlas

Additional information

For additional information, see the following topics in the MongoDB documentation:

- To move data to a serverless instance, use Compass to export and import data, or migrate data with self-managed tools. To learn more, see Serverless Instance Limitations.
- To load data into a new cluster in Atlas, see Load Data into Atlas.
- To make a copy of your cluster for testing purposes, see Backup Methods for a Self-Managed Deployment.
- If the application that you want to migrate requires a near-continuous uptime, contact MongoDB Support and share your uptime requirements and cluster configuration.
- For additional information, see Migrate or Import Data.

Migrate an Oracle JD Edwards EnterpriseOne database to AWS by using Oracle Data Pump and AWS DMS

Thanigaivel Thirumalai, Amazon Web Services

Summary

You can migrate and run your JD Edwards EnterpriseOne database on Amazon Relational Database Service (Amazon RDS). When you migrate your database to Amazon RDS, AWS can take care of backup tasks and high availability setup, so you can concentrate on maintaining your EnterpriseOne application and its functionality. For a comprehensive list of key factors to consider during the migration process, see Oracle database migration strategies in AWS Prescriptive Guidance.

There are multiple ways to migrate an EnterpriseOne database, including:

- Using Oracle Universal Batch Engine (UBE) R98403 for schema and table creation, and using AWS Database Migration Service (AWS DMS) for migration
- Using DB native tools for schema and table creation and using AWS DMS for migration
- Using DB native tools for the migration of existing data (full load) and using AWS DMS for change data capture (CDC) tasks

This pattern covers the third option. It explains how to migrate your on-premises EnterpriseOne databases to Amazon RDS for Oracle by using Oracle Data Pump with AWS DMS and its CDC feature.

Oracle JD Edwards EnterpriseOne is an enterprise resource planning (ERP) solution for organizations that manufacture, construct, distribute, service, or manage products or physical assets. JD Edwards EnterpriseOne supports various hardware, operating systems, and database platforms.

When you migrate critical ERP applications such as JD Edwards EnterpriseOne, minimizing downtime is key. AWS DMS minimizes downtime by supporting both full load and continuous replication from the source database to the target database. AWS DMS also provides real-time monitoring and logging for the migration, which can help you identify and resolve any issues that could cause downtime.

When you replicate changes with AWS DMS, you must specify a time or system change number (SCN) as the starting point for reading changes from the database logs. It's crucial to keep these

logs accessible on the server for a designated amount of time (we recommend 15 days) to ensure that AWS DMS has access to these changes.

Prerequisites and limitations

Prerequisites

- An Amazon RDS for Oracle database provisioned in your AWS Cloud environment as the target database. For instructions, see the Amazon RDS documentation.
- An EnterpriseOne database that's running on premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance on AWS.

> **Note**
>
> This pattern is designed for migrating from on premises to AWS, but it was tested by using an EnterpriseOne database on an EC2 instance. If you plan to migrate from your on-premises environment, you must configure the appropriate network connectivity.

- Schema details. Identify which Oracle database schema (for example, DV920) you plan to migrate for EnterpriseOne. Before you start the migration process, gather the following details about the schema:
  - Schema size
  - The number of objects per object type
  - The number of invalid objects

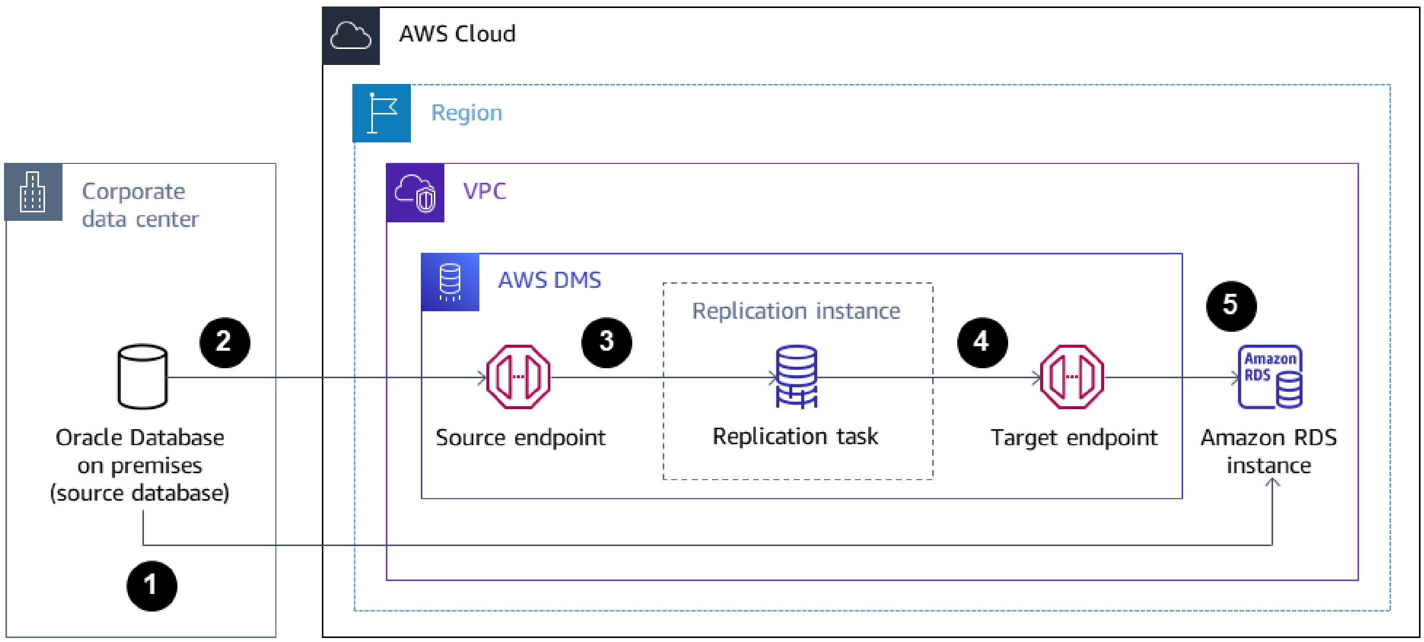
Limitations

- You have to create any schemas you want on the target Amazon RDS for Oracle database—AWS DMS doesn't create these for you. (The Epics section describes how to use Data Pump to export and import schemas.) The schema name must already exist for the target Oracle database. Tables from the source schema are imported to the user or the schema, and AWS DMS uses the administrator or system account to connect to the target instance. To migrate multiple schemas, you can create multiple replication tasks. You can also migrate data to different schemas on a target instance. To do this, use schema transformation rules on the AWS DMS table mappings.
- This pattern has been tested with a demo dataset. We recommend that you validate compatibility for your dataset and customization.

- This pattern uses an EnterpriseOne database that's running on Microsoft Windows. However, you can use the same process with other operating systems that are supported by AWS DMS.

Architecture

The following diagram shows a system that's running EnterpriseOne on an Oracle database as the source database, and an Amazon RDS for Oracle database as the target database. The data is exported from the source Oracle database and imported into the target Amazon RDS for Oracle database by using Oracle Data Pump, and replicated for CDC updates by using AWS DMS.



1. Oracle Data Pump extracts data from the source database, and the data is sent to the Amazon RDS for Oracle database target.
2. CDC data is sent from the source database to a source endpoint in AWS DMS.
3. From the source endpoint, the data is sent to the AWS DMS replication instance, where the replication task is performed.
4. After the replication task is complete, the data is sent to the target endpoint in AWS DMS.
5. From the target endpoint, the data is sent to the Amazon RDS for Oracle database instance.

Tools

AWS services

- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Amazon Relational Database Service (Amazon RDS) for Oracle helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.

Other services

- Oracle Data Pump helps you move data and metadata from one database to another at high speed.

Best practices

Migrating LOBs

If your source database contains large binary objects (LOBs) that need to be migrated to the target database, AWS DMS provides the following options:

- **Full LOB mode** – AWS DMS migrates all the LOBs from the source to the target database regardless of their size. Although the migration is slower than the other modes, the advantage is that data isn't truncated. For better performance, you can create a separate task on the new replication instance to migrate the tables that have LOBs that are larger than a few megabytes.
- **Limited LOB mode** – You specify the maximum size of LOB column data, which allows AWS DMS to pre-allocate resources and apply the LOBs in bulk. If the size of the LOB columns exceeds the size that is specified in the task, AWS DMS truncates the data and sends warnings to the AWS DMS log file. You can improve performance by using limited LOB mode if your LOB data size is within the limited LOB size.
- **Inline LOB mode** – You can migrate LOBs without truncating the data or slowing the performance of your task by replicating both small and large LOBs. First, specify a value for the `InlineLobMaxSize` parameter, which is available only when full LOB mode is set to `true`. The AWS DMS task transfers the small LOBs inline, which is more efficient. Then, AWS DMS migrates the large LOBs by performing a lookup from the source table. However, inline LOB mode works only during the full load phase.

Generating sequence values

During the AWS DMS CDC process, incremental sequence numbers aren't replicated from the source database. To avoid discrepancies in sequence values, you must generate the most recent sequence value from the source for all sequences, and apply it to the target Amazon RDS for Oracle database.

AWS Secrets Manager

To help manage your credentials, we recommend that you follow the instructions in the blog post Manage your AWS DMS endpoint credentials with AWS Secrets Manager.

Performance

- **Replication instances** – For guidance on choosing the best instance size, see Selecting the best size for a replication instance in the AWS DMS documentation.
- **Connectivity options** – To avoid latency issues, we recommend that you choose the right connectivity option. AWS Direct Connect provides the shortest path to AWS resources, because it is a dedicated connection between your corporate data centers and AWS. While in transit, your network traffic remains on the AWS global network and never goes over the internet. This reduces the chance of hitting bottlenecks or unexpected increases in latency when compared with using VPN or the public internet.
- **Network bandwidth** – To optimize performance, verify that your network throughput is fast. If you are using a VPN tunnel between your on-premises source database and AWS DMS, ensure that the bandwidth is sufficient for your workload.
- **Task parallelism** – You can speed up data replication by loading multiple tables in parallel during full load. This pattern uses of RDBMS endpoints, so this option applies only to the full load process. Task parallelism is controlled by the `MaxFullLoadSubTasks` parameter, which determines how many full load sub-tasks are run in parallel. By default, this parameter is set to 8, which means that eight tables (if selected in table mapping) are loaded together during full mode. You can adjust this parameter in the full-load task settings section of the JSON script for the task.
- **Table parallelism** – AWS DMS also enables you to load a single large table by using multiple parallel threads. This is particularly useful for Oracle source tables that have billions of records as well as multiple partitions and subpartitions. If the source table isn't partitioned, you can use column boundaries for parallel loads.
- **Split loads** – When you split loads across multiple tasks or AWS DMS instances, remember transaction boundaries when you capture changes.

Epics

Use Oracle Data Pump to export the EnterpriseOne schema

Task	Description	Skills required

Generate the SCN.	When the source database is active and in use by the EnterpriseOne application, initiate the data export with Oracle Data Pump. You must first generate a system change number (SCN) from the source database for both data consistency during the export with Oracle Data Pump and as a starting point for CDC in AWS DMS.
	
	To generate the current SCN from your source database, use the following SQL statement:
	
	`SQL> select current_scn from v$database;`
	
	`CURRENT_SCN`
	`-----------`
	`   30009727`
	
	Save the generated SCN. You will use the SCN when you export the data and to create the AWS DMS replication task.
		DBA

Create the parameter file.	To create a parameter file for exporting the schema, you can use the following code.	DBA

Task	Description	Skills required
	``` directory=DMS_DATA _PUMP_DIR logfile=export_dms.log dumpfile=export_dms_d ata.dmp schemas=<schema name> flashback_scn=<SCN from previous command> ``` **Note** You can also define your own `DATA_PUMP_DIR` by using the following commands, based on your requirements. ``` SQL> CREATE OR REPLACE DIRECTORY DMS_DATA_ PUMP_DIR AS '<Directo ry for dump>'; Directory created. SQL> GRANT READ, WRITE ON DIRECTORY DMS_DATA_ PUMP_DIR TO SYSTEM; Grant succeeded. ```	

Task	Description	Skills required
Export the schema.	To perform the export, use the `expdp` utility as follows: ``` C:\Users\Administr ator>expdp system/ ********@<DB Name> PARFILE='<Path to PAR file create above>' Export: Release 19.0.0.0.0 - Productio n on *** *** ** **:**:** **** Version 19.3.0.0.0 Copyright (c) 1982, 2019, Oracle and/or its affiliates. All rights reserved. Connected to: Oracle Database 19c Standard Edition 2 Release 19.0.0.0.0 - Productio n Starting "SYSTEM". "SYS_EXPORT_SCHEMA _02": system/** ******@<DB Name>PARF ILE='E:\exp_dms_da tapump.par' Processing object type SCHEMA_EXPORT/TABLE/ TABLE_DATA Processing object type SCHEMA_EXPORT/TABL E/INDEX/STATISTICS/ INDEX_STATISTICS Processing object type SCHEMA_EXPORT/TABL	DBA

Task	Description	Skills required

	``
	E/STATISTICS/TABLE
	_STATISTICS
	Processing object type
	 SCHEMA_EXPORT/STAT
	ISTICS/MARKER
	Processing object type
	 SCHEMA_EXPORT/USER
	Processing object type
	 SCHEMA_EXPORT/ROLE
	_GRANT
	Processing object type
	 SCHEMA_EXPORT/DEFA
	ULT_ROLE
	Processing object type
	 SCHEMA_EXPORT/TABL
	ESPACE_QUOTA
	Processing object type
	 SCHEMA_EXPORT/PRE_
	SCHEMA/PROCACT_SCHEMA
	Processing object type
	 SCHEMA_EXPORT/TABLE/
	TABLE
	Processing object type
	 SCHEMA_EXPORT/TABL
	E/GRANT/OWNER_GRANT/
	OBJECT_GRANT
	Processing object type
	 SCHEMA_EXPORT/TABLE/
	INDEX/INDEX
	Processing object type
	 SCHEMA_EXPORT/TABLE/
	CONSTRAINT/CONSTRAINT
	. . exported "<Schema
	 Name>"."<Table Name>"
	
	   228.9 MB 496397 rows
	``` ```
	Master table "SYSTEM".
	"SYS_EXPORT_SCHEMA
	_02" successfully
	 loaded/unloaded
	``	

Task	Description	Skills required
	``` ******************* ***************** ***************** ***************** **** Dump file set for SYSTEM.SYS_EXPORT_ SCHEMA_02 is: E:\DMSDUMP\EXPORT_ DMS_DATA.DMP Job "SYSTEM"."SYS_EXPO RT_SCHEMA_02" successfully completed at *** *** ** **:**:** **** elapsed 0 00:01:57 ```	

Use Oracle Data Pump to import the EnterpriseOne schema

Task	Description	Skills required
Transfer the dump file to the target instance.	To transfer your files by using the `DBMS_FILE_TRANSFER` utility, you need to create a database link from the source database to the Amazon RDS for Oracle instance. After the link is established, you can use the utility to transfer the Data Pump files directly to the Amazon RDS instance. Alternatively, you can transfer the Data Pump files to [Amazon Simple Storage Service (Amazon S3)](#) and then import them into the Amazon RDS for Oracle instance. For	DBA

Task	Description	Skills required

	more information about this option, see the [Additional information](#) section.
	
	To create a database link ORARDSDB that connects to the Amazon RDS master user at the target DB instance, run the following commands on the source database:
	
	`sqlplus / as sysdba`
	
	`SQL*Plus: Release 19.0.0.0.0 on *** *** ** **:**:** ****`
	`Version 19.3.0.0.0`
	
	`Copyright (c) 1982, 2019, Oracle.  All rights reserved.`
	
	`Connected to:`
	`Oracle Database 19c Standard Edition 2 Release 19.0.0.0.0`
	`Version 19.3.0.0.0`
	
	`SQL> create database link orardsdb connect to admin identified by "******" using '(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = orcl.******.us-east-1.rds.amazonaws.com)(PORT = 1521))(CONNECT_DATA = (SERVER = DEDICATED`	

Task	Description	Skills required
	```) (SERVICE_NAME = orcl)))'; Database link created. SQL> ```	
Test the database link.	Test the database link to make sure that you can connect to the Amazon RDS for Oracle target database by using `sqlplus`. ``` SQL> select name from v $database@orardsdb; NAME --------- ORCL ```	DBA

Task	Description	Skills required

Transfer the dump file to the target database.	To copy the dump file over to the Amazon RDS for Oracle database, you can either use the default `DATA_PUMP_DIR` directory or you can create your own directory by using the following code, which has to run on the target Amazon RDS instance:
	
	```exec rdsadmin.rdsadmin_util.create_directory(p_directory_name => 'DMS_TARGET_PUMP_DIR');```
	
	```PL/SQL procedure successfully completed.```
	
	The following script copies a dump file named `EXPORT_DMS_DATA.DMP` from the source instance to a target Amazon RDS for Oracle database by using the database link named `orardsdb`. You must run the script on the source database instance.
	
	```BEGIN DBMS_FILE_TRANSFER.PUT_FILE(source_directory_object => 'DMS_DATA_PUMP_DIR',```
		DBA

Task	Description	Skills required
	```source_file_name => 'EXPORT_DMS_DATA.DMP', destination_directory_object => 'DMS_TARGET_PUMP_DIR', destination_file_name => 'EXPORT_DMS_DATA.DMP', destination_database => 'orardsdb'); END; PL/SQL procedure successfully completed.```	
List the dump file in the target database.	After the PL/SQL procedure is complete, you can list the data dump file in the Amazon RDS for Oracle database by using the following code: ```select * from table (rdsadmin.rds_file_util.listdir(p_directory => 'DMS_TARGET_PUMP_DIR'));```	DBA

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create JDE-specific users in the target Instance.</td><td>Create a JD Edwards profile and role by using these commands in the target instance:

<pre><code>SQL> CREATE PROFILE
 "JDEPROFILE" LIMIT
 IDLE_TIME 15;
Profile created.

SQL> CREATE ROLE
 "JDE_ROLE";
Role created.

SQL> CREATE ROLE
 "JDEADMIN";
CREATE ROLE "JDEUSER";
Role created.
Role created.</code></pre>

Grant the required permissions to the role:

<pre><code>SQL> GRANT CREATE ANY
 SEQUENCE TO JDE_ROLE;
 GRANT DROP ANY SEQUENCE
 TO JDE_ROLE;
 GRANT CREATE ANY
 TRIGGER TO JDE_ROLE;
 GRANT DROP ANY TRIGGER
 TO JDE_ROLE;</code></pre></td><td>DBA, JDE CNC</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create tablespaces in the target instance.</td><td>Create the required tablespaces in the target instance by using the following commands for the schemas that are involved in this migration:
<pre>
SQL> CREATE TABLESPACE
 <Tablespace Name for
 Tables>;
Tablespace created.

SQL> CREATE TABLESPACE
 <Tablespace Name for
 Indexes>;
Tablespace created.
</pre></td><td>DBA, JDE CNC</td></tr>
</table>

Task	Description	Skills required
Initiate the import on the target database.	Before you start the import process, set up the roles, schemas, and tablespaces on the target Amazon RDS for Oracle database by using the data dump file. To perform the import, access the target database with the Amazon RDS primary user account, and use the connection string name in the `tnsnames.ora` file, which includes the Amazon RDS for Oracle Database `tns-entry`. If necessary, you can include a remap option to import the data dump file into a different tablespace or under a different schema name. To start the import, use the following code: `impdp admin@orardsdb directory=DMS_TARGET_PUMP_DIR logfile=import.log dumpfile=EXPORT_DMS_DATA.DMP` To ensure a successful import, check the import log file for any errors, and review details such as object count, row count, and invalid objects. If	DBA

Task	Description	Skills required
	there are any invalid objects, recompile them. Additionally, compare the source and target database objects to confirm that they match.	

Provision an AWS DMS replication instance with the source and target endpoints

Task	Description	Skills required
Download the template.	Download the AWS CloudFormation [DMS_instance.yaml](#) template to provision the AWS DMS replication instance and its source and target endpoints.	Cloud administrator, DBA
Start the stack creation.	1. Sign in to the AWS Management Console, and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation. 2. Choose **Create stack**. 3. For **Specify template**, choose **Upload a template file**. 4. Choose **Choose file**. 5. Choose the `DMS_instance.yaml` file. 6. Choose **Next**.	Cloud administrator, DBA
Specify the parameters.	1. For **Stack name**, enter your stack name.	Cloud administrator, DBA

Task	Description	Skills required
	2. For **AWS DMS Instance Parameters**, enter the following parameters: • **DMSInstanceType** – Choose the required instance for the AWS DMS replication instance, based on your business needs. • **DMSStorageSize** – Enter the storage size for the AWS DMS instance, based on the size of your migration. 3. For **Source Oracle Database Configura tion**, enter the following parameters: • **SourceOracleEndpoi ntID** – The source Oracle database server name • **SourceOracleDataba seName** – The source database service name or session ID (SID) as applicable • **SourceOracleUserName** – The source database username (the default is `system`) • **SourceOracleDBPass word** – The source	

Task	Description	Skills required
	database username's password • **SourceOracleDBPort** – The source database port 4. For **Target RDS for Oracle Database Configura tion**, enter the following parameters: • **TargetRDSOracleEnd pointID** – The target RDS database endpoint • **TargetRDSOracleDat abaseName** – The target RDS database name • **TargetRSOracleUser Name** – The target RDS username • **TargetRDSOracleDBP assword** – The target RDS password • **TargetOracleDBPort** – The target RDS database port 5. For **VPC, Subnet and Security Group Configura tion**, enter the following parameters: • **VPCID** – The VPC for the replication instance • **VPCSecurityGroupId** – The VPC security group for the replication instance	

Task	Description	Skills required
	• **DMSSubnet1** – The subnet for Availability Zone 1 • **DMSSubnet2** – The subnet for Availability Zone 2 6. Choose **Next**.	
Create the stack.	1. On the **Configure stack options** page, for **Tags**, enter any optional values. 2. Choose **Next**. 3. On the **Review** page, verify the details, and then choose **Submit**. The provisioning should complete in approximately 5–10 minutes. It is complete when the AWS CloudForm ation Stacks page shows **CREATE_COMPLETE**.	Cloud administrator, DBA

Task	Description	Skills required
Set up the endpoints.	1. Open the AWS DMS console at https://console.aws.amazon.com/dms/v2/. 2. For **Resource managemen t**, choose **Replication instances**, and then review the replication instances. 3. For **Resource managemen t**, choose **Endpoints**, and then review the endpoints.	Cloud administrator, DBA
Test connectivity.	After the source and target endpoints shows the status as **Active**, test the connectiv ity. Choose **Run test** for each endpoint (source and target) to make sure that the status shows as successful.	Cloud administrator, DBA

Create an AWS DMS replication task for live replication

Task	Description	Skills required
Create the replication task.	Create the AWS DMS replicati on task by using the following steps: 1. Open the AWS DMS console at https://console.aws.amazon.com/dms/v2/. 2. In the navigation pane, under **Migrate Data**,	Cloud administrator, DBA

Task	Description	Skills required
	choose **Database migration task**. 3. In the **Task configuration** box, for **Task identifier**, enter your task identifier. 4. For **Replication instance**, choose the DMS replication instance that you created. 5. For **Source database endpoint**, choose your source endpoint. 6. For **Target database endpoint**, choose your target Amazon RDS for Oracle database. 7. For **Migration type**, choose **Replicate data changes only**. If you receive a message that supplemen tal logging needs to be turned on, follow the instructions in the Troubleshooting section. 8. In the **Task settings** box, choose **Specify log sequence number**. 9. For **System change number**, enter the Oracle database SCN that you generated from the source Oracle database. 10. Choose **Enable validation**.	

Task	Description	Skills required
	11. Choose **Enable CloudWatch Logs**. By activating this feature, you can validate the data and [Amazon CloudWatch](https://aws.amazon.com/cloudwatch/) logs to review the AWS DMS replication instance logs. 12. Under **Selection rules**, complete the following: • For **Schema**, choose **Enter a schema**. • For **Schema name**, enter the JDE Schema name (for example: **DV920**). • For **Table name**, enter **%**. • For **Action**, choose **Include**. 13. Choose **Create task**. After you create the task, AWS DMS migrates ongoing changes to the Amazon RDS for Oracle database instance from the SCN that you provided under CDC start mode. You can also verify the migration by reviewing the CloudWatch logs.	

Task	Description	Skills required
Repeat the replication task.	Repeat the previous steps to create replication tasks for other JD Edwards schemas that are part of the migration.	Cloud administrator, DBA, JDE CNC administrator

Validate the database schema on the target Amazon RDS for Oracle database

Task	Description	Skills required
Validate the data transfer.	After the AWS DMS task starts, you can check the **Table statistics** tab on the **Tasks** page to see the changes made to the data. You can monitor the status of ongoing replication in the console on the **Database migration tasks** page. For more information, see AWS DMS data validation.	Cloud administrator, DBA

Cut over

Task	Description	Skills required
Stop replication.	Discontinue the replicati on procedure and stop the source application services.	Cloud administrator, DBA
Launch the JD Edwards application.	Launch the target JD Edwards presentation and logic tier application on AWS, and	DBA, JDE CNC administrator

Task	Description	Skills required
	direct it to the Amazon RDS for Oracle database. When you access the applicati on, you should notice that all connections are now establish ed with the Amazon RDS for Oracle database.	
Turn off the source database.	After you confirm that there are no more connections, you can turn the source database off.	DBA

Troubleshooting

Issue	Solution

You receive a warning message to enable [supplemental logging](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.Oracle.html#CHAP_Source.Oracle.Self-Managed.Configuration.SupplementalLogging) in the source database for on-going replication	Enter these commands to enable supplemen tal logging:
	
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (FOREIGN KEY) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;`
	`SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;`

AWS DMS has supplemental logging turned off.	Supplemental logging is turned off by default in AWS DMS. To turn it on for a source Oracle endpoint:

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td></td><td>

Sign in to the AWS Management Console and open the AWS DMS console at https://console.aws.amazon.com/dms/v2/.
Choose Endpoints.
Choose the Oracle source endpoint that you want to add supplemental logging to.
Choose Modify.
Choose Advanced, and then add the following code to the Extra connection attributes text box:
<pre><code>addSupplementalLogging=Y</code></pre>
Choose Modify.

</td></tr>
<tr><td>Supplemental logging isn’t enabled at the CDB level.</td><td>

Enter this command:
<pre><code>SQL> alter session set container =
 CDB$ROOT;

Session altered.</code></pre>
Repeat the steps to enable supplemental logging.

</td></tr>
<tr><td>You receive the error message: "Test Endpoint failed: Application-Status: 1020912, Applicati on-Message: LogMiner is not supported in Oracle PDB environment Endpoint initializ ation failed."</td><td>
If you encounter this error message, you can use Binary Reader instead of LogMiner.
Under Endpoint settings, add this line to the extra connection attributes for your source database:
<pre><code>useLogMinerReader=N;useBfile=Y;</code></pre>
</td></tr>
</table>

Related resources

- Getting started with AWS Database Migration Service
- Best Practices for AWS Database Migration Service
- Migrating Oracle Databases to the AWS Cloud
- AWS Database Migration Service resource type reference for AWS CloudFormation
- Manage your AWS DMS endpoint credentials with AWS Secrets Manager
- Troubleshooting migration tasks in AWS Database Migration Service
- Best practices for AWS Database Migration Service

Additional information

Transfer files using Amazon S3

To transfer the files to Amazon S3, you can use the AWS CLI or the Amazon S3 console. After you transfer the files to Amazon S3, you can use the Amazon RDS for Oracle instance to import the Data Pump files from Amazon S3.

If you choose to transfer the dump file using Amazon S3 integration as an alternate method, perform the follow steps:

1. Create an S3 bucket.
2. Export the data from the source database using Oracle Data Pump.
3. Upload the Data Pump files to the S3 bucket.
4. Download the Data Pump files from the S3 bucket to the target Amazon RDS for Oracle database.
5. Perform the import using the Data Pump files.

> **Note**
>
> To transfer large data files between S3 and RDS instances, we recommend that you use the Amazon S3 Transfer Acceleration feature.

Migrate an Oracle PeopleSoft database to AWS by using AWS DMS

sampath kathirvel, Amazon Web Services

Summary

Oracle PeopleSoft is an enterprise resource planning (ERP) solution for enterprise-wide processes. PeopleSoft has a three-tier architecture: client, application, and database. PeopleSoft can be run on Amazon Relational Database Service (Amazon RDS).

If you migrate your Oracle database to Amazon RDS, Amazon Web Services (AWS) can take care of backup tasks and high availability, leaving you free to concentrate on maintaining your PeopleSoft application and its functionality. For a comprehensive list of key factors to consider during the migration process, see Oracle database migration strategies in AWS Prescriptive Guidance.

This pattern provides a solution for migrating your on-premises Oracle databases to Amazon RDS for Oracle using Oracle Data Pump with AWS Database Migration Service (AWS DMS) and its change data capture (CDC) feature.

When migrating critical ERP applications such as Oracle PeopleSoft, minimizing the downtime is key. AWS DMS minimizes downtime by supporting both full load and continuous replication. from the source database to the target database. AWS DMS also provides real-time monitoring and logging of the migration, which can help you to identify and resolve any issues that could cause downtime.

When replicating changes with AWS DMS, you must specify a time or system change number (SCN) as the starting point for AWS DMS to read changes from the database logs. It's crucial to keep these logs accessible on the server for a designated amount of time to ensure that AWS DMS has access to these changes.

Prerequisites and limitations

Prerequisites

- Provisioned Amazon RDS for Oracle database in your AWS Cloud environment as the target database.
- An Oracle PeopleSoft database running on premises or on Amazon Elastic Compute Cloud (Amazon EC2) in the AWS Cloud.

> **Note**
>
> This pattern is designed for migrating from on premises to AWS, but it was tested by using Oracle Database on an Amazon EC2 instance. For migrating from on premises, you will need to configure the appropriate network connectivity.

- Schema details. When migrating an Oracle PeopleSoft application to Amazon RDS for Oracle, it is necessary to identify which Oracle database schema (for example, `SYSADM`) to migrate. Before starting the migration process, gather the following details about the schema:
  - Size
  - The number of objects per object type
  - The number of invalid objects.

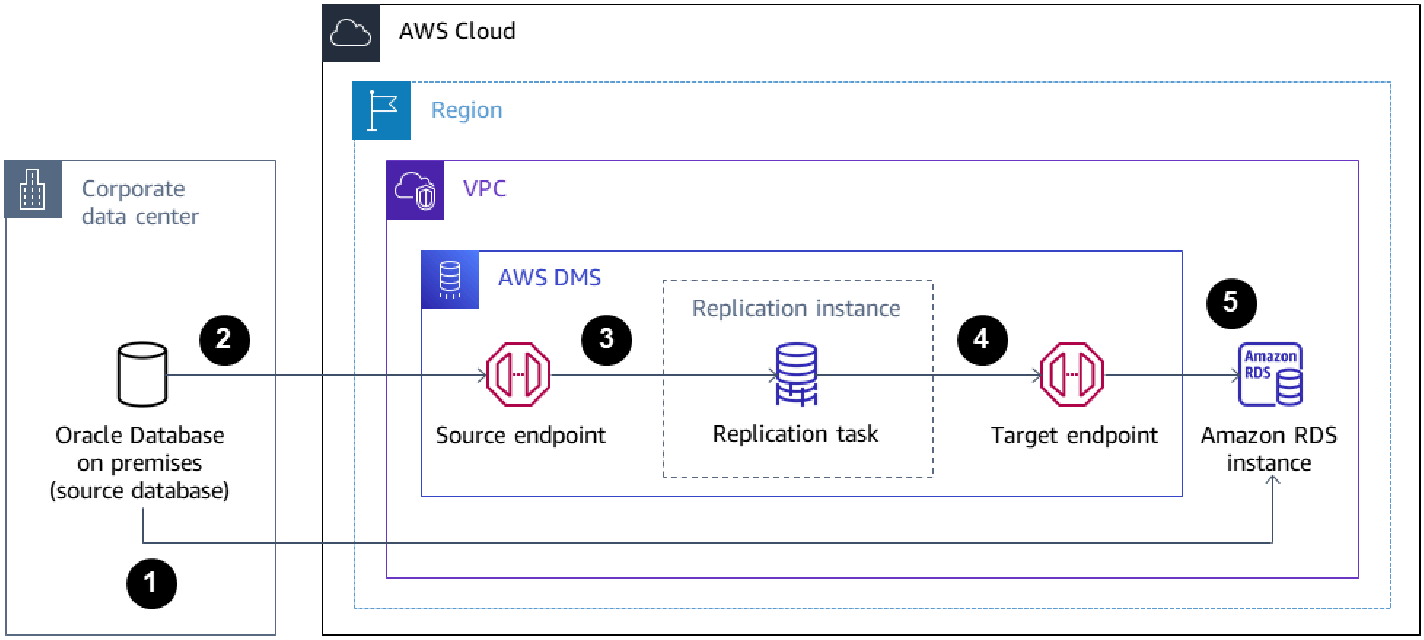
  This information will aid in the migration process.

Limitations

- This scenario has been tested only with the PeopleSoft DEMO database. It hasn't been tested with a large dataset.

Architecture

The following diagram shows an instance running an Oracle database as the source database and an Amazon RDS for Oracle database as the target database. The data is exported and imported from the source Oracle database to the target Amazon RDS for Oracle database using Oracle Data Pump and replicated for CDC changes using AWS DMS.



1. The initial step involves extracting data from the source database by using Oracle Data Pump, followed by sending it to the Amazon RDS for Oracle database target.
2. Data is sent from the source database to a source endpoint in AWS DMS.
3. From the source endpoint, the data is sent to the AWS DMS replication instance, where the replication task is performed.
4. After the replication task is completed, the data is sent to the target endpoint in AWS DMS.
5. From the target endpoint, the data is sent to the Amazon RDS for Oracle database instance.

Tools

AWS services

- [AWS Database Migration Service (AWS DMS)] helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- [Amazon Relational Database Service (Amazon RDS) for Oracle] helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.

Other services

- [Oracle Data Pump] helps you move data and metadata from one database to another at high speeds.

Best practices

Migrating LOBs

If your source database contains large binary objects (LOBs) that need to be migrated to the target database, AWS DMS provides the following options:

- **Full LOB mode –** AWS DMS migrates all the LOBs from the source to the target database regardless of their size. Although the migration is slower, the advantage is that data isn't truncated. For better performance, you can create a separate task on the new replication instance to migrate the tables that have LOBs larger than a few megabytes.
- **Limited LOB mode –** You specify the maximum size of LOB column data, which allows AWS DMS to pre-allocate resources and apply the LOBs in bulk. If the size of the LOB columns exceeds the size that is specified in the task, AWS DMS truncates the data and sends warnings to the AWS DMS log file. You can improve performance by using Limited LOB mode if your LOB data size is within the Limited LOB size.
- **Inline LOB mode –** You can migrate LOBs without truncating the data or slowing the performance of your task by replicating both small and large LOBs. First, specify a value for the InlineLobMaxSize parameter, which is available only when Full LOB mode is set to true. The AWS DMS task transfers the small LOBs inline, which is more efficient. Then, AWS DMS migrates the large LOBs by performing a lookup from the source table. However, Inline LOB mode works only during the full load phase.

Generating sequence values

Keep in mind that during the change data capture process with AWS DMS, incremental sequence numbers are not replicated from the source database. To avoid discrepancies in sequence values, you must generate the most recent sequence value from the source for all sequences, and apply it to the target Amazon RDS for Oracle database.

Credential management

To help secure your AWS resources, we recommend following the best practices for AWS Identity and Access Management (IAM).

Epics

Provision an AWS DMS replication instance with the source and target endpoints

Task	Description	Skills required
Download the template.	Download the DMS_instance.yaml AWS CloudFormation template to provision the AWS DMS replication instance and its source and target endpoints.	Cloud administrator, DBA
Start the stack creation.	1. On the AWS Management Console, choose **CloudFormation**. 2. Choose **Create stack**. 3. For **Specify template**, choose **Upload a template file**. 4. Choose **Choose file**. 5. Choose the `DMS_instance.yaml` file. 6. Choose **Next**.	Cloud administrator, DBA
Specify the parameters.	1. For **Stack name**, enter your stack name. 2. Under **AWS DMS Instance Parameters**, enter the following parameters: • **DMSInstanceType** – Choose the required instance for the AWS DMS replication instance, based on your business needs.	Cloud administrator, DBA

Task	Description	Skills required
	• **DMSStorageSize –** Enter the storage size for the AWS DMS instance, based on the size of your migration. 3. Under **Source Oracle Database Configura tion**, enter the following parameters: • **SourceOracleEndpoi ntID –** The source Oracle database server name • **SourceOracleDataba seName –** The source database service name or session ID (SID) as applicable • **SourceOracleUserName –** The source database username (the default is system) • **SourceOracleDBPass word –** The source database username's password • **SourceOracleDBPort –** The source database port 4. Under **Target RDS for Oracle Database Configuration**, enter the following parameters:	

Task	Description	Skills required
	• **TargetRDSOracleEndpointID** – The target RDS database endpoint • **TargetRDSOracleDatabaseName** – The target RDS database name • **TargetRSOracleUserName** – The target RDS username • **TargetRDSOracleDBPassword** – The target RDS password • **TargetOracleDBPort** – The target RDS database port 5. Under **VPC, Subnet and Security Group Configuration**, enter the following parameters: • **VPCID** – The VPC for the replication instance • **VPCSecurityGroupId** – The VPC security group for the replication instance • **DMSSubnet1** – The subnet for Availability Zone 1 • **DMSSubnet2** – The subnet for Availability Zone 2 6. Choose **Next**.	

Task	Description	Skills required
Create the stack.	1. On the **Configure stack options** page, for **Tags**, enter any optional values. 2. Choose **Next**. 3. On the **Review** page, verify the details, and then choose **Submit**. The provisioning should complete in approximately 5–10 minutes. It is complete when the AWS CloudForm ation Stacks page shows **CREATE_COMPLETE**.	Cloud administrator, DBA
Set up the endpoints.	1. From the AWS Managemen t Console, choose **Database Migration Services**. 2. Under Resource management, choose **Replication instances**. 3. Under Resource management, choose **Endpoints**.	Cloud administrator, DBA
Test connectivity.	After the source and target endpoints shows status as Active, test the connectivity. Choose **Run test** for each endpoint (source and target) to make sure that the status shows as successful.	Cloud administrator, DBA

Export the PeopleSoft schema from the on-premises Oracle database by using Oracle Data Pump

Task	Description	Skills required

Generate the SCN.	When the source database is active and in use by the application, initiate the data export with Oracle Data Pump. You must first generate a system change number (SCN) from the source database for both data consistency during the export with Oracle Data Pump and as a starting point for change data capture in AWS DMS.
	
	To generate the current SCN from your source database, enter the following SQL statement.
	
	`SQL> select name from v$database;`
	`SQL> select name from v$database;`
	`NAME`
	`---------`
	`PSFTDMO`
	`SQL> SELECT current_scn FROM v$database;`
	`CURRENT_SCN`
	`-----------`
	`23792008`
	
	Save the generated SCN to use when you export the data
		DBA

Task	Description	Skills required
	and for creating the AWS DMS replication task.	

Task	Description	Skills required

Create the parameter file.	To create a parameter file for exporting the schema, you can use the following code.
	`$ cat exp_datapmp.par`
	`userid=system/*******`
	`directory=DATA_PUMP_DIR`
	`logfile=export_dms_sample_user.log`
	`dumpfile=export_dms_sample_data_%U.dmp`
	`schemas=SYSADM`
	`flashback_scn=23792008`
	Note
	You can also define your own `DATA_PUMP_DIR` by using the following commands, based on your requirements.
	`SQL> CREATE OR REPLACE DIRECTORY DATA_PUMP_DIR AS '/opt/oracle/product/19c/dbhome_1/dmsdump/';`
	`Directory created.`
	`SQL> GRANT READ, WRITE ON DIRECTORY DATA_PUMP_DIR TO system;`
	`Grant succeeded.`
	`SQL>`
		DBA

Task	Description	Skills required
	``` SQL> SELECT owner, directory_name, directory_path FROM dba_directories WHERE directory_name='DA TA_PUMP_DIR'; OWNER DIRECTORY_NAME DIRECTORY_PATH --------------------- ------------------ ------------------ ------------------ ------------------ --------------------- SYS DATA_PUMP_DIR /opt/ oracle/product/19c/dbh ome_1/dmsdump/ ```	

Task	Description	Skills required
Export the schema.	To perform the export, use the `expdp` utility. `$ expdp parfile=e xp_datapmp.par .... Transferring the dump file with DBMS_FILE _TRANSFER to Target: . . exported "SYSADM". "PS_XML_TEMPLT_LNG" 6.320 KB 0 rows . . exported "SYSADM". "PS_XML_TEMPLT_LNK" 6.328 KB 0 rows . . exported "SYSADM". "PS_XML_XLATDEF_LNG" 6.320 KB 0 rows . . exported "SYSADM". "PS_XML_XLATITM_LNG" 7.171 KB 0 rows . . exported "SYSADM". "PS_XPQRYRUNCNTL" 7.601 KB 0 rows . . exported "SYSADM". "PS_XPQRYRUNPARM" 7.210 KB 0 rows . . exported "SYSADM". "PS_YE_AMOUNTS" 9.351 KB 0 rows . . exported "SYSADM". "PS_YE_DATA" 16.58 KB 0 rows . . exported "SYSADM". "PS_YE_EE" 6.75 KB 0 rows . . exported "SYSADM". "PS_YE_W2CP_AMOUNTS" 9.414 KB 0 rows`	DBA

Task	Description	Skills required
	``` . . exported "SYSADM". "PS_YE_W2CP_DATA" 20.94 KB 0 rows . . exported "SYSADM". "PS_YE_W2C_AMOUNTS" 10.27 KB 0 rows . . exported "SYSADM". "PS_YE_W2C_DATA" 20.95 KB 0 rows . . exported "SYSADM". "PS_ZBD_JOBCODE_TBL" 14.60 KB 0 rows . . exported "SYSADM". "PTGRANTTBL" 5.468 KB 0 rows Master table "SYSTEM". "SYS_EXPORT_SCHEMA _01" successfully loaded/unloaded ** Dump file set for SYSTEM.SYS_EXPORT_ SCHEMA_01 is: /opt/oracle/pr oduct/19c/dbhome_1 /dmsdump/export_dm s_sample_data_01.dmp Job "SYSTEM"."SYS_EXPO RT_SCHEMA_01" successfully completed at Mon Dec 19 20:13:57 2022 elapsed 0 00:38:22 ```	

Import the PeopleSoft schema into the Amazon RDS for Oracle database by using Oracle Data Pump

Task	Description	Skills required

Transfer the dump file to the target instance.	To transfer your files using `DBMS_FILE_TRANSFER` , you need to create a database link from the source database to the Amazon RDS for Oracle instance. After the link is established, you can use the utility to transfer the Data Pump files directly to the RDS instance.
	
	Alternatively, you can transfer the Data Pump files to [Amazon Simple Storage Service (Amazon S3)](#) and then import them into the Amazon RDS for Oracle instance. For more information about this option, see the Additional information section.
	
	To create a database link `ORARDSDB` that connects to the Amazon RDS master user at the target DB instance, run the following commands on the source database.
	
	`$sqlplus / as sysdba`
	`$ SQL> create database link orardsdb connect to admin identified by "*****" using '(DESCRIP`
		DBA

Task	Description	Skills required
	``` TION = (ADDRESS = (PROTOCOL = TCP)(HOST = testpsft.*******.u s-west-2.rds.amazo naws.com)(PORT = 1521))(CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME = orcl)))'; Database link created. ```	
Test the database link.	Test the database link to make sure that you can connect using sqlplus to the Amazon RDS for Oracle target database. ``` SQL> SQL> select name from v $database@orardsdb; NAME --------- ORCL SQL> ```	DBA

Task	Description	Skills required

Transfer the dump file to the target database.	To copy the dump file over to Amazon RDS for Oracle database, you can either use the default `DATA_PUMP_DIR` directory or you can create your own directory using the following code.
	`exec rdsadmin.rdsadmin_util.create_directory(p_directory_name => ‘TARGET_PUMP_DIR’);`
	The following script copies a dump file named `export_dms_sample_data_01.dmp` from the source instance to a target Amazon RDS for Oracle database using the database link named `orardsdb`.
	`$ sqlplus / as sysdba`
	`SQL>`
	`BEGIN`
	`DBMS_FILE_TRANSFER.PUT_FILE(`
	`source_directory_object => 'DATA_PUMP_DIR',`
	`source_file_name => 'export_dms_sample_data_01.dmp',`
	`destination_directory_object => 'TARGET_PUMP_DIR’',`
		DBA

Task	Description	Skills required
	```destination_file_name => 'export_dms_sample_data_01.dmp', destination_database => 'orardsdb'); END; / PL/SQL procedure successfully completed .```	
List the dump file in the target database.	After the PL/SQL procedure is completed, you can list the data dump file in the Amazon RDS for Oracle database by using the following code. ```SQL> select * from table (rdsadmin.rds_file_util.listdir(p_directory => ‘TARGET_PUMP_DIR’));```	DBA

Task	Description	Skills required
Initiate the import on the target database.	Before you start the import process, set up the roles, schemas, and tablespaces on the target Amazon RDS for Oracle database by using the data dump file. To perform the import, access the target database with the Amazon RDS master user account, and use the connection string name in the `tnsnames.ora` file, which includes the Amazon RDS for Oracle Database `tns-entry`. If necessary, you can include a remap option to import the data dump file into a different tablespace or under a different schema name. To start the import, use the following code. `impdp admin@orardsdb directory=TARGET_PUMP_DIR logfile=import.log dumpfile=export_dms_sample_data_01.dmp` To ensure a successful import, check the import log file for any errors, and review details such as object count, row	DBA

Task	Description	Skills required
	count, and invalid objects. If there are any invalid objects, recompile them. Additionally, compare the source and target database objects to confirm that they match.	

Create an AWS DMS replication task using CDC to perform live replication

Task	Description	Skills required
Create the replication task.	Create the AWS DMS replication task by using the following steps: 1. On the AWS DMS console, under **Conversion & migration**, choose **Database migration task**. 2. Under **Task configuration**, for **Task identifier**, enter your task identifier. 3. For **Replication instance**, choose the DMS replication instance that you created. 4. For **Source database endpoint**, choose your source endpoint. 5. For **Target database endpoint**, choose your target Amazon RDS for Oracle database. 6. For **Migration type**, choose **Replicate data**	Cloud administrator, DBA

Task	Description	Skills required
	changes only. If you receive a message that supplemental logging needs to be turned on, follow the instructions in the *Additional information* section. 7. Under **Task settings**, select **Specify log sequence number**. 8. For **System change number**, enter the Oracle database SCN that you generated from the source Oracle database. 9. Choose **Enable validation**. 10. Choose **Enable CloudWatch Logs**. By activating this feature, you can validate the data and [Amazon CloudWatch](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html) logs to review the AWS DMS replication instance logs. 11. Under **Selection rules**, complete the following: • For **Schema**, choose **Enter a schema**. • For **Schema name**, enter **SYSADM**. • For **Table name**, enter **%**.	

Task	Description	Skills required
	• For **Action**, choose **Include**. 12. Under **Transformation rules**, complete the following: • For **Target**, choose **Table**. • For **Scheme name**, choose **Enter a schema**. • For **Schema name**, enter SYSADM. • For **Action**, choose **Rename to**. 13. Choose **Create task**. After you create the task, it migrates the CDC to the Amazon RDS for Oracle database instance from the SCN that you provided under CDC start mode. You can also verify by reviewing the CloudWatch logs.	

Validate the database schema on the target Amazon RDS for Oracle database

Task	Description	Skills required
Validate the data transfer.	After the AWS DMS task starts, you can check the **Table statistics** tab on the **Tasks** page to see the changes made to the data.	Cloud administrator, DBA

Task	Description	Skills required
	You can monitor the status of ongoing replication in the console on the **Database migration tasks** page. For more information, see AWS DMS data validation.	

Cut over

Task	Description	Skills required
Stop replication.	Discontinue the replication procedure and halt the source application services.	Cloud administrator, DBA
Launch the PeopleSoft middle tier.	Launch the target PeopleSof t middle tier application in AWS, and direct it to the recently migrated Amazon RDS for Oracle database. When you access the applicati on, you should notice that all app connections are now established with the Amazon RDS for Oracle database.	DBA, PeopleSoft administr ator
Turn off the source database.	After you confirm that there are no more connections to the source database, it can be turned off.	DBA

Related resources

- [Getting started with AWS Database Migration Service]
- [Best Practices for AWS Database Migration Service]
- [Migrating Oracle Databases to the AWS Cloud]

Additional information

Transfer files using Amazon S3

To transfer the files to Amazon S3, you can use the AWS CLI or the Amazon S3 console. After you transfer the files to Amazon S3, you can use the Amazon RDS for Oracle instance to import the Data Pump files from Amazon S3.

If you choose to transfer the dump file using Amazon S3 integration as an alternate method, perform the follow steps:

1. Create an S3 bucket.
2. Export the data from the source database using Oracle Data Pump.
3. Upload the Data Pump files to the S3 bucket.
4. Download the Data Pump files from the S3 bucket to the target Amazon RDS for Oracle database.
5. Perform the import using the Data Pump files.

> **Note**
>
> To transfer large data files between S3 and RDS instances, it is recommended to use the Amazon S3 Transfer Acceleration feature.

Activate supplemental logging

If you receive a warning message to enable supplemental logging in the source database for on-going replication, use the following steps.

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
```

```
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (FOREIGN KEY) COLUMNS;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (UNIQUE) COLUMNS;
```

Migrate an on-premises MySQL database to Amazon RDS for MySQL

Lorenzo Mota, Amazon Web Services

Summary

This pattern provides guidance for migrating an on-premises MySQL database to Amazon Relational Database Service (Amazon RDS) for MySQL. The pattern discusses the use of AWS Database Migration Service (AWS DMS) or native MySQL tools such as **mysqldump** for a full database migration. This pattern is primarily for DBAs and solution architects. It can be used in small or large projects as a testing procedure (we recommend at least one testing cycle) or as a final migration procedure.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A MySQL source database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- MySQL versions 5.5, 5.6, 5.7, 8.0. For the latest list of supported versions, see MySQL on Amazon RDS in the AWS documentation. If you're using AWS DMS, see also Using a MySQL-Compatible Database as a Target for AWS DMS for MySQL versions currently supported by AWS DMS.

Architecture

Source technology stack

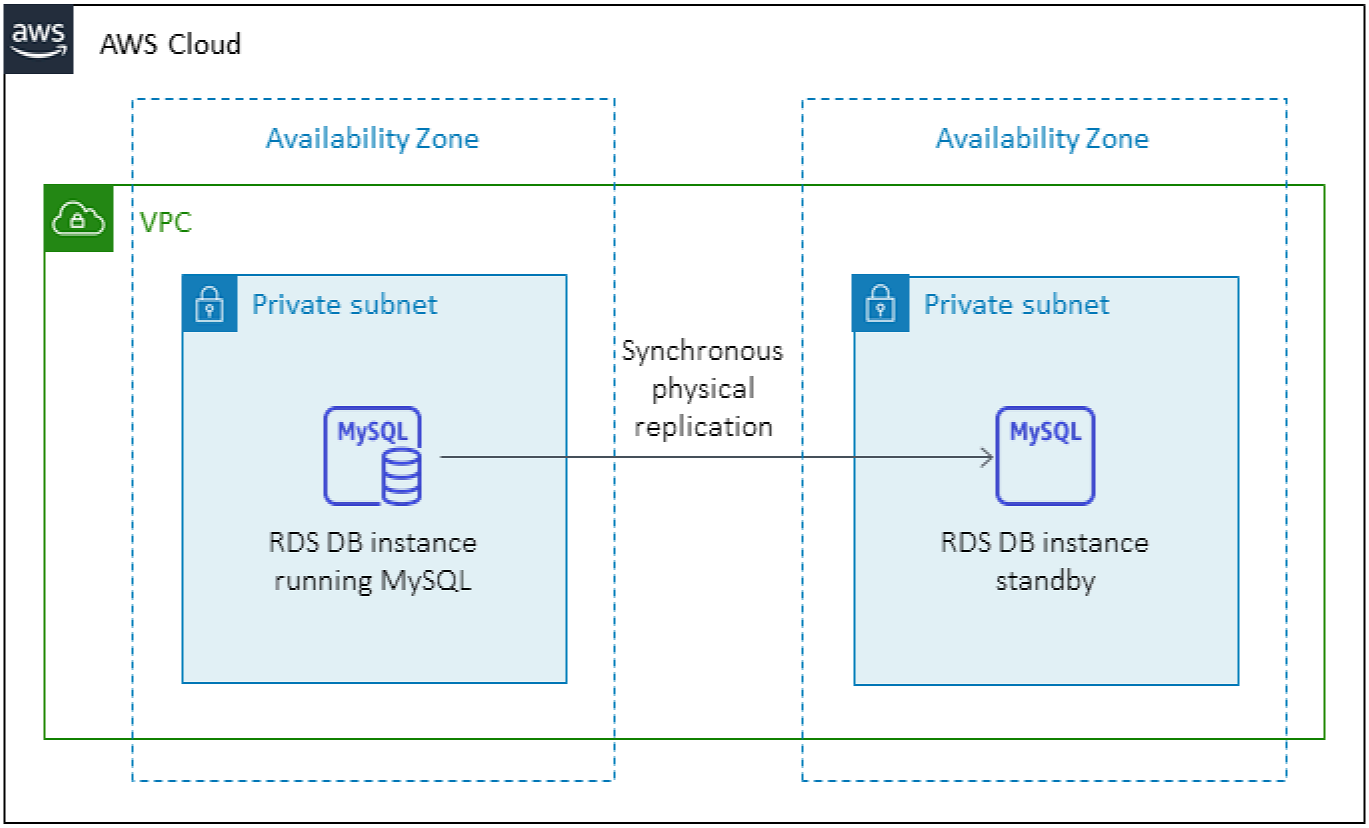
- An on-premises MySQL database

Target technology stack

- An Amazon RDS DB instance running MySQL

Target architecture

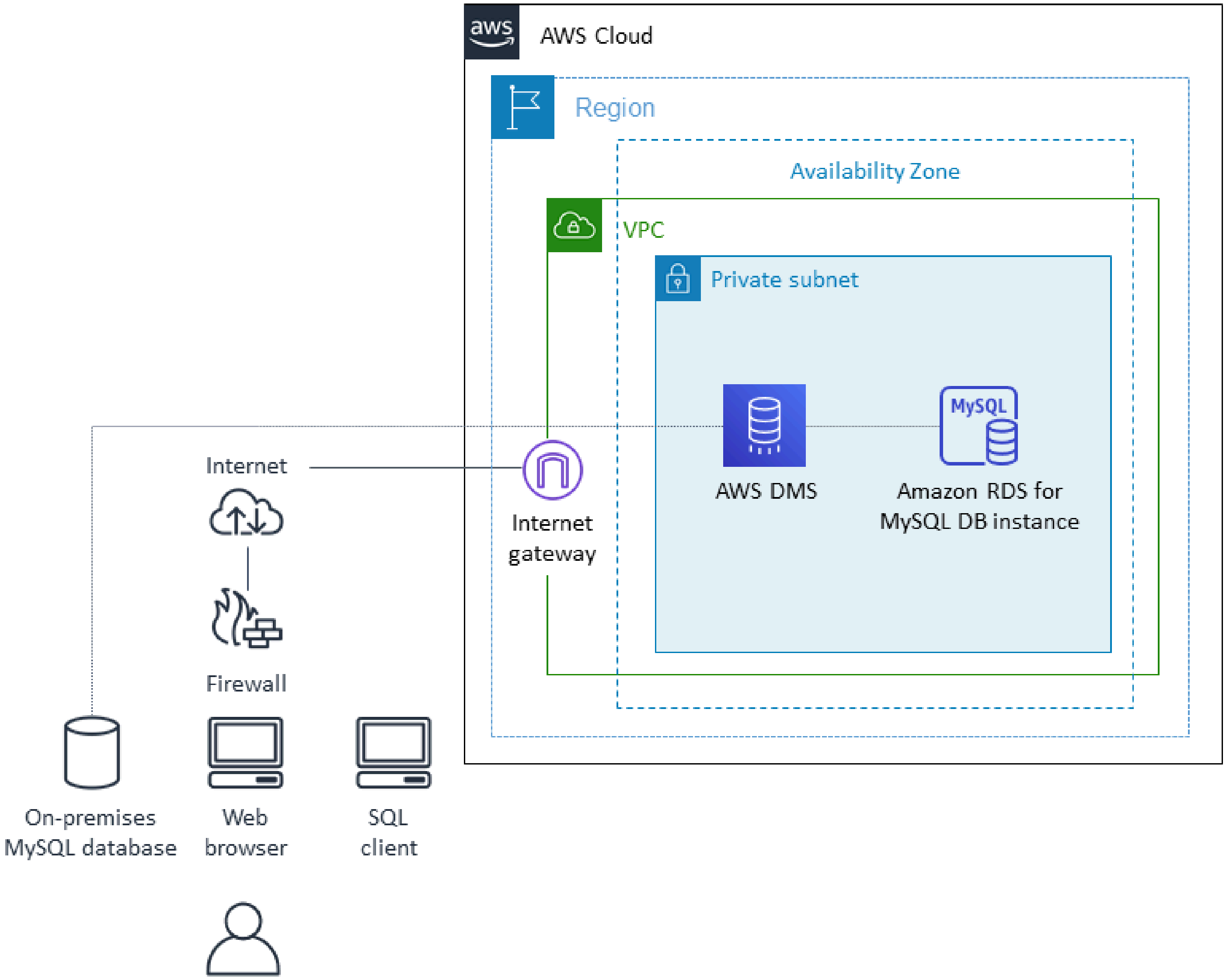
The following diagram shows the target Amazon RDS for MySQL implementation after migration.



AWS data migration architecture

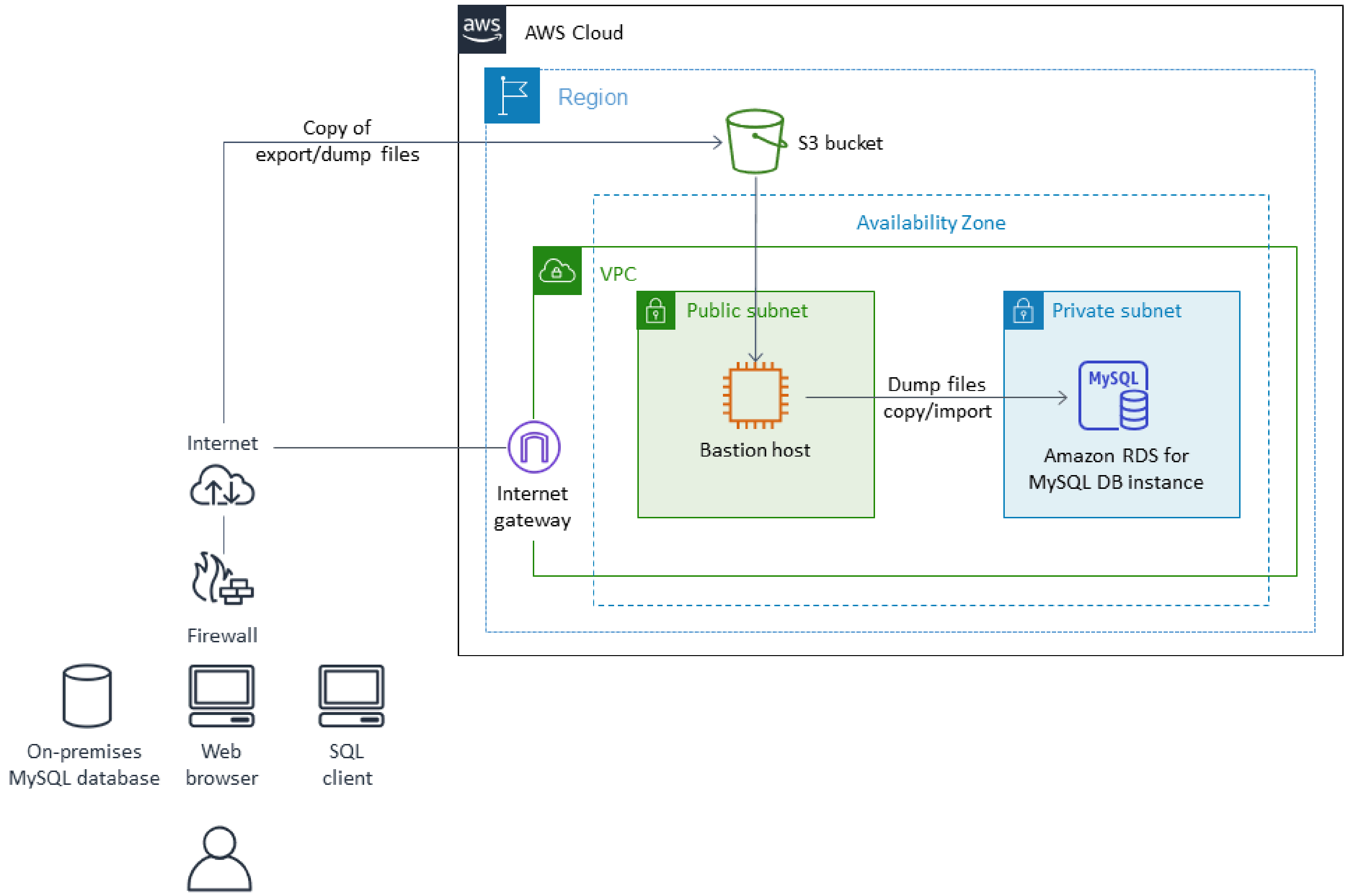
Using AWS DMS:

The following diagram shows the data migration architecture when you use AWS DMS to send full and incremental changes until cutover. The network connection from on premises to AWS depends on your requirements and is out of scope for this pattern.



Using native MySQL tools:

The following diagram shows the data migration architecture when you use native MySQL tools. The export dump files are copied to Amazon Simple Storage Service (Amazon S3) and imported into the Amazon RDS for MySQL database in AWS before the cutover. The network connection from on premises to AWS depends on your requirements and is out of scope for this pattern.



Notes:

- Depending on downtime requirements and the size of the database, using AWS DMS or a change data capture (CDC) tool minimizes cutover time. AWS DMS can help reduce cutover time to the new target to a minimum (typically minutes). An offline strategy with **mysqldump** can suffice if the size of the database and network latency allow for a short window. (We recommend testing to get an approximate time.)
- Usually a CDC strategy such as AWS DMS requires more monitoring and complexity than offline options.

Tools

- **AWS services**: AWS Database Migration Service (AWS DMS) helps you migrate data stores to the AWS Cloud or between combinations of cloud and on-premises setups. For information about MySQL source and target databases supported by AWS DMS, see Migrating MySQL-Compatible

Databases to AWS. If your source database isn't supported by AWS DMS, you must choose another method to migrate your data.

- **Native MySQL tools**: mysqldump
- **Third-party tools**: Percona XtraBackup

Epics

Plan the migration

Task	Description	Skills required
Validate database versions.	Validate the source and target database versions.	DBA
Identify hardware requireme nts.	Identify the hardware requirements for the target server.	DBA, Systems administrator
Identify storage requireme nts.	Identify storage requireme nts (such as storage type and capacity) for the target database.	DBA, Systems administrator
Choose the instance type.	Choose the target instance type based on capacity, storage features, and networking features.	DBA, Systems administrator
Identify network access requirements.	Identify the security requirements for network access for the source and target databases.	DBA, Systems administrator
Identify unsupported objects.	Identify unsupported objects (if any) and determine the migration effort.	DBA

Task	Description	Skills required
Identify dependencies.	Identify any dependencies on remote databases.	DBA
Determine the application migration strategy.	Determine the strategy for migrating client applications.	DBA, App owner, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC).	Configure route tables, internet gateway, NAT gateways, and subnets. For more information, see VPCs and Amazon RDS in the Amazon RDS documentation.	Systems administrator
Create security groups.	Configure ports and CIDR ranges or specific IPs depending on your requireme nts. The default port for MySQL is 3306. For more information, see Controlling access with security groups in the Amazon RDS documenta tion.	Systems administrator
Configure and start an Amazon RDS for MySQL DB instance.	For instructions, see Creating an Amazon RDS DB instance in the Amazon RDS documentation. Check for supported versions.	Systems administrator

Migrate data – option 1 (using native tools)

Task	Description	Skills required
Use native MySQL tools or third-party tools to migrate database objects and data.	For instructions, see the documentation for MySQL tools such as mysqldump and Percona XtraBackup (for physical migration). For more information about options, see the blog post Migration options for MySQL to Amazon RDS for MySQL or Amazon Aurora MySQL.	DBA

Migrate data – option 2 (using AWS DMS)

Task	Description	Skills required
Migrate data with AWS DMS.	For instructions, see the AWS DMS documentation.	DBA

Perform preliminary tasks before cutover

Task	Description	Skills required
Fix object count discrepancies.	Collect object counts from the source database and new target database. Fix discrepan cies in the target database.	DBA
Check dependencies.	Check whether dependenc ies (links) to and from other databases are valid and work as expected.	DBA

Task	Description	Skills required
Perform tests.	If this is a testing cycle, perform query testing, gather metrics, and fix issues.	DBA

Cut over

Task	Description	Skills required
Switch to the target database.	Switch client applications to the new infrastructure.	DBA, App owner, Systems administrator
Provide testing support.	Provide support for functiona l application testing.	DBA

Close the project

Task	Description	Skills required
Shut down resources.	Shut down the temporary AWS resources you created for the migration.	DBA, Systems administrator
Validate project documents.	Review and validate the project documents.	DBA, App owner, Systems administrator
Gather metrics.	Gather metrics such as time to migrate, percentage of manual versus automated effort, cost savings, and so on.	DBA, App owner, Systems administrator
Close out the project.	Close out the project and provide feedback.	DBA, App owner, Systems administrator
Decommission the source database.	When all migration and cutover tasks are complete,	DBA, Systems administrator

Task	Description	Skills required
	decommission the on-premises database.	

Related resources

References

- Migration strategy for relational databases
- AWS DMS website
- AWS DMS documentation
- Amazon RDS documentation
- Amazon RDS pricing
- Amazon VPC and Amazon RDS
- Amazon RDS Multi-AZ deployments
- Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3
- Amazon RDS DB instance storage

Tutorials

- Getting Started with AWS DMS
- Getting Started with Amazon RDS

Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server

Henrique Lobao, Jonathan Pereira Cruz, and Vishal Singh, Amazon Web Services

Summary

This pattern provides guidance for migrating from an on-premises Microsoft SQL Server database to Amazon Relational Database Service (Amazon RDS) for SQL Server. It describes two options for migration: using AWS Data Migration Service (AWS DMS) or using native Microsoft SQL Server tools such as Copy Database Wizard.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source Microsoft SQL Server database in an on-premises data center

Limitations

- Database size limit: 16 TB

Product versions

- SQL Server 2014-2019, Enterprise, Standard, Workgroup, and Developer editions. For the latest list of supported versions and features, see Microsoft SQL Server on Amazon RDS in the AWS documentation. If you're using AWS DMS, see also Using a Microsoft SQL Server Database as a Target for AWS DMS for SQL Server versions supported by AWS DMS.

Architecture

Source technology stack

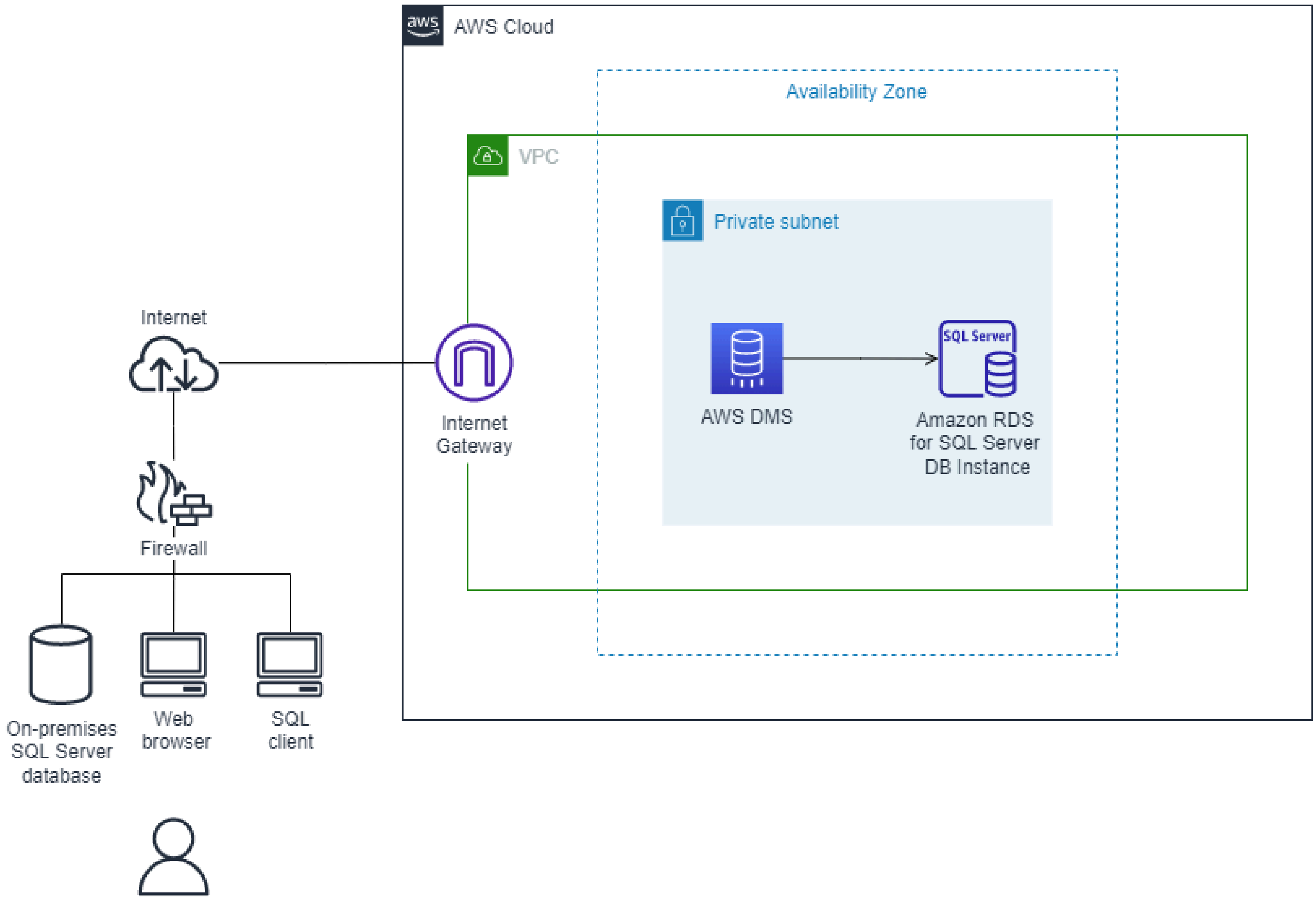
- An on-premises Microsoft SQL Server database

Target technology stack

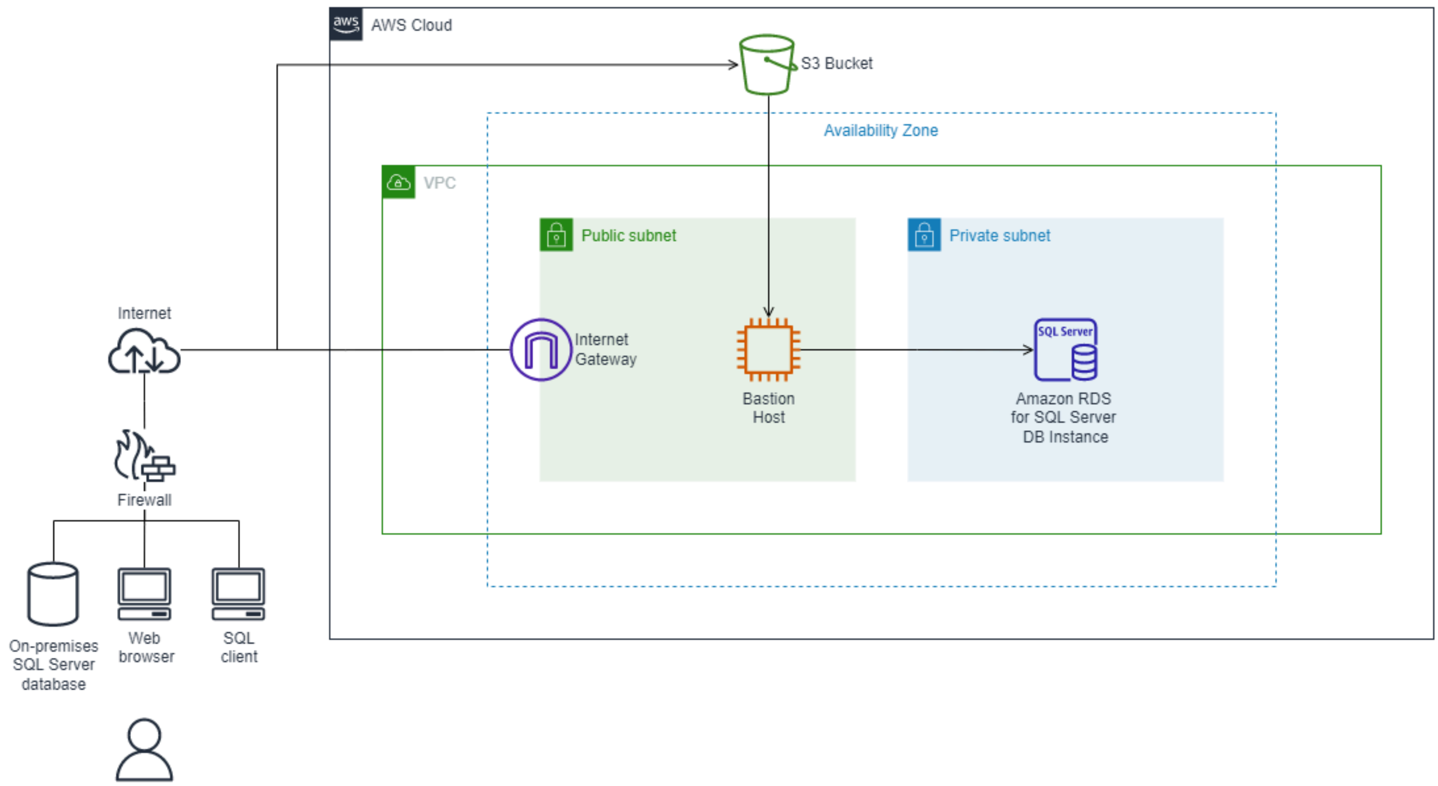
- An Amazon RDS for SQL Server DB instance

Source and target architecture

Using AWS DMS:



Using native SQL Server tools:



Tools

- AWS DMS supports several types of source and target databases. For details, see AWS DMS Step-by-Step Walkthroughs. If AWS DMS doesn't support the source database, select another method for migrating the data.
- Native Microsoft SQL Server tools include backup and restore, Copy Database Wizard, copy and attach database.

Epics

Plan the migration

Task	Description	Skills required
Validate the source and target database version and engine.		DBA

Task	Description	Skills required
Identify the hardware requirements for the target server instance.		DBA, Systems administrator
Identify the storage requirements (storage type and capacity).		DBA, Systems administrator
Choose the proper instance type based on capacity, storage features, and network features.		DBA, Systems administrator
Identify the network access security requirements for source and target databases.		DBA, Systems administrator
Identify the application migration strategy.		DBA, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC).		Systems administrator
Create security groups.		Systems administrator
Configure and start an Amazon RDS DB instance.		DBA, Systems administrator

Migrate data - option 1

Task	Description	Skills required
Use native SQL Server tools or third-party tools to migrate database objects and data.		DBA

Migrate data - option 2

Task	Description	Skills required
Migrate data with AWS DMS.		DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.		DBA, App owner, Systems administrator

Cut over

Task	Description	Skills required
Switch the application clients over to the new infrastru cture.		DBA, App owner, Systems administrator

Close the project

Task	Description	Skills required
Shut down the temporary AWS resources.		DBA, Systems administrator
Review and validate the project documents.		DBA, App owner, Systems administrator
Gather metrics such as time to migrate, percentage of manual versus automated tasks, and cost savings.		DBA, App owner, Systems administrator
Close out the project and provide feedback.		DBA, App owner, Systems administrator

Related resources

References

- Deploying Microsoft SQL Server on Amazon Web Services
- AWS DMS website
- Amazon RDS Pricing
- Microsoft Products on AWS
- Microsoft Licensing on AWS
- Microsoft SQL Server on AWS
- Using Windows Authentication with a Microsoft SQL Server DB Instance
- Amazon RDS Multi-AZ Deployments

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started with Amazon RDS
- AWS DMS (video)

- Amazon RDS (video)

Migrate data from Microsoft Azure Blob to Amazon S3 by using Rclone

Suhas Basavaraj, Aidan Keane, and Corey Lane, Amazon Web Services

Summary

This pattern describes how to use Rclone to migrate data from Microsoft Azure Blob object storage to an Amazon Simple Storage Service (Amazon S3) bucket. You can use this pattern to perform a one-time migration or an ongoing synchronization of the data. Rclone is a command-line program written in Go and is used to move data across various storage technologies from cloud providers.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Data stored in Azure Blob container service

Architecture

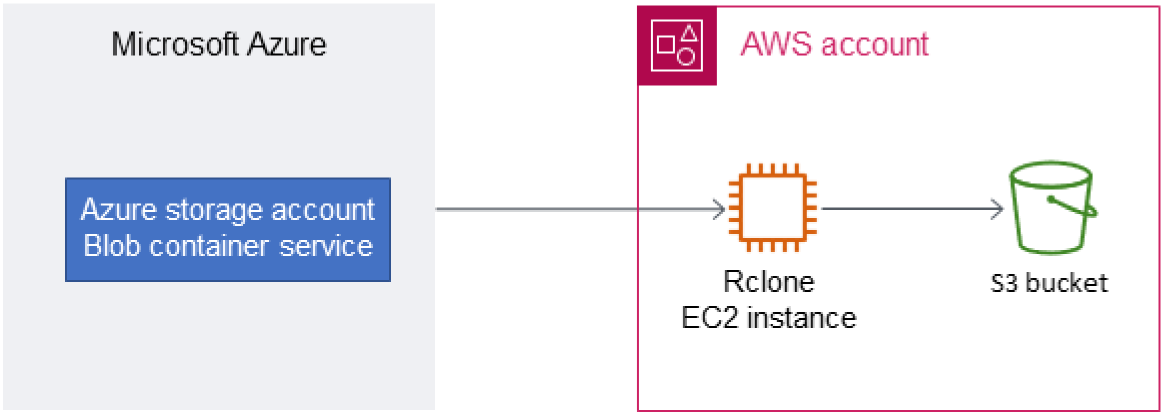
Source technology stack

- Azure Blob storage container

Target technology stack

- Amazon S3 bucket
- Amazon Elastic Compute Cloud (Amazon EC2) Linux instance

Architecture



Tools

- [Amazon Simple Storage Service (Amazon S3)](#) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- [Rclone](#) is an open-source command-line program inspired by **rsync**. It is used to manage files across many cloud storage platforms.

Best practices

When you migrate data from Azure to Amazon S3, be mindful of these considerations to avoid unnecessary costs or slow transfer speeds:

- Create your AWS infrastructure in the same geographical Region as the Azure storage account and Blob container—for example, AWS Region `us-east-1` (N. Virginia) and Azure region `East US`.
- Avoid using NAT Gateway if possible, because it accrues data transfer fees for both ingress and egress bandwidth.
- Use a [VPC gateway endpoint for Amazon S3](#) to increase performance.
- Consider using an AWS Graviton2 (ARM) processor-based EC2 instance for lower cost and higher performance over Intel x86 instances. Rclone is heavily cross-compiled and provides a precompiled ARM binary.

Epics

Prepare AWS and Azure cloud resources

Task	Description	Skills required
Prepare a destination S3 bucket.	Create a new S3 bucket in the appropriate AWS Region or choose an existing bucket as the destination for the data you want to migrate.	AWS administrator
Create an IAM instance role for Amazon EC2.	Create a new AWS Identity and Access Management (IAM) role for Amazon EC2. This role gives your EC2 instance write access to the destination S3 bucket.	AWS administrator
Attach a policy to the IAM instance role.	Use the IAM console or AWS Command Line Interface (AWS CLI) to create an inline policy for the EC2 instance role that allows write access permissions to the destinati on S3 bucket. For an example policy, see the Additional information section.	AWS administrator
Launch an EC2 instance.	Launch an Amazon Linux EC2 instance that is configured to use the newly created IAM service role. This instance will also need access to Azure public API endpoints through the internet.	AWS administrator

Task	Description	Skills required
	Note Consider using [AWS Graviton-based EC2 instances](#) to lower costs. Rclone provides ARM-compiled binaries.	
Create an Azure AD service principal.	Use the Azure CLI to create an Azure Active Directory (Azure AD) service principal that has read-only access to the source Azure Blob storage container. For instructions, see the [Additional information](#) section. Store these credentials on your EC2 instance to the location `~/azure-principal.json` .	Cloud administrator, Azure

Install and configure Rclone

Task	Description	Skills required
Download and install Rclone.	Download and install the Rclone command-line program. For installation instructions, see the [Rclone installation documentation](#).	General AWS, Cloud administrator
Configure Rclone.	Copy the following `rclone.conf` sample file. Replace `AZStorage`	General AWS, Cloud administrator

Task	Description	Skills required

	`Account` with your Azure Storage account name and `us-east-1` with the AWS Region where your S3 bucket is located. Save this file to the location `~/.config/rclone/rclone.conf` on your EC2 instance.
	
	`[AZStorageAccount]`
	`type = azureblob`
	`account = AZStorage Account`
	`service_principal_file = azure-principal.json`
	
	`[s3]`
	`type = s3`
	`provider = AWS`
	`env_auth = true`
	`region = us-east-1`	

Task	Description	Skills required

Verify Rclone configuration.	To confirm that Rclone is configured and permissions are working properly, verify that Rclone can parse your configuration file and that objects inside your Azure Blob container and S3 bucket are accessible. See the following for example validation commands.
	
	• List the configured remotes in the configuration file. This will ensure that your configuration file is being parsed correctly. Review the output to make sure that it matches your `rclone.conf` file.
	
	`rclone listremotes`
	`AZStorageAccount:`
	`s3:`
	
	• List the Azure Blob containers in the configured account. Replace `AZStorageAccount` with the storage account name that you used in the `rclone.conf` file.
	
	`rclone lsd AZStorageAccount:`
	`2020-04-29 08:29:26 docs`
		General AWS, Cloud administrator

Task	Description	Skills required

	• List the files in the Azure Blob container. Replace docs in this command with an actual Blob container name in your Azure storage account.
	`rclone ls AZStorageAccount:docs`
	`824884 administrator-en.a4.pdf`
	• List the buckets in your AWS account.
	`[root@ip-10-0-20-157 ~]# rclone lsd s3:`
	`2022-03-07 01:44:40 amzn-s3-demo-bucket1`
	`2022-03-07 01:45:16 amzn-s3-demo-bucket2`
	`2022-03-07 02:12:07 amzn-s3-demo-bucket3`
	• List the files in the S3 bucket.
	`[root@ip-10-0-20-157 ~]# rclone ls s3:amzn-s3-demo-bucket1`
	`template0.yaml`
	`template1.yaml`	

Migrate data using Rclone

Task	Description	Skills required

Migrate data from your containers.	Run the Rclone copy or sync command.
	
	Example: copy
	
	This command copies data from the source Azure Blob container to the destination S3 bucket.
	
	`rclone copy AZStorageAccount:blob-container s3:amzn-s3-demo-bucket1`
	
	Example: sync
	
	This command synchronizes data between the source Azure Blob container and the destination S3 bucket.
	
	`rclone sync AZStorageAccount:blob-container s3:amzn-s3-demo-bucket1`
	
	Important
	When you use the **sync** command, data that isn't present in the source container will be deleted from
		General AWS, Cloud administrator

Task	Description	Skills required
	the destination S3 bucket.	
Synchronize your containers.	After the initial copy is complete, run the Rclone **sync** command for ongoing migration so that only new files that are missing from the destination S3 bucket will be copied.	General AWS, Cloud administr ator
Verify that data has been migrated successfully.	To check that data was successfully copied to the destination S3 bucket, run the Rclone lsd and ls commands.	General AWS, Cloud administr ator

Related resources

- Amazon S3 User Guide (AWS documentation)
- IAM roles for Amazon EC2 (AWS documentation)
- Creating a Microsoft Azure Blob container (Microsoft Azure documentation)
- Rclone commands (Rclone documentation)

Additional information

Example role policy for EC2 instances

This policy gives your EC2 instance read and write access to a specific bucket in your account. If your bucket uses a customer managed key for server-side encryption, the policy might need additional access to AWS Key Management Service (AWS KMS) .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
              "arn:aws:s3:::amzn-s3-demo-bucket/*",
              "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

Creating a read-only Azure AD service principal

An Azure service principal is a security identity that is used by customer applications, services, and automation tools to access specific Azure resources. Think of it as a user identity (login and password or certificate) with a specific role and tightly controlled permissions to access your resources. To create a read-only service principal to follow least privilege permissions and protect data in Azure from accidental deletions, follow these steps:

1. Log in to your Microsoft Azure cloud account portal and launch Cloud Shell in PowerShell or use the Azure Command-Line Interface (CLI) on your workstation.
2. Create a service principal and configure it with [read-only](#) access to your Azure Blob storage account. Save the JSON output of this command to a local file called `azure-principal.json`. The file will be uploaded to your EC2 instance. Replace the placeholder variables that are shown in braces ({ and }) with your Azure subscription ID, resource group name, and storage account name.

   ```
   az ad sp create-for-rbac `
   --name AWS-Rclone-Reader `
   --role "Storage Blob Data Reader" `
   ```

```
--scopes /subscriptions/{Subscription ID}/resourceGroups/{Resource Group Name}/
providers/Microsoft.Storage/storageAccounts/{Storage Account Name}
```

Migrate from Couchbase Server to Couchbase Capella on AWS

Battulga Purevragchaa and Saurabh Shanbhag, Amazon Web Services

Mark Gamble, None

Summary

Couchbase Capella is a fully managed, NoSQL database as a service (DBaaS) for mission-critical applications (for example, user profiles or online catalogs and inventory management). Couchbase Capella manages your DBaaS workload in a Couchbase-managed Amazon Web Services (AWS) account. Capella makes it easy to run and manage multiple-cluster, multiple–AWS Region, multicloud, and hybrid-cloud replication within a single interface.

Couchbase Capella helps you instantly scale your Couchbase Server applications, helping you create multi-node clusters in minutes. Couchbase Capella supports all Couchbase Server features, including SQL++, Full Text Search, Eventing Service, and Analytics Service. It also removes the need to manage installations, upgrades, backups, and general database maintenance.

This pattern describes the steps and best practices for migrating a self-managed Couchbase Server environment to the AWS Cloud. The pattern provides a repeatable process for migrating data and indexes from Couchbase Server clusters, running either on premises or in the cloud, to Couchbase Capella. Using these steps helps you avoid issues during your migration and speeds up your overall migration process.

This pattern provides the following two migration options:

- **Option 1** is appropriate if you have fewer than 50 indexes to migrate.

- **Option 2** is appropriate if you have more than 50 indexes to migrate.

You can also set up sample data on your self-managed Couchbase Server to follow along with the migration guide.

If you choose migration **option 2**, or if you are using scopes or collections other than the default value, you must use the example configuration file, which is in the *Additional information* section.

Prerequisites and limitations

Prerequisites

- An existing Couchbase Capella paid account. You can also create a Couchbase Capella account on AWS and use the Couchbase Capella free trial, and then upgrade to a paid account to configure your cluster for the migration.. To start with the trial version, follow the instructions in Getting Started with Couchbase Capella.
- An existing self-managed Couchbase Server environment either on premises or deployed on a cloud service provider.
- For migration option 2, Couchbase Shell and a configuration file. To create the configuration file, you can use the example file that's in the *Additional information* section.
- Familiarity with administering Couchbase Server and Couchbase Capella.
- Familiarity with opening TCP ports and running commands in a command line interface (CLI).

The migration process also requires the roles and expertise described in the following table.

Role	Expertise	Responsibilities
Couchbase administrator	• Familiarity with Couchbase Server and Couchbase Capella • Basic command line knowledge is helpful but not required	• Couchbase Server and Capella–specific tasks
Systems administrator, IT administrator	• Familiarity with the self-managed Couchbase Server system environment and administration	• Opening ports and determining IP addresses on self-managed Couchbase Server cluster nodes

Limitations

- This pattern is used to migrate data, indexes, and Couchbase Full Text Search indexes from Couchbase Server to Couchbase Capella on AWS. The pattern doesn't apply to migrating Couchbase Eventing Service, or to Couchbase Analytics.
- Couchbase Capella is available in multiple AWS Regions. For up-to-date information on the Regions that Capella supports, see Amazon Web Services in the Couchbase documentation.

Product versions

- [Couchbase Server (Community or Enterprise) Edition version 5.x or later]

Architecture

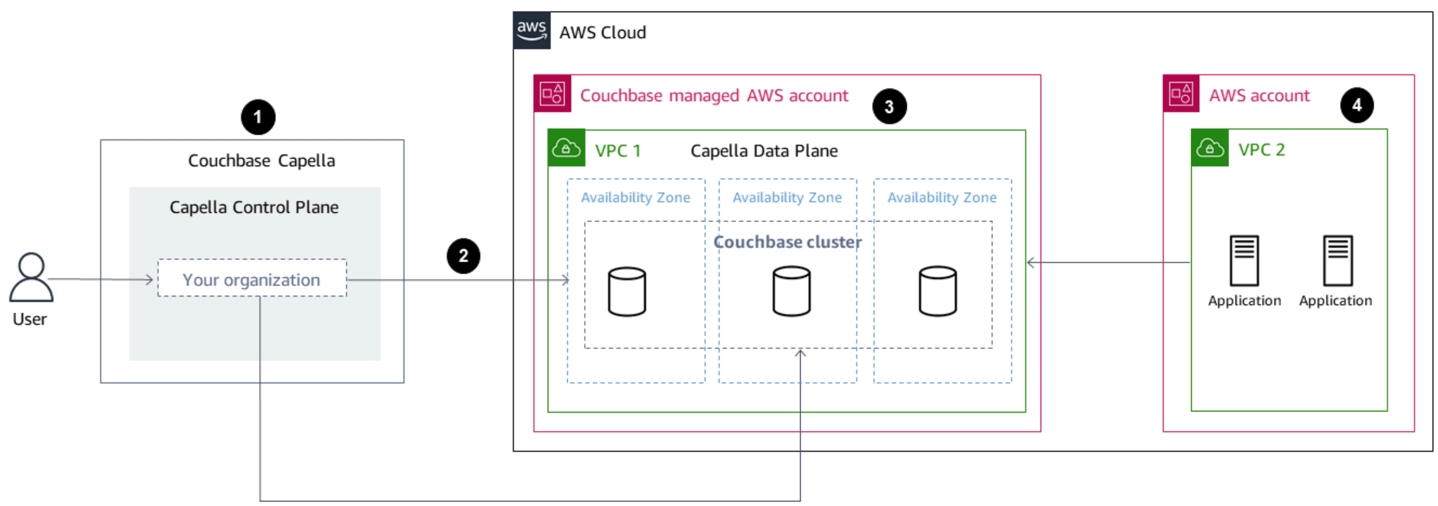
Source technology stack

- Couchbase Server

Target technology stack

- Couchbase Capella

Target architecture



1. You access Couchbase Capella by using the **Capella Control Plane**. You can use the Capella Control Plane to do the following:
   - Control and monitor your account.
   - Manage clusters and data, indexes, users and groups, access permissions, monitoring, and events.
2. Clusters are created.
3. The **Capella Data Plane** is in the Couchbase-managed AWS account. After you create a new cluster, Couchbase Capella deploys it across multiple Availability Zones in the selected AWS Region.

4. You can develop and deploy Couchbase applications in a VPC in your AWS account. Typically, this VPC accesses the Capella Data Plane through VPC peering.

Tools

- Couchbase Cross Data Center Replication (XDCR) helps replicate data across clusters that are located in different cloud providers and different data centers. It is used to migrate data into Couchbase Capella from self-managed Couchbase Server clusters.

> **Note**
>
> XDCR cannot be used with Couchbase Server Community Edition to migrate to Couchbase Capella. Instead, you can use cbexport. For more information, see the *Migrate data from Community Edition* epic.

- Couchbase Shell is a command line shell for Couchbase Server and Couchbase Capella to access local and remote Couchbase clusters. In this pattern, Couchbase Shell is used to migrate indexes.
- cbexport is a Couchbase utility for exporting data from Couchbase cluster. Included in Couchbase Server CLI tools.

Epics

Prepare the migration

Task	Description	Skills required
Evaluate the size of the self-managed Couchbase Server cluster.	Log in to the Couchbase Web Console for Couchbase Server, and assess your self-managed cluster's nodes and buckets. 1. To show a list of cluster nodes, choose the **Servers** tab in the navigation bar. 2. Record the number of nodes, and then choose	Couchbase administrator

Task	Description	Skills required
	each node on the list to display its properties. 3. Record the memory and storage for each individual node. 4. Choose the **Buckets** tab in the navigation bar, and then choose each bucket in the list to display its properties. Record the RAM quota and conflict resolutio n setting for each bucket. You will use your self-mana ged Couchbase Server cluster configurations as a general guide for sizing and configuri ng the target cluster on Couchbase Capella. For help with a more detailed Couchbase Capella sizing exercise, [contact Couchbase]().	

Task	Description	Skills required
Record Couchbase Service distribution on the self-managed Couchbase Server cluster.	1. On the Couchbase Web Console, choose the **Servers** tab to display the list of cluster nodes. 2. Choose each node to display its properties and then record the Couchbase Service distribution for each node (Data Service, Query Service, Index Service, Search Service, Analytics Service, and Eventing Service).	Couchbase administrator
Record the IP addresses of the self-managed Couchbase Server cluster nodes.	(Ignore this step if you are using Community Edition.) Record the IP address for each node in your cluster. They will be added to the allow list on your Couchbase Capella cluster later.	Couchbase administrator, Systems administrator

Deploy and configure resources on Couchbase Capella

Task	Description	Skills required
Choose a template.	1. Log in to your Couchbase Capella Control Plane, choose the **Dashboard** tab or the **Clusters** tab in the main navigation, and then choose **Create Cluster**. 2. Using the information that you recorded from	Couchbase administrator

Task	Description	Skills required
	the evaluation of your self-managed Couchbase Server cluster, choose the cluster template that meets the configura tion's requirements. If you don't find an appropriate template, choose **Custom Template** in the **Cluster Sizing** editor.	
Choose and configure the nodes.	Choose and configure the nodes to match your self-managed Couchbase Server cluster environme nt, including the number of nodes, services distribut ion, compute or RAM, and storage. Couchbase Capella uses multidimensional scaling best practices. Services and nodes can be chosen only according to deployment best practices. This might mean that you can't exactly match your self-managed Couchbase Server cluster's configura tions.	Couchbase administrator

Task	Description	Skills required
Deploy the cluster.	Choose a support zone and support package, and then deploy the cluster. For detailed steps and instructions, see [Create a cluster](#) in the Couchbase documentation. **Important** If you are using the Couchbase Capella free trial, you must convert it to a paid account before beginning your migration. To convert your account, open the **Billing** section of the Couchbase Capella Control Plane, and then choose **Add Activation ID**. The Activation ID is sent to your billing contact email address after you complete a purchase agreement with Couchbase Sales, or after you make a purchase through [AWS Marketplace](#).	Couchbase administrator

Task	Description	Skills required
Create a database credential user.	A database credential user is specific to a cluster and consists of a user name, password, and a set of bucket privileges. This user is required for creating buckets and accessing bucket data. In the Couchbase Capella Control Plane, create a database credential for the new cluster by following the instructions in [Configure database credentials](#) in the Couchbase Capella documentation. **Note** An organization user needs organizat ional role credentia ls assigned to them if they want to access bucket data on a particular cluster, either remotely or through the Couchbase Capella UI. This is separate from database credentia ls, which are typically used by apps and integrations. Creating the organizational	Couchbase administrator

Task	Description	Skills required
	user allows you to create and manage the target buckets on your Couchbase Capella cluster.	
If using migration option 2, install Couchbase Shell.	You can install Couchbase Shell on any system that has network access to both your self-managed Couchbase Server and the Couchbase Capella clusters. For more information, see [Install Couchbase Shell version 1.0.0-beta.5]() in the Couchbase Shell documentation. Confirm that Couchbase Shell is installed by [testing a connection to your self-managed cluster]() in a command line terminal.	Couchbase administrator, Systems administrator

Task	Description	Skills required
Allow IP addresses.	1. In the Couchbase Capella Control Plane, choose **Clusters**, and then choose your target cluster. 2. Choose the **Connect** tab for the cluster, and record the connection endpoint for your cluster that is listed under **Manage Allowed IP**. 3. To add the IP address for the system where you installed Couchbase Shell and the IP address of your self-managed Couchbase Server cluster instances as allowed IP addresses, do the following: a. Under **Wide Area Network**, choose **Manage Allowed IP**. b. Choose **Add Allowed IP**, enter the IP address for the system where you installed Couchbase Shell, and then choose **Add IP**. c. Repeat the previous step to add the IP address of your self-managed Couchbase Server cluster instance.	Couchbase administrator, Systems administrator

Task	Description	Skills required
	For more information about allowed IP addresses, see Configure allowed IP addresses in the Couchbase documentation.	

Task	Description	Skills required
Configure certificates.	1. To download the root certificate for your cluster, under **Root Certificate**, choose **Download**. 2. Save the root certifica te using the .pem file extension in a folder on the system that will run Couchbase Shell. 3. Next, log in to your self-managed Couchbase Server Web Console, choose **Security** in the left navigation bar, and then choose the **Certificates** tab. 4. Copy the root certifica te for your self-managed Couchbase Server cluster and save it as a .pem file to the same folder where you saved the root certificate file for your Couchbase Capella cluster. For more information about the root certifica te, see [Root certificate in the Couchbase Server documentation]().	Couchbase administrator, Systems administrator

Task	Description	Skills required
Create the configuration file for Couchbase Shell.	Create a configuration dotfile in the Couchbase Shell installation's home directory (for example, `/<HOME_DIRECTORY>/.cbsh/config`). For more information, see Config dotfiles in the Couchbase documentation. Add connection properties for the source and target clusters to the configuration file. You can use the example configuration file that's in the *Additional information* section and edit the settings for your clusters. Save the configuration file with the updated settings to the `.cbsh` folder (for example, `/<HOME_DIRECTORY>/.cbsh/config`).	Couchbase administrator, Systems administrator

Task	Description	Skills required
Create target buckets.	For each source bucket, create one target bucket in your Couchbase Capella cluster by following the instructions in [Create a bucket](#) in the Couchbase documentation. Your target bucket configurations must match the bucket names, memory settings, and conflict resolution settings of the buckets in your self-managed Couchbase Server cluster.	Couchbase administrator

Task	Description	Skills required

Create scopes and collections.	Every bucket contains a default scope and collection with the keyspace `_default._default` . If you are using any other keyspaces for your scope and collection, you must create identical keyspaces in the target Capella cluster.
	
	1. Open the command line terminal on the system where you installed Couchbase Shell.
	2. To start Couchbase Shell, run the following command.
	`./cbsh`
	3. For each bucket that you want to migrate, create scopes and collections in the Capella cluster by running the following commands. Make sure that you replace <BUCKET_NAME> with the name of the bucket that you want to migrate.
	
	`scopes --clusters "On-Prem-Cluster" --bucket <BUCKET_NAME> | select scope | where scope !`
		Couchbase administrator

Task	Description	Skills required
	(continued)	

```
= "_default" | each
 { |it| scopes create
 $it.scope --clusters
 "Capella-Cluster" }
collections --cluster
s "On-Prem-Cluster"
 --bucket <BUCKET_N
AME> | select scope
 collection | where
 $it.scope != "_default
" | where $it.colle
ction != "_default" |
 each { |it| collectio
ns create $it.colle
ction --clusters
 "Capella-Cluster" --
bucket <BUCKET_NAME> --
scope $it.scope }
```

Migrate the data from Enterprise Edition

Task	Description	Skills required
Open TCP ports on the self-managed Couchbase Server cluster nodes.	Make sure that the appropriate ports are open for XDCR communication on the self-managed Couchbase Server cluster's nodes. For more information, see the [Couchbase Server ports documentation]().	Couchbase administrator, Systems administrator
If you are using Couchbase Server Enterprise Edition, set up Couchbase XDCR.	1. In the Couchbase Capella Control Plane main navigation, choose **Clusters**, and then choose	Couchbase administrator

Task	Description	Skills required
	the target cluster for migration. 2. Under **Root Certificate**, choose **Copy**. 3. Log in to your self-managed Couchbase Server Web Console, and in the main navigation, choose **XDCR**. Then choose **Add Remote**. 4. Enter the following settings: • **Cluster Name** – A name for the Capella cluster connection • **IP/Hostname** – The connection endpoint for your Couchbase Capella cluster • **Username for Remote Cluster** – The database user for your Couchbase Capella cluster • **Password** – The database user password for your Couchbase Capella cluster • **Enable Secure Connection** – Selected • **Full (TLS encrypt password and data)** – Selected 5. Paste the Capella cluster root certificate you copied	

Task	Description	Skills required
	earlier, and then choose **Save**.	
Start Couchbase XDCR.	1. In your self-managed Couchbase Server Web Console, choose **XDCR** in the main navigation, and then choose **Add Replicati on**. 2. Enter the following settings: • **Replicate From Bucket** – Select the source bucket for migration. • **Remote Bucket** – Enter the target bucket name. • **Remote Cluster** – Select the target cluster you created earlier. 3. Choose **Save Replication**. The replication process should begin within a few seconds.	Couchbase administrator

Migrate the indexes by using option 1

Task	Description	Skills required
Migrate self-managed cluster indexes to Couchbase Capella.	**Important** We recommend this process if you have fewer than 50 indexes to migrate. If you	Couchbase administrator, Systems administrator

Task	Description	Skills required
	have more than 50 indexes to migrate, we recommend that you use migration option 2. 1. On the Couchbase Web Console, choose **Indexes**. 2. In the list of indexes, choose the first index that you want to migrate. The index definition is then displayed. 3. Copy the index definition by using the `CREATE` statement, but don't copy `WITH { "defer_build":true }`. For example, from the following example index definition, you would copy only ``CREATE INDEX `cityindex` ON `travel-sample`(`city`)``. ``CREATE INDEX `cityindex` ON `travel-sample`(`city`) WITH { "defer_build":true }`` 4. In the Couchbase Capella Control Plane, choose	

Task	Description	Skills required
	Clusters, and then choose the target cluster. 5. On the Tools dropdown list, choose **Query Workbench**. Paste the `CREATE` statement that you copied earlier into the **Query Editor**, and then choose **Execute**. This creates and builds the index. 6. To confirm that the index is created, choose **Indexes** from the **Tools** dropdown list. The list shows that the index was created and built. 7. Repeat this process for each index that must be migrated.	

Migrate the indexes by using option 2

Task	Description	Skills required
Migrate the index definitions.	**Important** We recommend this process if you have more than 50 indexes to migrate. If you have fewer than 50 indexes to migrate, we recommend that	Couchbase administrator, Systems administrator

Task	Description	Skills required

	you use migration option 1.
	
	1. Open the command line terminal on the system where you installed Couchbase Shell.
	2. To start Couchbase Shell, run the following command.
	`./cbsh`
	3. To connect to the self-managed Couchbase Server cluster, run the following command.
	`cb-env cluster On-Prem-Cluster`
	4. To migrate index definitions from the self-managed Couchbase Server cluster to the Couchbase Capella cluster, run the following command for each bucket that you want to migrate. Make sure that you replace `<BUCKET_NAME>` with the bucket name that corresponds to the indexes that you want to migrate. This migration option requires that your	

Task	Description	Skills required
	target bucket names are identical to the source bucket names. `query indexes --definitions | where bucket =~ <BUCKET_NAME>| get definition | each { |it| query $it --clusters Capella-Cluster }`	

Task	Description	Skills required

Build the index definitions.	1. To switch context to the Couchbase Capella cluster, run the following command:
	`cb-env cluster Capella-Cluster`
	2. To build the index definitions that were migrated to the Couchbase Capella cluster, run the following command, replacing `<BUCKET_NAME>` with the bucket name that corresponds to the indexes that you want to build.
	`query 'SELECT RAW CONCAT("BUILD INDEX ON ", k , "(['", CONCAT2 ("','", inames), "']);") FROM system:indexes AS s LET bid = CONCAT("` ",s.bucket_id, "`"), sid = CONCAT("`", s.scope_id, "`"), kid = CONCAT("` ", s.keyspace_id, "`"), k = NVL2(bid, CONCAT2(".", bid, sid, kid), kid) WHERE s.namespace_id = "default" AND s.bucket_id = "" GROUP BY k LETTING`
		Couchbase administrator, Systems administrator

Task	Description	Skills required
	` inames = ARRAY_AGG (s.name) FILTER (WHERE s.state = 'deferred') HAVING ARRAY_LENGTH(inames) > 0;' | each { | it| query $it }` 3. Repeat for each bucket.	

Migrate full-text search indexes

Task	Description	Skills required
Migrate self-managed cluster full-text search indexes to Couchbase Capella.	1. In the Couchbase Web Console, choose **Search**. 2. In the list of full-text search (FTS) indexes, choose the first FTS index that you want to migrate, choose **Show index definition JSON,** and choose **Copy to Clipboard**. Make a note of the **index name** and the **bucket** it belongs to. 3. In the Couchbase Capella Control Plane, choose **Clusters** and then choose the target cluster. 4. On the Tools dropdown list, choose **Full Text Search**.	Couchbase administrator

Task	Description	Skills required
	5. Choose **Import Index**, and paste the FTS index definition. 6. Enter the **Index Name**, select the correct **Bucket**, as noted on the self-managed cluster, and then choose **Create**. 7. Repeat this process for each FTS index that must be migrated.	

Migrate data from Couchbase Community Edition

Task	Description	Skills required

Export data from self-managed Couchbase Server Community Edition.	Encrypted XDCR is not available in Couchbase Community Edition. You can export data from Couchbase Community Edition and then manually import the data into Couchbase Capella.
	
	To export data from the source bucket, use `cbexport` at the command line.
	
	The following command is provided as an example.
	
	`cbexport json \`
	`--cluster localhost \`
	`--bucket <SOURCE BUCKET NAME> \`
		Couchbase administrator

Task	Description	Skills required

	<pre>--format lines \
	--username <USERNAME>
	 \
	--password <PASSWORD>
	 \
	--include-key cbkey \
	--scope-field cbscope \
	--collection-field
	 cbcoll \
	--output cbexporte
	d_data.json</pre> Note that `cbkey`, `cbscope`, `cbcoll`, and `cbexported_data.json` are arbitrary labels. They will be referenced later in the process, so if you choose to name them differently, make note of it.	

Task	Description	Skills required
Import data into Couchbase Capella.	1. In the Couchbase Capella Control Plane, choose **Clusters**, and then choose the target cluster. 2. In the **Tools** dropdown list, choose **Import**. This will open a wizard with the following six steps: a. **Bucket** – Choose the target bucket. b. **File** – Choose **JSON**, choose **Lines**, and then choose **Using your web browser**. If you have a large amount of data, you can explore the **Manually** option. Select the file created by `cbexport`. c. **Collections** – Choose **Custom Collection Mapping**. If your Community Edition database does not use scopes or collections, or it uses only **_default**, you can choose the **Select Single Collection** option instead. For **Collection Mapping Expression**, enter	Couchbase administrator

Task	Description	Skills required
	`%cbscope%.%cbcoll%` . To verify that this expression works correctly, you can paste example data, such as the following. `{ "cbscope" :"inventory", "cbcoll":"landmark ", "cbkey":" landmark_3991" }` d. **Key** – Choose **Customer Generation**. (If you don't care about preserving the keys of the data you're importing, you can select **Automatically Generated UUID** instead and proceed to step 5.) For **Key Name Generator Expression**, enter **%cbkey%**. To verify that this expression works correctly, paste some example data. e. **Configurations** – Choose **Ignore fields**, and enter **cbscope,cbcoll,cbkey**. These fields contain transitory information that does not need to be in the target bucket after an import. Leave	

Task	Description	Skills required
	the other settings at their defaults. f. **Import** – Review, and choose **Import** when you're ready. Wait for the upload and data import. For large files, Couchbase Capella supports command line import using cURL. You can explore Import options in more detail at [Import data](#) in the Couchbase Capella documentation.	

Test and verify the migration

Task	Description	Skills required
Verify data migration.	1. In the Couchbase Capella Control Plane, choose **Clusters**, and then choose the target cluster in your cluster list. 2. Choose the **Buckets** tab for your target cluster. Verify that the number of **Items** (documents) in the target bucket matches the number of items in the source bucket.	Couchbase administrator

Task	Description	Skills required
	3. In the target cluster, in the **Tools** dropdown list, choose **Documents**. Verify that all documents were migrated. 4. (Optional) After all data is migrated, you can shut down the replicati on by deleting it. For more information, see [Delete a replication](#) in the Couchbase documentation.	
Verify index migration.	In the Couchbase Capella Control Plane, in the **Tools** dropdown list for your target cluster, choose **Indexes**. Verify that the indexes are migrated and built.	Couchbase administrator
Verify query results.	1. In the Couchbase Capella Control Plane, in the **Tools** dropdown list for your target cluster, choose **Query Workbench**. 2. Run a sample N1QL query or a query used in your application. Make sure that you receive the same results as when you run the query in your self-managed Couchbase Server cluster.	Couchbase administrator

Task	Description	Skills required
Verify full-text search results (applicable if you migrated FTS indexes).	1. In the Couchbase Capella Control Plane, in the **Tools** dropdown list for your target cluster, choose **Full Text Search**. 2. Select an FTS index by choosing its name. 3. Choose **Search**. 4. Enter a sample search query, and choose **Search**. 5. Verify that the results are the same as when you run the search on your self-managed cluster.	Couchbase administrator

Related resources

Prepare the migration

- Get started with the Couchbase Capella free trial
- Cloud provider requirements for Couchbase Capella
- Couchbase Capella sizing guidelines

Migrate the data and indexes

- Couchbase XDCR
- Couchbase Shell documentation

Couchbase Capella SLAs and support

- Couchbase Capella service-level agreements (SLAs)
- Couchbase Capella Service support policy

Additional information

The following code is an example configuration file for Couchbase Shell.

```
Version = 1

[[clusters]]
identifier = "On-Prem-Cluster"
hostnames = ["<SELF_MANAGED_COUCHBASE_CLUSTER>"]
default-bucket = "travel-sample"
username = "<SELF_MANAGED_ADMIN>"
password = "<SELF_MANAGED_ADMIN_PWD>"
tls-cert-path = "/<ABSOLUTE_PATH_TO_SELF_MANAGED_ROOT_CERT>"
data-timeout = "2500ms"
connect-timeout = "7500ms"
query-timeout = "75s"

[[clusters]]
identifier = "Capella-Cluster"
hostnames = ["<COUCHBASE_CAPELLA_ENDPOINT>"]
default-bucket = "travel-sample"
username = "<CAPELLA_DATABASE_USER>"
password = "<CAPELLA_DATABASE_USER_PWD>"
tls-cert-path = "/<ABSOLUTE_PATH_TO_COUCHBASE_CAPELLA_ROOT_CERT>"
data-timeout = "2500ms"
connect-timeout = "7500ms"
query-timeout = "75s"
```

Before you save the configuration file, use the following table to make sure that you added your own source and target cluster information.

<SELF_MANAGED_COUCHBASE_CLUSTER>	Use the IP address for your self-managed Couchbase Server cluster.
<SELF_MANAGED_ADMIN>	Use the administrator user for your self-managed Couchbase Server cluster.
<ABSOLUTE_PATH_TO_SELF_MANAGED_ROOT_CERT>	Use the absolute path to the saved root certificate file for your self-managed Couchbase Server cluster.

<COUCHBASE_CAPELLA_ENDPOINT>	Use the connection endpoint for your Couchbase Capella cluster.
<CAPELLA_DATABASE_USER>	Use the database user for your Couchbase Capella cluster.
<CAPELLA_DATABASE_USER_PWD>	Use the database user password for your Couchbase Capella cluster.
<ABSOLUTE_PATH_TO_COUCHBASE_CAPELLA_ROOT_CERT>	Use the absolute path to the saved root certificate file for your Couchbase Capella cluster.

Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2

Neal Ardeljan and Afroz Khan, Amazon Web Services

Summary

This pattern walks you through the steps for migrating from an on-premises Red Hat Enterprise Linux (RHEL) 6.9 or later system that's running IBM WebSphere Application Server (WAS) to RHEL 8 running Apache Tomcat on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

The pattern can be applied to the following source and target versions:

- WebSphere Application Server 7.x to Apache Tomcat 8 (with Java 7 or later)
- WebSphere Application Server 8.x to Apache Tomcat 8 (with Java 7 or later)
- WebSphere Application Server 8.5.5.x to Apache Tomcat 9 (with Java 8 or later)
- WebSphere Application Server 8.5.5.x to Apache Tomcat 10 (with Java 8 or later)

Prerequisites and limitations

Prerequisites

- An active AWS account
- Source Java code, with the following assumptions:
  - Uses the Java Development Kit (JDK) version of Java 7 or later
  - Uses the Spring or Apache Struts framework
  - Doesn't use the Enterprise Java Beans (EJB) framework or any other WebSphere server functionality that's not readily available for Tomcat
  - Primarily uses servlets or Java Server Pages (JSPs)
  - Uses Java Database Connectivity (JDBC) connectors to connect to databases
- Source IBM WebSphere Application Server version 7.x or higher
- Target Apache Tomcat version 8.5 or higher

Architecture

Source technology stack

- A web application built using the Apache Struts Model-View-Controller (MVC) framework
- A web application running on IBM WebSphere Application Server version 7.x or 8.x
- A web application that uses a Lightweight Directory Access Protocol (LDAP) connector to connect to an LDAP directory (iPlanet/eTrust)
- An application that uses IBM Tivoli Access Manager (TAM) connectivity to update the TAM user password (in the present implementation, applications use PD.jar)

On-premises databases

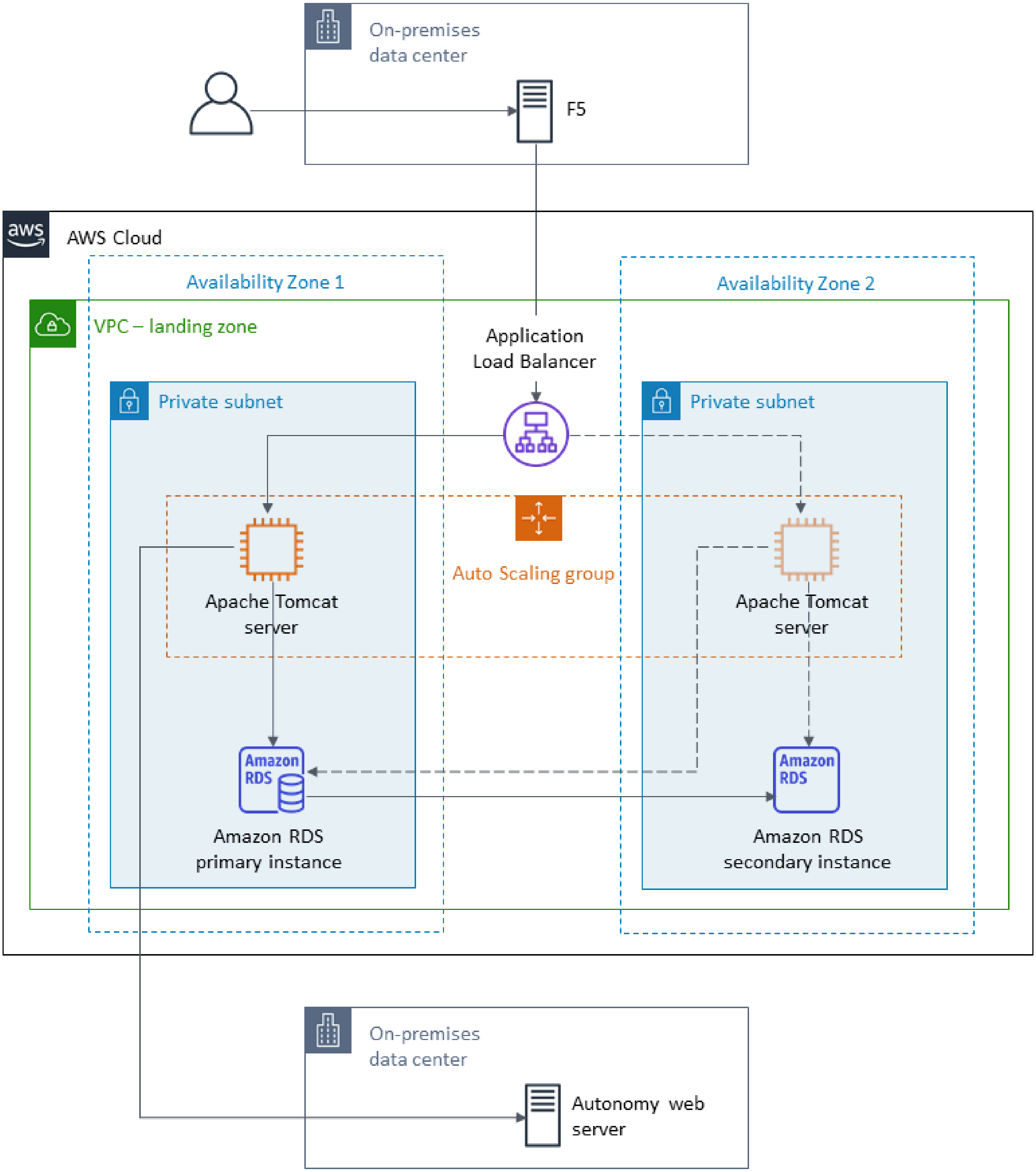
- Oracle Database 21c (21.0.0.0)
- Oracle Database 19c (19.0.0.0)
- Oracle Database 12c Release 2 (12.2.0.1)
- Oracle Database 12c Release 1 (12.1.0.2)

Target technology stack

- Apache Tomcat version 8 (or later) running on RHEL on an EC2 instance
- Amazon Relational Database Service (Amazon RDS) for Oracle

For more information about the Oracle versions supported by Amazon RDS, see the [Amazon RDS for Oracle](#) website.

Target architecture


On-premises
data center
F5
AWS Cloud
Availability Zone 1
Availability Zone 2
VPC – landing zone
Application
Load Balancer
Private subnet
Private subnet
Auto Scaling group
Apache Tomcat
server
Apache Tomcat
server
Amazon RDS
Amazon RDS
primary instance
Amazon RDS
Amazon RDS
secondary instance
On-premises
data center
Autonomy web
server

Tools

- Application tier: Rebuilding Java application into a WAR file.
- Database tier: Oracle native backup and restore.
- Apache Tomcat migration tool for Jakarta EE. This tool takes a web application written for Java EE 8 that runs on Apache Tomcat 9 and converts it automatically to run on Apache Tomcat 10, which implements Jakarta EE 9.

Epics

Plan the migration

Task	Description	Skills required
Complete the application discovery, current state footprint, and performance baseline.		BA, Migration Lead
Validate the source and target database versions.		DBA
Identify the hardware requirements for the target server EC2 instance.		DBA, SysAdmin
Identify storage requirements (storage type and capacity).		DBA, SysAdmin
Choose the proper EC2 instance type based on capacity, storage features, and network features.		DBA, SysAdmin
Identify the network access security requirements for the source and target databases.		DBA, SysAdmin

Task	Description	Skills required
Identify the application migration strategy and tooling.		DBA, Migration Lead
Complete the migration design and migration guide for the application.		Build Lead, Migration Lead
Complete the application migration runbook.		Build Lead, Cutover Lead, Testing Lead, Migration Lead

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC).		SysAdmin
Create the security groups.		SysAdmin
Configure and start Amazon RDS for Oracle.		DBA, SysAdmin

Migrate data

Task	Description	Skills required
Create or obtain access to the endpoints to fetch the database backup files.		DBA
Use the native database engine or a third-party tool to migrate database objects and data.	For details, see "Migrating database objects and data" in the *Additional information* section.	DBA

Migrate the application

Task	Description	Skills required
Lodge the change request (CR) for migration.		Cutover Lead
Obtain the CR approval for migration.		Cutover Lead
Follow the application migration strategy per the application migration runbook.	For details, see "Setting up the application tier" in the *Additional information* section.	DBA, Migration Engineer, App owner
Upgrade the application (if necessary).		DBA, Migration Engineer, App owner
Complete the functional, non-functional, data validation, SLA, and performance tests.		Testing Lead, App Owner, App Users

Cut over

Task	Description	Skills required
Obtain signoff from the application owner or business owner.		Cutover Lead
Switch the application clients to the new infrastructure.		DBA, Migration Engineer, App owner

Close the project

Task	Description	Skills required
Shut down temporary AWS resources.		DBA, Migration Engineer, SysAdmin
Review and validate the project documents.		Migration Lead
Gather metrics such as time to migrate, percentage of manual versus automated tasks, and cost savings.		Migration Lead
Close out the project and provide feedback.		Migration Lead, App Owner

Related resources

References

- Apache Tomcat 10.0 documentation
- Apache Tomcat 9.0 documentation
- Apache Tomcat 8.0 documentation
- Apache Tomcat 8.0 installation guide
- Apache Tomcat JNDI documentation
- Amazon RDS for Oracle website
- Amazon RDS pricing
- Oracle and Amazon Web Services
- Oracle on Amazon RDS
- Amazon RDS Multi-AZ Deployments

Tutorials and videos

- Getting Started with Amazon RDS

Additional information

Migrating database objects and data

For example, if you're using native Oracle backup/restore utilities:

1. Create the Amazon Simple Storage Service (Amazon S3) backup for database backup files (optional).
2. Back up the Oracle DB data to the network shared folder.
3. Log in to the migration staging server to map the network share folder.
4. Copy data from the network share folder to the S3 bucket.
5. Request an Amazon RDS Multi-AZ deployment for Oracle.
6. Restore the on-premises database backup to Amazon RDS for Oracle.

Setting up the application tier

1. Install Tomcat 8 (or 9/10) from the Apache Tomcat website.
2. Package the application and shared libraries into a WAR file.
3. Deploy the WAR file in Tomcat.
4. Monitor the start log to `Linux cat` any missing shared libraries from WebSphere.
5. Watch the start record to `Linux cat` any WebSphere-specific deployment descriptor extensions.
6. Collect any missing dependent Java libraries from the WebSphere server.
7. Amend WebSphere-specific deployment descriptor elements with Tomcat-compatible equivalents.
8. Rebuild the WAR file with the dependent Java libraries and updated deployment descriptors.
9. Update the LDAP configuration, database configuration, and test connections (see Realm Configuration HOW-TO and JNDI Datasource HOW-TO in the Apache Tomcat documentation).
10. Test the installed application against the restored Amazon RDS for Oracle database.
11. Create an Amazon Machine Image (AMI) for Linux from the EC2 instance.
12. Launch the completed architecture with the Application Load Balancer and Auto Scaling group.
13. Update the URLs (by using the WebSEAL junction) to point to the Application Load Balancer.
14. Update the configuration management database (CMDB).

Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling

Kevin Yung and Afroz Khan, Amazon Web Services

Summary

This pattern provides guidance for migrating a Java application from IBM WebSphere Application Server to Apache Tomcat on an Amazon Elastic Compute Cloud (Amazon EC2) instance with Amazon EC2 Auto Scaling enabled.

By using this pattern, you can achieve:

- A reduction in IBM licensing costs
- High availability using Multi-AZ deployment
- Improved application resiliency with Amazon EC2 Auto Scaling

Prerequisites and limitations

Prerequisites

- Java applications (version 7.*x* or 8.*x*) should be developed in LAMP stacks.
- The target state is to host Java applications on Linux hosts. This pattern has been successfully implemented in a Red Hat Enterprise Linux (RHEL) 7 environment. Other Linux distributions can follow this pattern, but the configuration of the Apache Tomcat distribution should be referenced.
- You should understand the Java application's dependencies.
- You must have access to the Java application source code to make changes.

Limitations and replatforming changes

- You should understand the enterprise archive (EAR) components and verify that all libraries are packaged in the web component WAR files. You need to configure the [Apache Maven WAR Plugin] and produce WAR file artifacts.
- When using Apache Tomcat 8, there is a known conflict between servlet-api.jar and the application package built-in jar files. To resolve this issue, delete servlet-api.jar from the application package.

- You must configure WEB-INF/resources located in the *classpath* of the Apache Tomcat configuration. By default, the JAR libraries are not loaded in the directory. Alternatively, you can deploy all the resources under src/main/resources.
- Check for any hard-coded context roots within the Java application, and update the new context root of Apache Tomcat.
- To set JVM runtime options, you can create the configuration file setenv.sh in the Apache Tomcat bin folder; for example, JAVA_OPTS, JAVA_HOME**,** etc.
- Authentication is configured at the container level and is set up as a realm in Apache Tomcat configurations. Authentication is established for any of the following three realms:
  - JDBC Database Realm looks up users in a relational database accessed by the JDBC driver.
  - DataSource Database Realm looks up users in a database that is accessed by JNDI.
  - JNDI Directory Realm looks up users in the Lightweight Directory Access Protocol (LDAP) directory that is accessed by the JNDI provider. The look-ups require:
    - LDAP connection details: user search base, search filter, role base, role filter
    - The key JNDI Directory Realm: Connects to LDAP, authenticates users, and retrieves all groups in which a user is a member
- Authorization: In the case of a container with a role-based authorization that checks the authorization constraints in web.xml, web resources must be defined and compared to the roles defined in the constraints. If LDAP doesn't have group-role mapping, you must set the attribute <security-role-ref> in web.xml to achieve group-role mapping. To see an example of a configuration document, see the Oracle documentation.
- Database connection: Create a resource definition in Apache Tomcat with an Amazon Relational Database Service (Amazon RDS) endpoint URL and connection details. Update the application code to reference a DataSource by using JNDI lookup. An existing DB connection defined in WebSphere would not work, as it uses WebSphere's JNDI names. You can add a <resource-ref> entry in web.xml with the JNDI name and DataSource type definition. To see a sample configuration document, see the Apache Tomcat documentation.
- Logging: By default, Apache Tomcat logs to the console or to a log file. You can enable realm-level tracing by updating *logging.properties* (see Logging in Tomcat). If you are using Apache Log4j to append logs to a file, you must download tomcat-juli and add it to the *classpath*.
- Session management: If you are retaining IBM WebSEAL for application load balancing and session management, no change is required. If you are using an Application Load Balancer or Network Load Balancer on AWS to replace the IBM WebSEAL component, you must set up

session management by using an Amazon ElastiCache instance with a Memcached cluster and set up Apache Tomcat to use open-source session management.

- If you are using the IBM WebSEAL forward proxy, you must set up a new Network Load Balancer on AWS. Use the IPs provided by the Network Load Balancer for WebSEAL junction configurations.
- SSL configuration: We recommend that you use Secure Sockets Layer (SSL) for end-to-end communications. To set up an SSL server configuration in Apache Tomcat, follow the instructions in the Apache Tomcat documentation.

Architecture

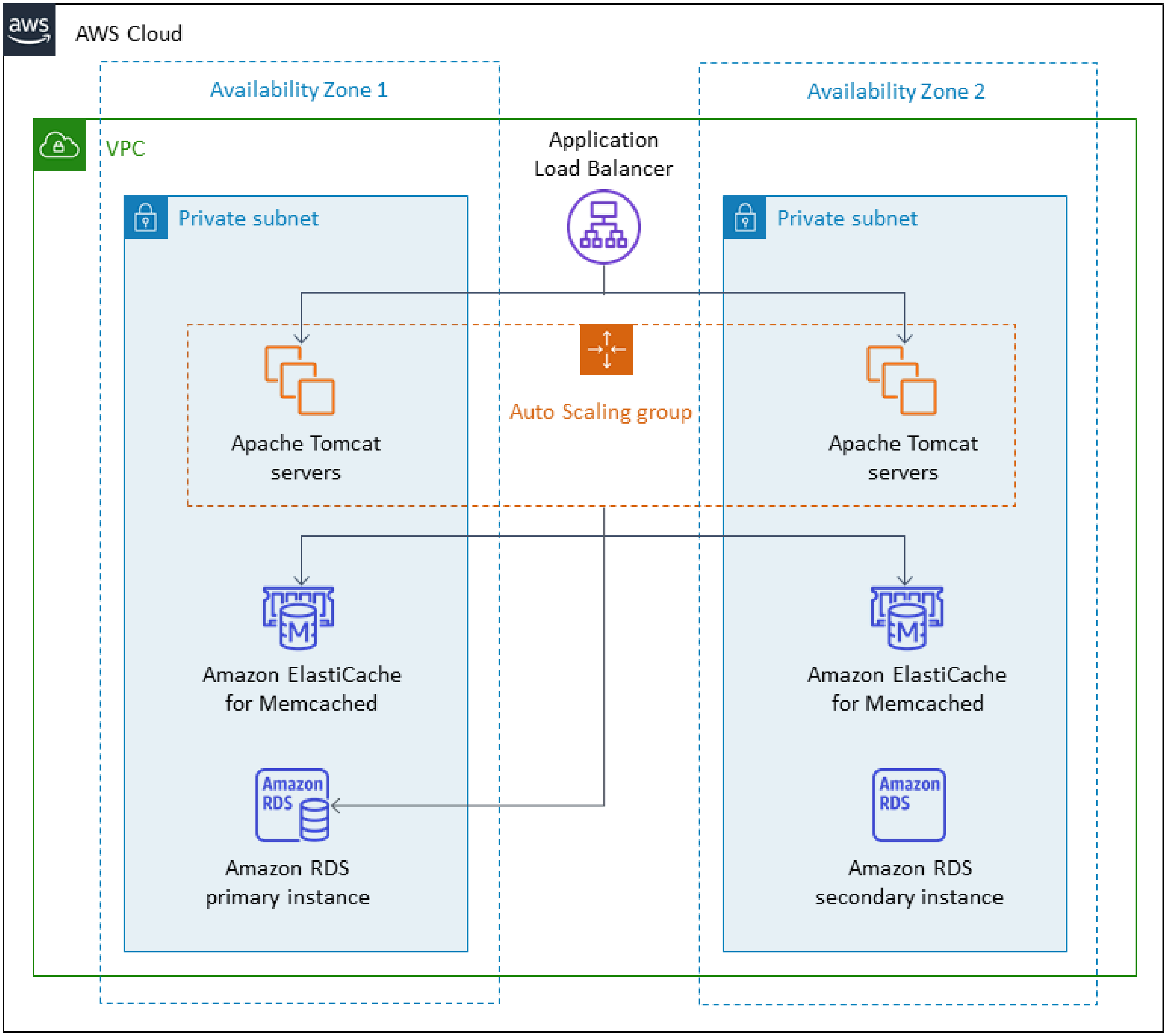
Source technology stack

- IBM WebSphere Application Server

Target technology stack

- The architecture uses Elastic Load Balancing (version 2). If you are using IBM WebSEAL for Identify management and load balancing, you can select a Network Load Balancer on AWS to integrate with the IBM WebSEAL reverse proxy.
- Java applications are deployed to an Apache Tomcat application server, which runs on an EC2 instance in an Amazon EC2 Auto Scaling group. You can set up a scaling policy based on Amazon CloudWatch metrics such as CPU utilization.
- If you're retiring the use of IBM WebSEAL for load balancing, you can use Amazon ElastiCache for Memcached for session management.
- For the back-end database, you can deploy High Availability (Multi-AZ) for Amazon RDS and select a database engine type.

Target architecture



Tools

- [AWS CloudFormation]
- [AWS Command Line Interface (AWS CLI)]
- Apache Tomcat (version 7.*x* or 8.*x*)
- RHEL 7 or Centos 7
- [Amazon RDS Multi-AZ deployment]
- [Amazon ElastiCache for Memcached] (optional)

Epics

Set up the VPC

Task	Description	Skills required
Create a virtual private cloud (VPC).		
Create subnets.		
Create routing tables if necessary.		
Create network access control lists (ACLs).		
Set up AWS Direct Connect or a corporate VPN connection.		

Replatform the application

Task	Description	Skills required
Refactor the application build Maven configuration to generate the WAR artifacts.		
Refactor the application dependency data sources in Apache Tomcat.		
Refactor the application source codes to use JNDI names in Apache Tomcat.		
Deploy the WAR artifacts into Apache Tomcat.		

Task	Description	Skills required
Complete application validations and tests.		

Configure the network

Task	Description	Skills required
Configure the corporate firewall to allow the connection to dependency services.		
Configure the corporate firewall to allow end-user access to Elastic Load Balancing on AWS.		

Create the application infrastructure

Task	Description	Skills required
Create and deploy the application on an EC2 instance.		
Create an Amazon ElastiCac he for Memcached cluster for session management.		
Create an Amazon RDS Multi-AZ instance for the backend database.		

Task	Description	Skills required
Create SSL certificates and import them into AWS Certificate Manager (ACM).		
Install SSL certificates on load balancers.		
Install SSL certificates for Apache Tomcat servers.		
Complete application validations and tests.		

Cut over

Task	Description	Skills required
Shut down the existing infrastructure.		
Restore the database from production to Amazon RDS.		
Cut over the application by making DNS changes.		

Related resources

References

- Apache Tomcat 7.0 documentation
- Apache Tomcat 7.0 installation guide
- Apache Tomcat JNDI documentation
- Amazon RDS Multi-AZ Deployments
- Amazon ElastiCache for Memcached

Tutorials and videos

- Getting Started with Amazon RDS

Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk

Raghavender Madamshitti, Amazon Web Services

Summary

This pattern describes how to migrate a .NET web application hosted on Microsoft Azure App Service to AWS Elastic Beanstalk. There are two ways to migrate applications to Elastic Beanstalk:

- Use AWS Toolkit for Visual Studio - This plugin for the Microsoft Visual Studio IDE provides the easiest and most straightforward way to deploy custom .NET applications to AWS. You can use this approach to deploy .NET code directly to AWS and to create supporting resources, such as Amazon Relational Database Service (Amazon RDS) for SQL Server databases, directly from Visual Studio.
- Upload and deploy to Elastic Beanstalk - Each Azure App Service includes a background service called Kudu, which is useful for capturing memory dumps and deployment logs, viewing configuration parameters, and accessing deployment packages. You can use the Kudu console to access Azure App Service content, extract the deployment package, and then upload the package to Elastic Beanstalk by using the upload and deploy option in the Elastic Beanstalk console.

This pattern describes the second approach (uploading your application to Elastic Beanstalk through Kudu). The pattern also uses the following AWS services: AWS Elastic Beanstalk, Amazon Virtual Private Cloud (Amazon VPC), Amazon CloudWatch, Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling, Amazon Simple Storage Service (Amazon S3), and Amazon Route 53.

The .NET web application is deployed to AWS Elastic Beanstalk, which runs in an Amazon EC2 Auto Scaling Group. You can set up a scaling policy based on Amazon CloudWatch metrics such as CPU utilization. For a database, you can use Amazon RDS in a Multi-AZ environment, or Amazon DynamoDB, depending on your application and business requirements.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A .NET web application running in Azure App Service
- Permission to use the Azure App Service Kudu console

Product versions

- .NET Core (x64) 1.0.1, 2.0.0, or later, or .NET Framework 4.x, 3.5 (see .NET on Windows Server platform history)
- Internet Information Services (IIS) version 8.0 or later, running on Windows Server 2012 or later
- .NET 2.0 or 4.0 Runtime.

Architecture

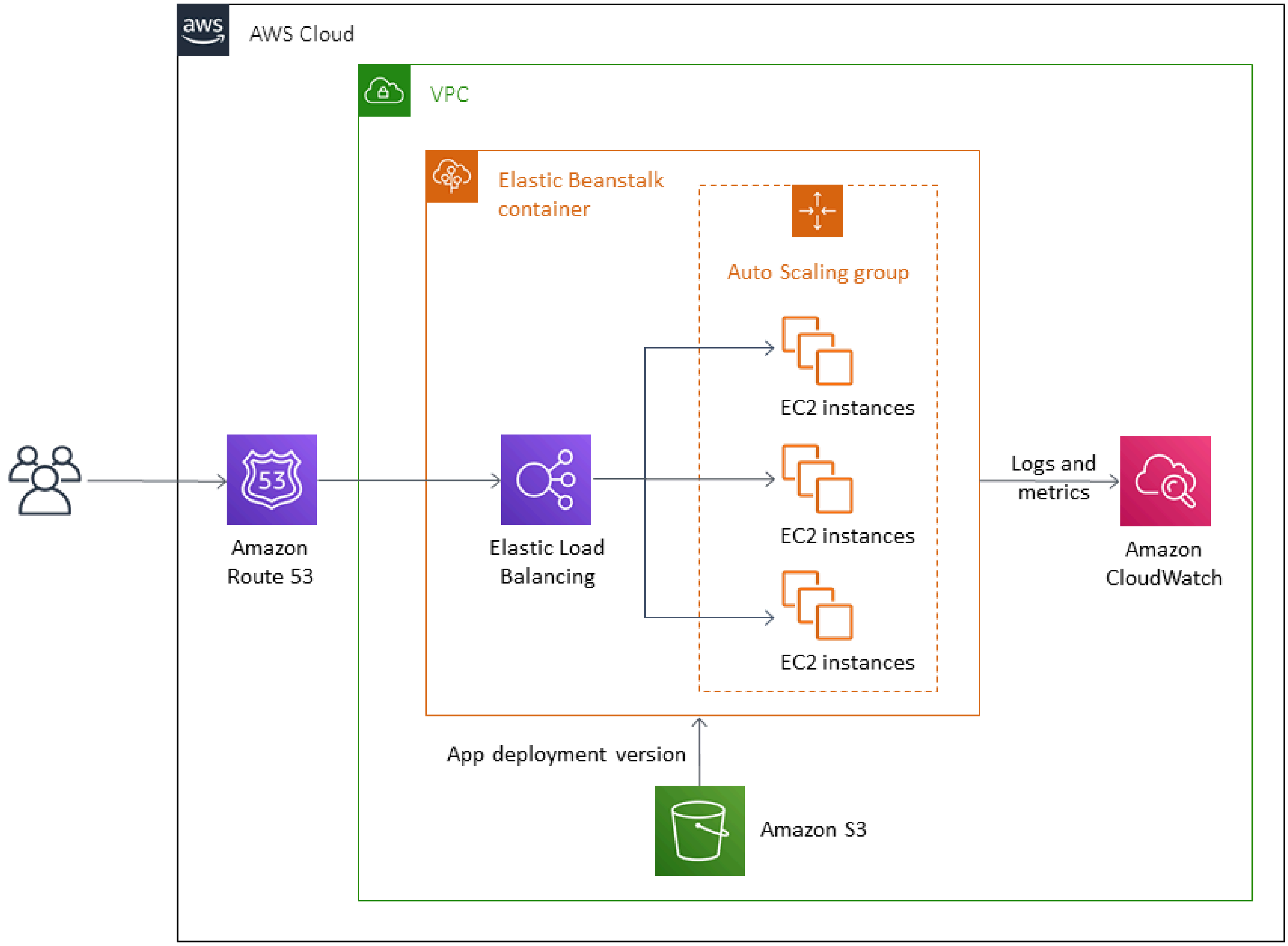
Source technology stack

- Application developed using .NET Framework 3.5 or later, or .NET Core 1.0.1, 2.0.0, or later, and hosted on Azure App Service (web app or API app)

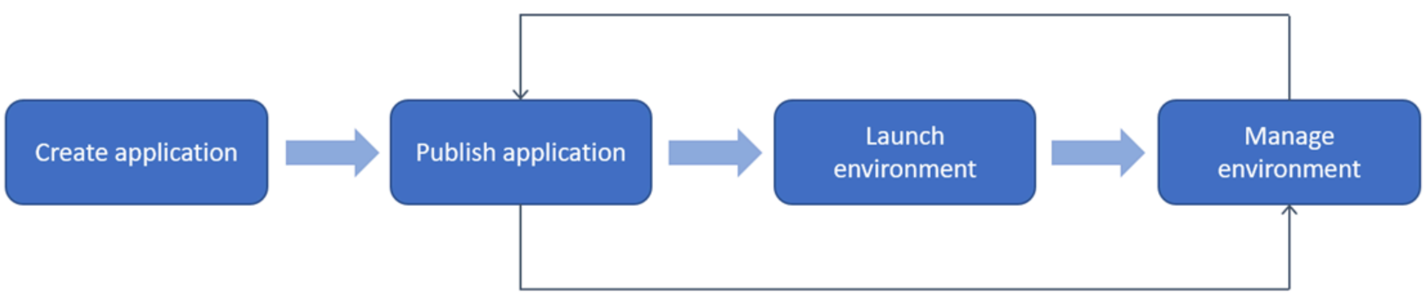
Target technology stack

- AWS Elastic Beanstalk running in an Amazon EC2 Auto Scaling group

Migration architecture



Deployment workflow



Tools

Tools

- .NET Core or .NET Framework

- C#
- IIS
- Kudu console

AWS services and features

- AWS Elastic Beanstalk – Elastic Beanstalk is an easy-to-use service for deploying and scaling .NET web applications. Elastic Beanstalk automatically manages capacity provisioning, load balancing, and auto scaling.
- Amazon EC2 Auto Scaling group – Elastic Beanstalk includes an Auto Scaling group that manages the Amazon EC2 instances in the environment. In a single-instance environment, the Auto Scaling group ensures that there is always one instance running. In a load-balanced environment, you can configure the group with a range of instances to run, and Amazon EC2 Auto Scaling adds or removes instances as needed, based on load.
- Elastic Load Balancing – When you enable load balancing in AWS Elastic Beanstalk, it creates a load balancer that distributes traffic among the EC2 instances in the environment.
- Amazon CloudWatch – Elastic Beanstalk automatically uses Amazon CloudWatch to provide information about your application and environment resources. Amazon CloudWatch supports standard metrics, custom metrics, and alarms.
- Amazon Route 53 – Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. You can use Route 53 alias records to map custom domain names to AWS Elastic Beanstalk environments.

Epics

Set up a VPC

Task	Description	Skills required
Set up a virtual private cloud (VPC).	In your AWS account, create a VPC with the required information.	System administrator
Create subnets.	Create two or more subnets in your VPC.	System administrator

Task	Description	Skills required
Create a route table.	Create a route table, based on your requirements.	System administrator

Set up Elastic Beanstalk

Task	Description	Skills required
Access the Azure App Service Kudu console.	Access Kudu through the Azure portal by navigating to the App Service dashboard, and then choosing **Advanced Tools**, **Go**. Or, you can modify the Azure App Service URL as follows: `https://<appservicename>.scm.azurewebsites.net` .	App developer, System administrator
Download the deployment package from Kudu.	Navigate to Windows PowerShell by choosing the **DebugConsole** option. This will open the Kudo console. Go to the `wwwroot` folder and download it. This will download the Azure App Service deployment package as a zip file. For an example, see the attachment.	App developer, System administrator
Create a package for Elastic Beanstalk.	Unzip the deployment package that you downloaded from Azure App Service. Create a JSON file called `aws-windows-deployment-manifest.json`	App developer, System administrator

Task	Description	Skills required
	(this file is required only for .NET Core applications). Create a zip file that includes `aws-windows-deployment-manifest.json` and the Azure App Service deployment package file. For an example, see the attachment.	
Create a new Elastic Beanstalk application.	Open the Elastic Beanstalk console. Choose an existing application or create a new application.	App developer, System administrator
Create the environment.	In the Elastic Beanstalk console **Actions** menu, choose **Create environment**. Select the web server environment and .NET/IIS platform. For application code, choose **Upload**. Upload the zip file that you prepared for Elastic Beanstalk, and then choose **Create Environment**.	App developer, System administrator
Configure Amazon CloudWatch.	By default, basic CloudWatch monitoring is enabled. If you want to change the configuration, in the Elastic Beanstalk wizard, choose the published application, and then choose **Monitoring**.	System administrator

Task	Description	Skills required
Verify that the deployment package is in Amazon S3.	When the application environment has been created, you can find the deployment package in the S3 bucket.	App developer, System administrator
Test the application.	When the environment has been created, use the URL provided in the Elastic Beanstalk console to test the application.	System administrator

Related resources

- AWS Elastic Beanstack concepts (Elastic Beanstalk documentation)
- Getting Started with .NET on Elastic Beanstalk (Elastic Beanstalk documentation)
- Kudu console (GitHub)
- Using "Kudu" to Manage Azure Web Apps (GS Lab article)
- Custom ASP.NET Core Elastic Beanstalk Deployments (AWS Toolkit for Visual Studio user guide)
- Elastic Load Balancing documentation
- AWS Elastic Beanstalk Supported Platforms (Elastic Beanstalk documentation)
- Deploy a Web Application to AWS (C# Corner article)
- Scaling the Size of Your Auto Scaling Group (Amazon EC2 documentation)
- High Availability (Multi-AZ) for Amazon RDS (Amazon RDS documentation)

Additional information

Notes

- If you are migrating an on-premises or Azure SQL Server database to Amazon RDS, you must update the database connection details as well.
- For testing purposes, a sample demo application is attached.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS

Anya Epishcheva and Harshad Gohil, Amazon Web Services

Summary

This pattern discusses the steps for migrating an on-premises Oracle Solaris SPARC system running Oracle WebLogic to a Docker container-based installation running [Apache TomEE](https://tomee.apache.org/) (Apache Tomcat with added container support) with Amazon Elastic Container Service (Amazon ECS).

For information about migrating databases that are associated with the applications you are migrating from Oracle WebLogic to Tomcat, see the database migration patterns in this catalog.

Best practices

Steps for migrating Java and Java Enterprise Edition (Java EE) web applications vary, depending on the number of container-specific resources used by the application. Spring-based applications are typically easier to migrate, because they have a small number of dependencies on the deployment container. In contrast, Java EE applications that use Enterprise JavaBeans (EJBs) and managed container resources such as thread pools, Java Authentication and Authorization Service (JAAS), and Container-Managed Persistence (CMP) require more effort.

Applications developed for Oracle Application Server frequently use the Oracle Identity Management suite. Customers migrating to open-source application servers frequently choose to re-implement identity and access management using SAML-based federation. Others use Oracle HTTP Server Webgate for cases when migrating from the Oracle Identity Management suite isn't an option.

Java and Java EE web applications are great candidates for deployment on AWS services that are Docker-based, such as AWS Fargate and Amazon ECS. Customers frequently choose a Docker image with the latest version of the target application server (such as TomEE) and the Java Development Kit (JDK) pre-installed. They install their applications on top of the base Docker image, publish it in their Amazon Elastic Container Registry (Amazon ECR) registry, and use it for scalable deployment of their applications on AWS Fargate or Amazon ECS.

Ideally, application deployment is elastic; that is, the number of application instances scales in or out, depending on traffic or workload. This means that application instances need to come online or be terminated to adjust capacity for demand.

When moving a Java application to AWS, consider making it stateless. This is a key architectural principle of the AWS Well-Architected Framework that will enable horizontal scaling using containerization. For example, most Java-based web applications store user-session information locally. To survive application instance termination due to automatic scaling in Amazon Elastic Compute Cloud (Amazon EC2) or for other reasons, user-session information should be stored globally so that web application users can continue to work seamlessly and transparently without reconnecting or relogging into a web application. There are several architectural options for this approach, including Amazon ElastiCache for Redis, or storing session state in a global database. Application servers such as TomEE have plugins, which enable session storage and management via Redis, databases, and other global data stores.

Use a common, centralized logging and debugging tool that is easily integrated with Amazon CloudWatch and AWS X-Ray. Migration provides an opportunity to improve application lifecycle capabilities. For example, you might want to automate the build process so that changes are easily made using a continuous integration and continuous delivery (CI/CD) pipeline. This may require changes to the application so that it can be deployed without any downtime.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Source Java code and JDK
- Source application built with Oracle WebLogic
- Defined solution for identity and access management (SAML or Oracle Webgate)
- Defined solution for application session management (moving like-for-like or with Amazon ElastiCache, or making the application stateless if needed)
- Understanding if the team needs to refactor J2EE-specific libraries for portability to Apache TomEE (see Java EE 7 Implementation Status on the Apache website)
- Hardened TomEE image based on your security requirements
- Container image with pre-installed target TomEE
- Application remediation agreed and implemented if needed (for example, logging debug build, authentication)

Product versions

- Oracle WebLogic OC4J, 9i, 10g
- Tomcat 7 (with Java 1.6 or later)

Architecture

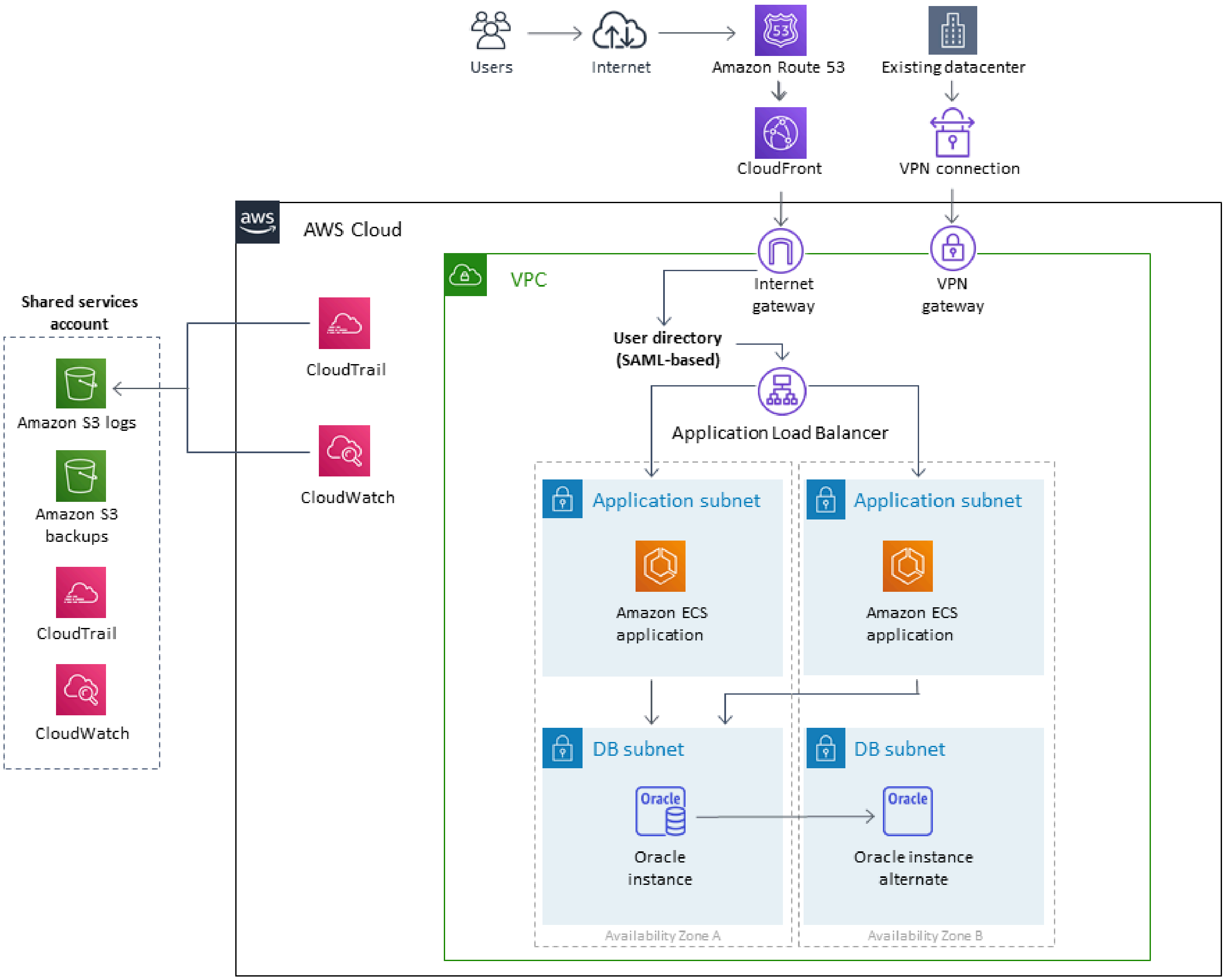
Source technology stack

- Web application built using Oracle WebLogic
- Web application using Oracle Webgate or SAML authentication
- Web applications connected to Oracle Database version 10g and later

Target technology stack

- TomEE (Apache Tomcat with added container support) running on Amazon ECS (see also Deploying Java Web Applications and Java Microservices on Amazon ECS)
- Amazon Relational Database Service (Amazon RDS) for Oracle; for Oracle versions supported by Amazon RDS, see Amazon RDS for Oracle

Target architecture



Tools

To operate on TomEE, a Java application must be rebuilt into a .war file. In some cases, application changes may be required to operate the application on TomEE; you should check to make sure that the necessary configuration options and environment properties are defined correctly.

Also, Java Naming and Directory Interface (JNDI) lookups and JavaServer Pages (JSP) namespaces should be defined correctly. Consider checking file names used by the application to avoid naming collisions with built-in T libraries. For example, persistence.xml is a file name used by the Apache OpenJPA framework (which is bundled with OpenEJB in TomEE) for configuration purposes. The persistence.xml file in PUI contains Spring framework bean declarations.

TomEE version 7.0.3 and later (Tomcat 8.5.7 and later) returns an HTTP 400 response (bad request) for raw (unencoded) URLs with special characters. The server response appears as a blank page to the end-user. Earlier versions of TomEE and Tomcat allowed the use of certain unencoded special characters in URLs; however, it's considered unsafe, as stated on the CVE-2016-6816 website. To resolve the URL encoding issue, the URLs passed to the browser directly via JavaScript must be encoded with the **encodeURI()** method instead of being used as raw strings.

After you deploy the .war file in TomEE, monitor *start log* to *Linux cat* for any missing shared libraries and Oracle-specific extensions to add missing components from Tomcat libraries.

General procedure

- Configure the application on TomEE.
- Identify and reconfigure application server-specific configuration files and resources from source to target format.
- Identify and reconfigure JNDI resources.
- Adjust EJB namespace and lookups to the format required by the target application server (if applicable).
- Reconfigure JAAS application container-specific security roles and principle mappings (if applicable).
- Package the application and shared libraries into a .war file.
- Deploy the .war file in TomEE by using the Docker container provided.
- Monitor *start log* to identify any missing shared library and deployment descriptor extensions. If any are found, go back to the first task.
- Test the installed application against the restored Amazon RDS database.
- Launch the complete architecture with a load balancer and Amazon ECS cluster by following the instructions in Deploy Docker Containers.
- Update the URLs to point to the load balancer.
- Update the configuration management database (CMDB).

Epics

Plan the migration

Task	Description	Skills required
Perform application discovery (current state footprint and performance baseline).		BA, Migration Lead
Validate source and target database versions and engines.		DBA
Validate the source and target application design (identity and session management).		DBA, Migration Engineer, App owner
Identify hardware and storage requirements for the target server instance.		DBA, SysAdmin
Choose the proper instance type based on capacity, storage features, and network features.		DBA, SysAdmin
Identify network access security requirements for the source and target databases.		DBA, SysAdmin
Identify application migration strategy and tooling.		DBA, Migration Lead
Complete the migration design and migration guide for the application.		Build Lead, Migration Lead

Task	Description	Skills required
Complete the application migration runbook.		Build Lead, Cutover Lead, Testing Lead, Migration Lead

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC).		SysAdmin
Create security groups.		SysAdmin
Configure and start the Amazon RDS DB instance.		DBA, SysAdmin
Configure Amazon ECS deployment.		SysAdmin
Package your application as a Docker image.		SysAdmin
Push the image to the Amazon ECR registry (or skip this step and push it to the Amazon ECS cluster).		SysAdmin
Configure the task definitio n for the application and Amazon ECS service options.		SysAdmin
Configure your cluster, review security settings, and set AWS Identity and Access Management (IAM) roles.		SysAdmin

Task	Description	Skills required
Launch your setup and run tests according to your application migration runbook.		SysAdmin

Migrate data

Task	Description	Skills required
Get your security assurance team's permission to move production data to AWS.		DBA, Migration Engineer, App owner
Create and obtain access to endpoints to fetch database backup files.		DBA
Use the native database engine or third-party tools to migrate database objects and data.		DBA
Run necessary tests from the application migration runbook to confirm successful data migration.		DBA, Migration Engineer, App owner

Migrate the application

Task	Description	Skills required
Create a change request (CR) for migration.		Cutover Lead

Task	Description	Skills required
Obtain CR approval for migration.		Cutover Lead
Follow the application migration strategy from the application migration runbook.		DBA, Migration Engineer, App owner
Upgrade the application (if required).		DBA, Migration Engineer, App owner
Complete functional, non-functional, data validation, SLA, and performance tests.		Testing Lead, App Owner, App Users

Cut over

Task	Description	Skills required
Obtain signoff from the application or business owner.		Cutover Lead
Run a table topic exercise to walk through all the steps of the cutover runbook.		DBA, Migration Engineer, App owner
Switch application clients to the new infrastructure.		DBA, Migration Engineer, App owner

Close the project

Task	Description	Skills required
Shut down temporary AWS resources.		DBA, Migration Engineer, SysAdmin

Task	Description	Skills required
Review and validate the project documents.		Migration Lead
Gather metrics around time to migrate, % of manual vs. tool, cost savings, etc.		Migration Lead
Close out the project and provide feedback.		Migration Lead, App Owner

Related resources

References

- Apache Tomcat 7.0 documentation
- Apache Tomcat 7.0 installation guide
- Apache Tomcat JNDI documentation
- Apache TomEE documentation
- Amazon RDS for Oracle
- Amazon RDS pricing
- Oracle and AWS
- Oracle on Amazon RDS documentation
- Amazon RDS Multi-AZ deployments
- Getting started with Amazon ECS
- Getting started with Amazon RDS

Tutorials and videos

- Best Practices for Running Oracle Databases on Amazon RDS (re:Invent 2018 presentation)

Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS

Chethan Gangadharaiah and Brian motzer, Amazon Web Services

Summary

This pattern describes the steps for migrating an Oracle database on Amazon Elastic Compute Cloud (Amazon EC2) to Amazon Relational Database Service (Amazon RDS) for Oracle by using AWS Database Migration Service (AWS DMS). The pattern also uses Oracle SQL Developer or SQL *Plus to connect to your Oracle DB instance, and includes an AWS CloudFormation template that automates some of the tasks.

Migrating to Amazon RDS for Oracle enables you to focus on your business and applications while Amazon RDS takes care of database administration tasks such as provisioning databases, backup and recovery, security patches, version upgrades, and storage management.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon Machine Image (AMI) for Oracle Database on Amazon EC2

Product versions

- AWS DMS supports Oracle versions 11g (version 11.2.0.3.v1 and later), 12c, and 18c for Amazon RDS instance databases for the Enterprise, Standard, Standard One, and Standard Two editions. For the latest information about supported versions, see [Using an Oracle Database as a Target for AWS DMS] in the AWS documentation. (The attached AWS CloudFormation templates use Oracle version 12c as the source database.)
- Oracle SQL Developer 4.0.3

Architecture

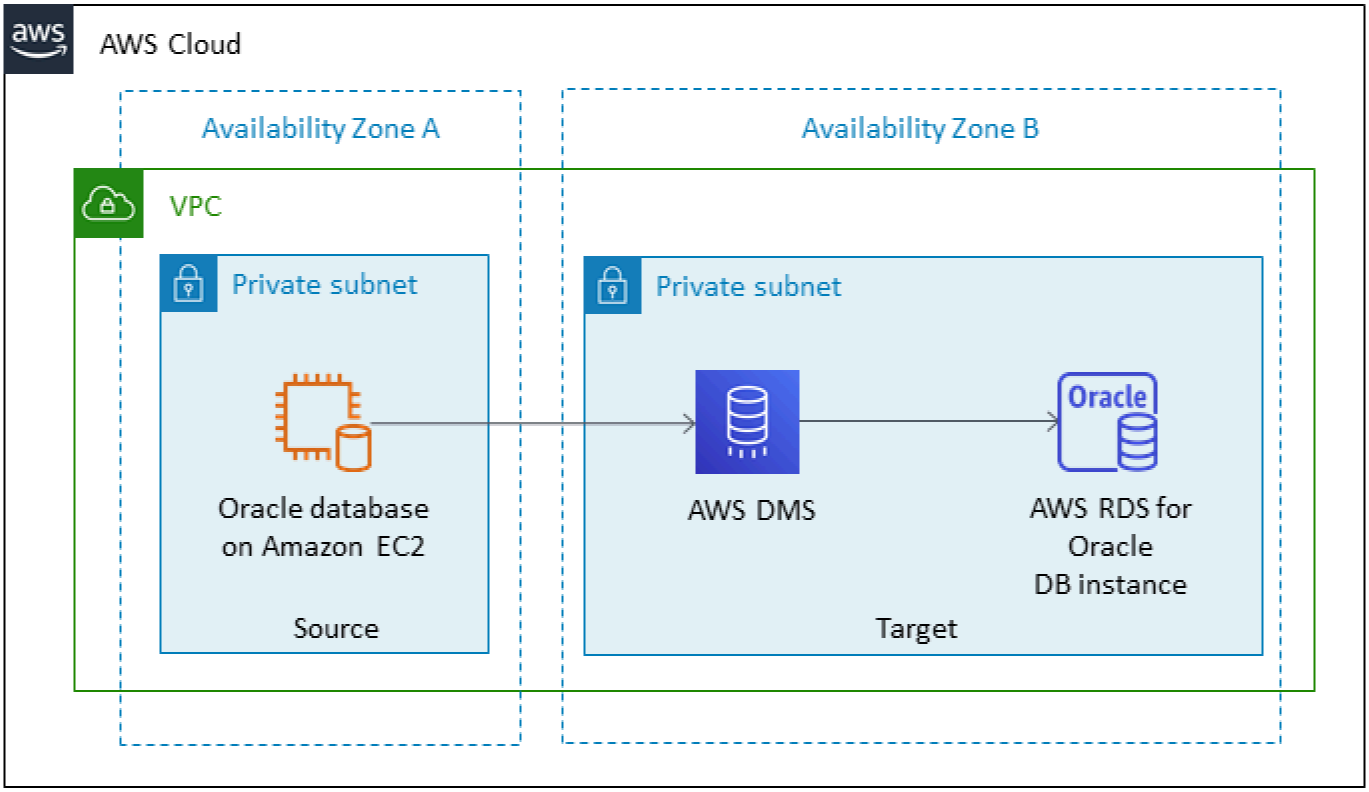
Source architecture

- Oracle Database on Amazon EC2

Target architecture

- Amazon RDS for Oracle

Migration architecture



Tools

- [AWS DMS](#) – AWS Database Migration Service (AWS DMS) helps you migrate databases to AWS quickly and securely. It supports both homogeneous and heterogeneous migrations. For information about the Oracle database versions and editions that are supported, see [Using an Oracle Database as a Source for AWS DMS](#) and [Using an Oracle Database as a Target for AWS DMS](#) in the AWS documentation.
- **Oracle SQL Developer or SQL *Plus** – These tools enable you to connect to the Amazon RDS for Oracle DB instance.

Epics

Set up your target database

Task	Description	Skills required
Create an Amazon RDS for Oracle DB instance.	Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/. Create an Oracle DB instance by selecting the appropriate engine, template, database credentials setting, instance type, storage, Multi-AZ settings, virtual private cloud (VPC) and configuration, login credentials, and additional settings for the Oracle database. For instructions, view the links in the "Related resources" section. Or use the AWS CloudFormation template (Create_RDS.yaml) in the attachment to create the Amazon RDS for Oracle DB instance.	Developer
Connect to Amazon RDS and grant privileges to the Oracle user.	Modify the security group to open the appropriate ports to connect from the local machine and the AWS DMS replication instance. When you configure connectivity, make sure that the "Publicly accessible" option is selected	Developer

Task	Description	Skills required
	so you can connect to the database from outside the VPC. Connect to Amazon RDS with Oracle SQL Developer or SQL *Plus by using the login credentials, create an AWS DMS user, and provide the required privileges to the AWS DMS user to modify the database.	

Configure the security group of the source EC2 instance

Task	Description	Skills required
Check if the Oracle database is up and running.	Use Secure Shell (SSH) to connect to the EC2 instance, and try connecting to the Oracle database by using SQL *Plus.	Developer
Modify the security group.	Modify the security group of the EC2 instance to open appropriate ports, so you can connect from your local machine and the AWS DMS replication instance.	Developer

Set up AWS DMS

Task	Description	Skills required
Create an AWS DMS replicati on instance.	In AWS DMS, create a replicati on instance in the same VPC	DBA

Task	Description	Skills required
	as your Amazon RDS for Oracle DB instance. Specify the name and description for the replication instance, choose the instance class and replication engine version (use the default), choose the VPC in which you created the Amazon RDS DB instance, set Multi-AZ settings if required, allocate storage, specify the Availability Zone, and configure additional settings. Alternatively, you can use the AWS CloudFormation template (DMS.yaml) in the attachment to implement this step.	

Task	Description	Skills required
Connect to the source and target database endpoints.	Create the source and target database endpoints by specifying the endpoint identifier, engine, server, port, login credentials, and extra connection attributes. For the source server, use the public DNS of the EC2 instance that's hosting the Oracle database. For the target server, use the endpoint of Amazon RDS for Oracle. Run a test to verify that the source and target connections are working. Alternatively, you can use the AWS CloudFormation template (DMS.yaml) in the attachment to implement this step.	DBA

Task	Description	Skills required
Create an AWS DMS task.	Create an AWS DMS task to migrate data from the source endpoint to the target endpoint, to set up replicati on between the source and destination endpoint, or both. When creating the AWS DMS task, specify the replication instance, source endpoint, target endpoint, migration type (data only, replication only, or both), table mapping, and filter. Run the AWS DMS task, monitor the task, check the table statistics, and check logs in Amazon CloudWatc h. Alternatively, you can use the AWS CloudFormation template (DMS.yaml) in the attachment to implement this step.	DBA

Related resources

- Creating an Amazon RDS DB instance
- Connecting to a DB Instance Running the Oracle Database Engine
- AWS DMS documentation
- AWS DMS Step-by-Step Walkthroughs
- Migrating Oracle Databases to the AWS Cloud

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash

Aditya Goteti, Amazon Web Services

Summary

This pattern describes how to move data from an on-premises Oracle database to Amazon OpenSearch Service using Logstash. It includes architectural considerations and some required skill sets and recommendations. The data can be from a single table or from multiple tables in which a full-text search will need to be performed.

OpenSearch Service can be configured within a virtual private cloud (VPC), or it can be placed publicly with IP-based restrictions. This pattern describes a scenario where OpenSearch Service is configured within a VPC. Logstash is used to collect the data from the Oracle database, parse it to JSON format, and then feed the data into OpenSearch Service.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Java 8 (required by Logstash 6.4.3)
- Connectivity between the on-premises database servers and Amazon Elastic Compute Cloud (Amazon EC2) instances in a VPC, established using AWS Virtual Private Network (AWS VPN)
- A query to retrieve the required data to be pushed to OpenSearch Service from the database
- Oracle Java Database Connectivity (JDBC) drivers

Limitations

- Logstash cannot identify records that are hard-deleted from the database

Product versions

- Oracle Database 12c
- OpenSearch Service 6.3
- Logstash 6.4.3

Architecture

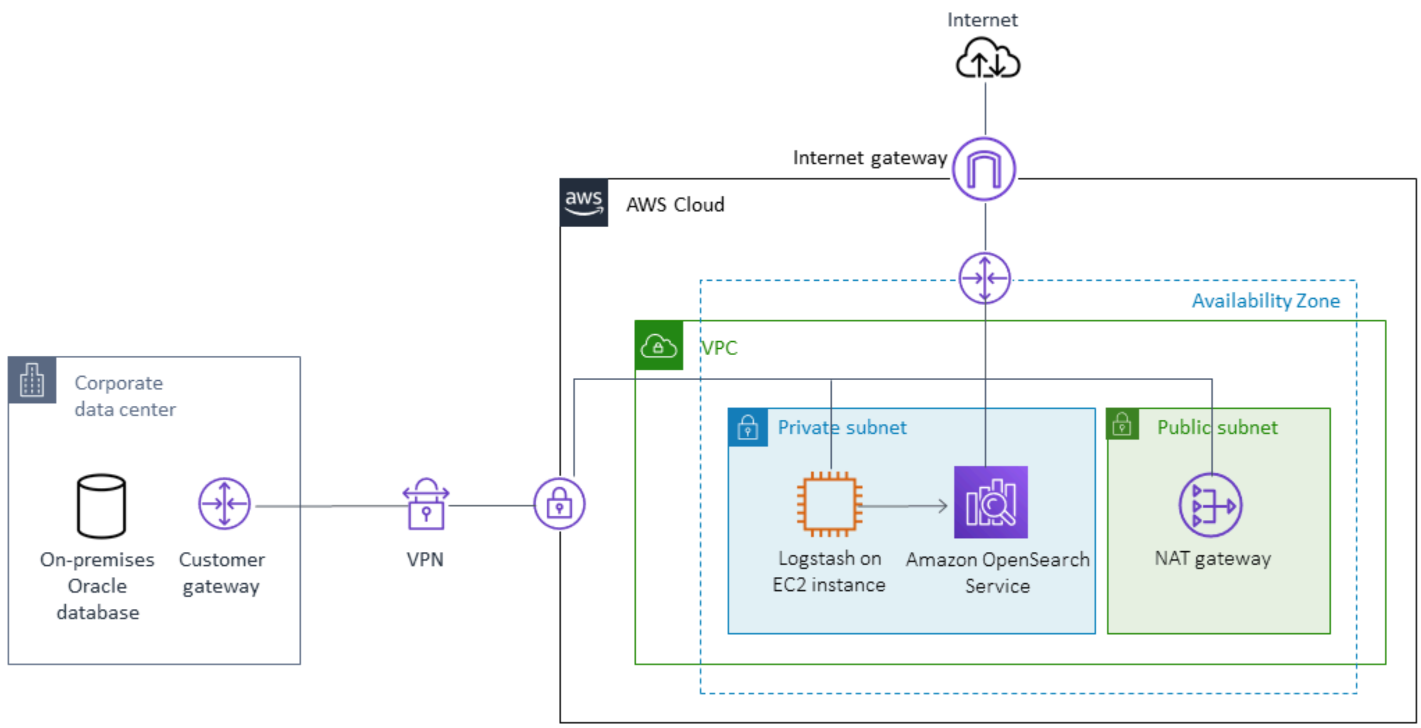
Source technology stack

- On-premises Oracle database
- On-premises AWS VPN

Target technology stack

- VPC
- EC2 instance
- OpenSearch Service
- Logstash
- NAT Gateway (for operating system updates on EC2 instances and to install Java 8, Logstash, and plugins)

Data migration architecture



Tools

- Logstash 6.4.3

- JDBC input plugin (download and more information)
- Logstash output plugin (logstash-output-amazon_es)
- Oracle JDBC drivers

Epics

Plan the migration

Task	Description	Skills required
Identify the size of the source data.	The size of the source data is one of the parameters that you use to determine the number of shards to be configured in an index.	DBA, Database developer
Analyze the data types of each column and correspon ding data.	OpenSearch Service dynamically maps the data type when a previously unseen field is found in the document. If there are any specific data types or formats (for example, date fields) that need to be explicitly declared, identify the fields and define the mapping for those fields during index creation.	App owner, Developer, Database developer
Determine if there are any columns with primary or unique keys.	To avoid duplication of records in Amazon OpenSearc h Service during updates or inserts, you need to configure the `document_id` setting in the output section of the `amazon_es` plugin (for example, `document_id => "%{customer_id}"` where	App owner, Developer

Task	Description	Skills required
	`customer_id` is a primary key).	
Analyze the number and frequency of new records added; check how frequently the records are deleted.	This task is required to understand the growth rate of source data. If data is intensively read and insertion s are rare, you can have a single index. If new records are inserted frequently and there are no deletions, the shard size can easily exceed the recommended maximum size of 50 GB. In this case, you can dynamically create an index by configuring index patterns in Logstash and in the code where you can access it by using an alias.	App owner, Developer
Determine how many replicas are required.		App owner, Developer
Determine the number of shards to be configured on the index.		App owner, Developer
Identify the instance types for dedicated master nodes, data nodes, and the EC2 instance.	For more information, see the Related resources section.	App owner, Developer
Determine the number of dedicated master nodes and data nodes required.	For more information, see the Related resources section.	

Migrate data

Task	Description	Skills required
Launch an EC2 instance.	Launch an EC2 instance within the VPC to which AWS VPN is connected.	Amazon VPC constructs, AWS VPN
Install Logstash on the EC2 instance.		Developer
Install the Logstash plugins.	Install the required Logstash plugins `jdbc-input` and `logstash-output-amazon_es` .	Developer
Configure Logstash.	Create the Logstash keystore to store sensitive information such as AWS Secrets Manager keys and database credentials, and then place the references in a Logstash configuration file.	Developer
Configure the dead letter queue and persistent queue.	By default, when Logstash encounters an event that it cannot process because the data contains a mapping error or some other issue, the Logstash pipeline either hangs or drops the unsuccessful event. To protect against data loss in this situation, you can configure Logstash to write unsuccessful events to a dead letter queue instead of dropping them. To protect against data loss	Developer

Task	Description	Skills required
	during abnormal terminati on, Logstash has a persistent queue feature that will store the message queue on disk. Persistent queues provide the data durability in Logstash.	
Create the Amazon OpenSearch Service domain.	Create the Amazon OpenSearch Service domain with an access policy that doesn't require signing requests with AWS Identity and Access Managemen t (IAM) credentials. The Amazon OpenSearch Service domain must be created within the same VPC. You should also select the instance types and set the number of dedicated and master nodes based on your analysis.	Developer
Configure the required Amazon OpenSearch Service logs.	For more information, see the [OpenSearch Service documentation]().	
Create the index.		Developer

Task	Description	Skills required
Start Logstash.	Run Logstash as a backgroun d service. Logstash runs the configured SQL query, pulls the data, converts it to JSON format, and feeds it to OpenSearch Service. For the initial load, do not configure the scheduler in the Logstash configuration file.	Developer

Task	Description	Skills required
Check documents.	Check the number of documents on the index and whether all documents are present in the source database. During initial load, they are added to the index and used to stop Logstash. Change the Logstash configuration to add a scheduler that runs at a fixed interval depending on client requirements, and restart Logstash. Logstash will pick only the records that were updated or added after the last run, and the last run timestamp is stored in the file that is configured with the property `last_run_metadata_path => "/usr/share/logstash/.logstash_jdbc_last_run"` in the Logstash configuration file.	Developer

Related resources

- [Recommended CloudWatch Alarms]
- [Dedicated Amazon OpenSearch Service Master Nodes]
- [Sizing Amazon OpenSearch Service Domains]
- [Logstash documentation]
- [JDBC input plugin]

- Logstash output plugin
- Amazon OpenSearch Service website

Migrate an on-premises Oracle database to Amazon RDS for Oracle

Baji Shaik and Pavan Pusuluri, Amazon Web Services

Summary

This pattern describes the steps for migrating on-premises Oracle databases to Amazon Relational Database Service (Amazon RDS) for Oracle. As part of the migration process, you create a migration plan and consider important factors about your target database infrastructure based on your source database. You can choose one of two migration options based on your business requirements and use case:

- AWS Database Migration Service (AWS DMS) – You can use AWS DMS to migrate databases to the AWS Cloud quickly and securely. Your source database remains fully operational during the migration, which minimizes downtime to applications that rely on the database. You can reduce migration time by using AWS DMS to create a task that captures ongoing changes after you complete an initial full-load migration through a process called change data capture (CDC).
- Native Oracle tools – You can migrate databases by using native Oracle tools, such as Oracle and Data Pump Export and Data Pump Import with Oracle GoldenGate for CDC. You can also use native Oracle tools such as the original Export utility and original Import utility to reduce the full-load time.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An on-premises Oracle database
- An Amazon RDS Oracle database (DB) instance

Limitations

- Database size limit: 64 TB

Product versions

- Oracle versions 11g (versions 11.2.0.3.v1 and later) and up to 12.2 and 18c. For the latest list of supported versions and editions, see Amazon RDS for Oracle in the AWS documentation.

For Oracle versions supported by AWS DMS, see [Using an Oracle database as a source for AWS DMS] in the AWS DMS documentation.

Architecture

Source technology stack

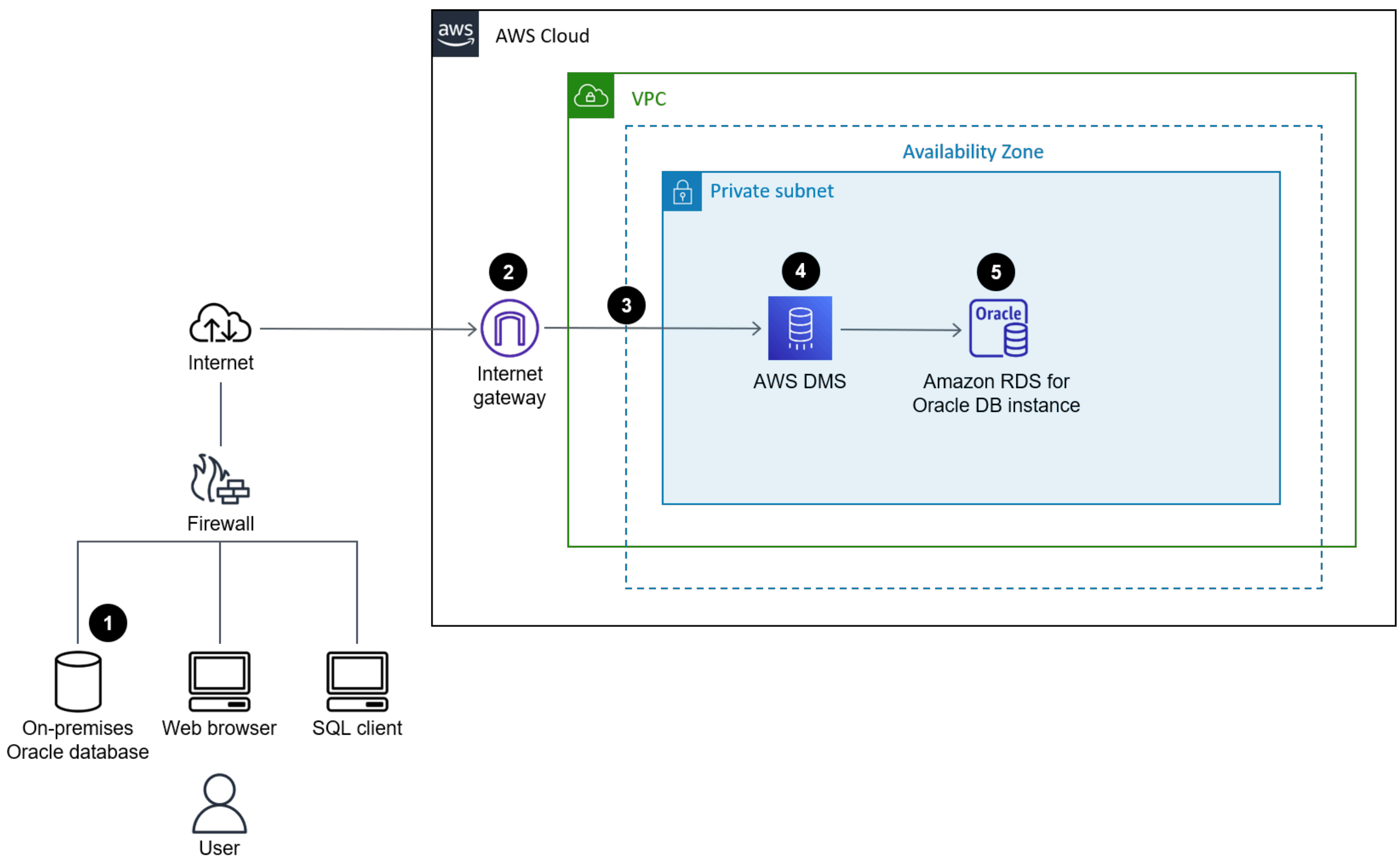
- On-premises Oracle databases

Target technology stack

- Amazon RDS for Oracle

Source and target architecture

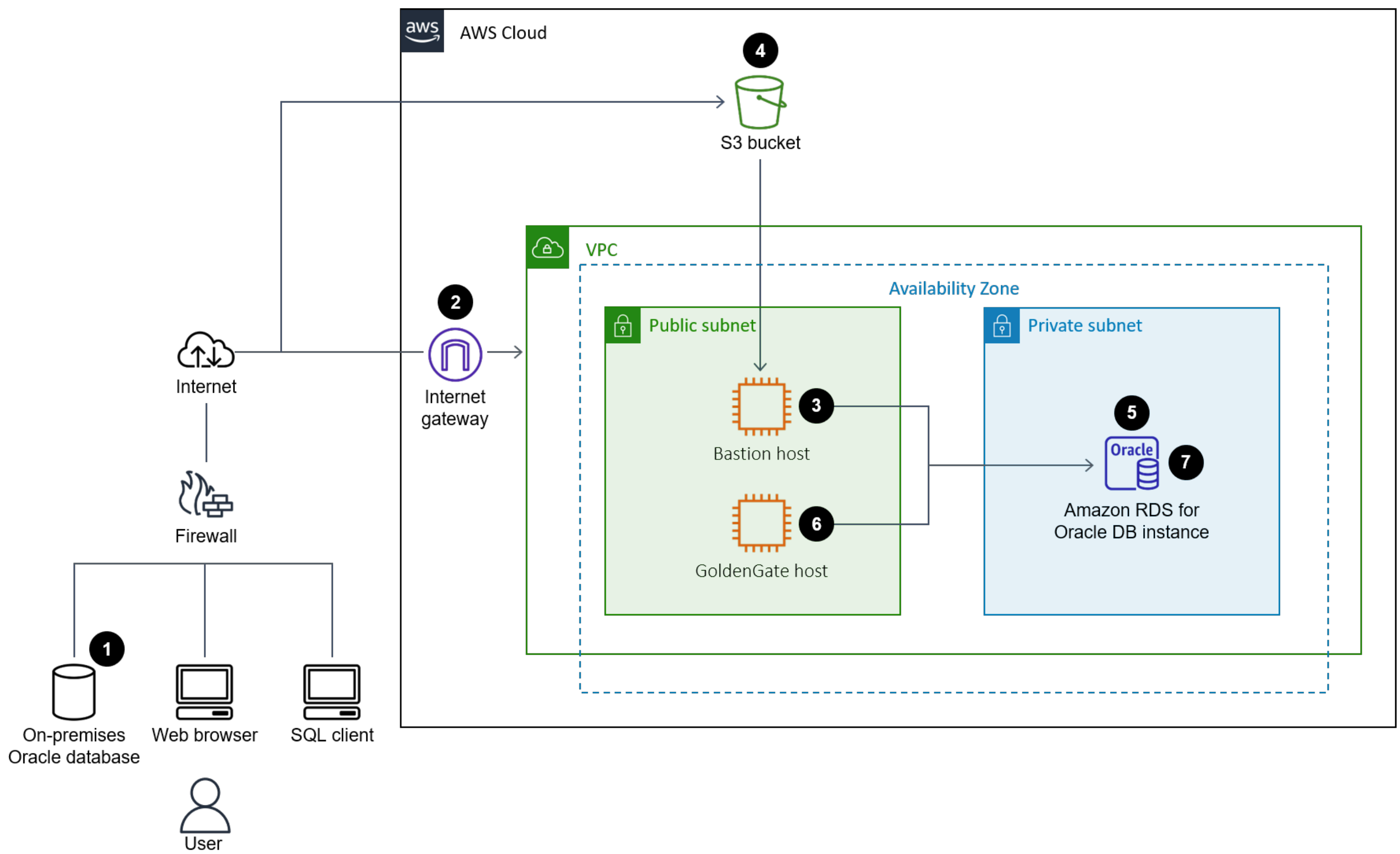
The following diagram shows how to migrate an on-premises Oracle database to Amazon RDS for Oracle by using AWS DMS.



The diagram shows the following workflow:

1. Create or use an existing database user, grant the required AWS DMS permissions to that user, turn on ARCHIVELOG mode, and then set up supplemental logging.
2. Configure the internet gateway between the on-premises and AWS network.
3. Configure source and target endpoints for AWS DMS.
4. Configure AWS DMS replication tasks to migrate the data from the source database to the target database.
5. Complete the post-migration activities on the target database.

The following diagram shows how to migrate an on-premises Oracle database to Amazon RDS for Oracle by using native Oracle tools.



The diagram shows the following workflow:

1. Create or use an existing database user and grant the required permissions to back up the Oracle database by using Oracle Export (`exp`) and Import (`imp`) utilities.
2. Configure the internet gateway between the on-premises and AWS network.
3. Configure the Oracle client on the Bastion host to take the backup database.

4. Upload the backup database to an Amazon Simple Storage Service (Amazon S3) bucket.
5. Restore the database backup from Amazon S3 to an Amazon RDS for Oracle database.
6. Configure Oracle GoldenGate for CDC.
7. Complete the post-migration activities on the target database.

Tools

- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises setups.
- Native Oracle tools help you perform a homogeneous migration. You can use Oracle Data Pump to migrate data between your source and target databases. This pattern uses Oracle Data Pump to perform the full load from the source database to the target database.
- Oracle GoldenGate helps you perform logical replication between two or more databases. This pattern uses GoldenGate to replicate the delta changes after the initial load by using Oracle Data Pump.

Epics

Plan the migration

Task	Description	Skills required
Create project documents and record database details.	1. Document your migration goals, migration requireme nts, key project stakehold ers, project milestones, project deadlines, key metrics, migration risks, and risk mitigation plans. 2. Document critical informati on about your source database, including RAM, IOPS, and CPUs. You will later use this information to determine the appropria te target DB instance.	DBA

Task	Description	Skills required
	3. Validate the versions of your source and target databases.	
Identify storage requireme nts.	Identify and document your storage requirements, including the following: 1. Calculate the storage allocated for the source DB instance. 2. Gather the historical growth metrics from the source DB instance. 3. Forecast future growth for the target DB instance. **Note** For [General Purpose (gp2) SSD volumes](), you get three IOPS per 1 GB of storage. Allocate storage by calculati ng the total number of read and write IOPS on the source database.	DBA, SysAdmin

Task	Description	Skills required
Choose the proper instance type based on compute requirements.	1. Determine the compute requirements of the target DB instance. 2. Identify performance issues. 3. Consider the factors for determining the appropria te instance type: • CPU utilization of the source DB instance • IOPS (read and write) for the source DB instance • Memory footprint on the source DB instance	SysAdmin
Identify network access security requirements.	1. Identify and document the network access security requirements for your source and target databases. 2. Configure the appropria te security groups for enabling the application to communicate with the database.	DBA, SysAdmin

Task	Description	Skills required
Identify the application migration strategy.	1. Determine and document the migration cutover strategy. 2. Determine and document your application’s recovery time objective (RTO) and recovery point objective (RPO), and then plan for the cutover accordingly.	DBA, SysAdmin, App owner
Identify migration risks.	Assess the database and document migration specific risks and mitigations. For example: • Identify no-logging tables and highlight the risk of data loss in the event of recovery. • Extract the source database users and privileges, and highlight the conflicts with Amazon RDS privileges. • Review the alert log for any Oracle-specific errors and warnings. • Identify the supported and unsupported features of the target DB instance. • Review the deprecated features of the target DB version engine.	DBA

Configure the infrastructure

Task	Description	Skills required
Create a VPC.	Create a new Amazon Virtual Private Cloud (Amazon VPC) for the target DB instance.	SysAdmin
Create security groups.	Create a security group in your new VPC to allow inbound connections to the DB instance.	SysAdmin
Create an Amazon RDS for Oracle DB instance.	Create the target DB instance with the new VPC and security group, and then start the instance.	SysAdmin

Option 1 - Use native Oracle or third-party tools to migrate data

Task	Description	Skills required
Prepare the source database.	1. Create a Data Pump directory or use an existing one. 2. Create a migration user and grant permissions to perform the Data Pump extract. 3. Extract roles, users, and tablespaces from the source database as a SQL script. 4. Transfer the extracted Data Pump dump to the target	DBA, SysAdmin

Task	Description	Skills required
	DB instance `data pump` directory.	

Task	Description	Skills required
Prepare the target database.	1. Confirm that all the database options (for example, text and Java) are installed or enabled on the target Amazon RDS for Oracle DB instance. 2. Create a Data Pump directory or use an existing one. 3. Create a migration user and grant permissions to perform the Data Pump import. 4. Create the required tablespaces, users, and roles on the target DB instance. 5. Import the transferred Data Pump export dump to the target database. 6. Create any indexes excluded during import or object creation. 7. Create any constraints excluded during import. 8. Validate or recompile invalid objects. 9. Rebuild the invalid indexes. 10. Validate the database object counts between the source and the target databases.	DBA, SysAdmin

Task	Description	Skills required
	11. Resolve any discrepancies found between object counts.	

Option 2 - Use AWS DMS to migrate data

Task	Description	Skills required
Prepare the data.	1. Clean the data in the source database. 2. Create a replication instance. 3. Create a source endpoint and target endpoint. 4. Identify the number of tables and objects to be migrated.	DBA
Migrate the data.	1. Drop foreign key constraints and triggers on the target database. 2. Drop secondary indexes on the target database. 3. Configure AWS DMS full-load task settings from the source database to the target database. 4. Enable foreign keys. 5. Enable AWS DMS CDC to replicate ongoing changes. 6. Enable triggers. 7. Update the sequences.	DBA

Task	Description	Skills required
	8. Validate the source and target data.	

Cut over to the target database

Task	Description	Skills required
Switch the application clients to the new infrastructure.	1. Stop all application services and client connections pointing to Oracle. 2. Run the AWS DMS tasks. 3. Set up a rollback task (for example, reverse CDC from the Amazon RDS database to the on-premises Oracle database). 4. Validate the data. 5. Start the application services on the new target database by configuri ng Amazon Route 53 to the new Amazon RDS for Oracle DB instance. 6. Add Amazon CloudWatc h monitoring to your new Amazon RDS for Oracle DB instance.	DBA, SysAdmin, App owner
Implement your rollback plan.	1. Stop all application services pointing to the Amazon RDS for Oracle DB instance.	DBA, App owner

Task	Description	Skills required
	2. Roll back the changes to the source on-premises Oracle database by using an AWS DMS task. 3. Stop the AWS DMS tasks running from the on-premises Oracle database to the Amazon RDS for Oracle database. 4. Configure the applications back on the source Oracle database. 5. Confirm the rollback deployment is complete.	

Close out the migration project

Task	Description	Skills required
Clean up resources.	Shut down or remove the temporary AWS resources, such as the AWS DMS replicati on instance and S3 bucket.	DBA, SysAdmin
Review project documents.	Review your migration planning documents and goals, and then confirm that you completed all required migration steps.	DBA, SysAdmin, App owner
Gather metrics.	Record key migration metrics, including how long it took to complete the migration, the percentage of manual vs.	DBA, SysAdmin, App owner

Task	Description	Skills required
	tool-based tasks, cost savings, and other relevant metrics.	
Close out the project.	Close out the migration project and capture feedback about the effort.	DBA, SysAdmin, App owner

Related resources

References

- Migrating Oracle databases to the AWS Cloud (AWS Prescriptive Guidance)
- AWS Database Migration Service (AWS DMS documentation)
- Amazon RDS Pricing (Amazon RDS documentation)

Tutorials and videos

- Getting Started with AWS Database Migration Service (AWS DMS documentation)
- Amazon RDS resources (Amazon RDS documentation)
- AWS Database Migration Service (DMS) (YouTube)

Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump

Mohan Annam and Brian motzer, Amazon Web Services

Summary

This pattern describes how to migrate an Oracle database from an on-premises data center to an Amazon Relational Database Service (Amazon RDS) for Oracle DB instance by using Oracle Data Pump.

The pattern involves creating a data dump file from the source database, storing the file in an Amazon Simple Storage Service (Amazon S3) bucket, and then restoring the data to an Amazon RDS for Oracle DB instance. This pattern is useful when you encounter limitations using AWS Database Migration Service (AWS DMS) for the migration.

Prerequisites and limitations

Prerequisites

- An active AWS account
- The required permissions to create roles in AWS Identity and Access Management (IAM) and for an Amazon S3 multipart upload
- The required permissions to export data from the source database
- AWS Command Line Interface (AWS CLI) installed and configured

Product versions

- Oracle Data Pump is available only for Oracle Database 10g Release 1 (10.1) and later versions.

Architecture

Source technology stack

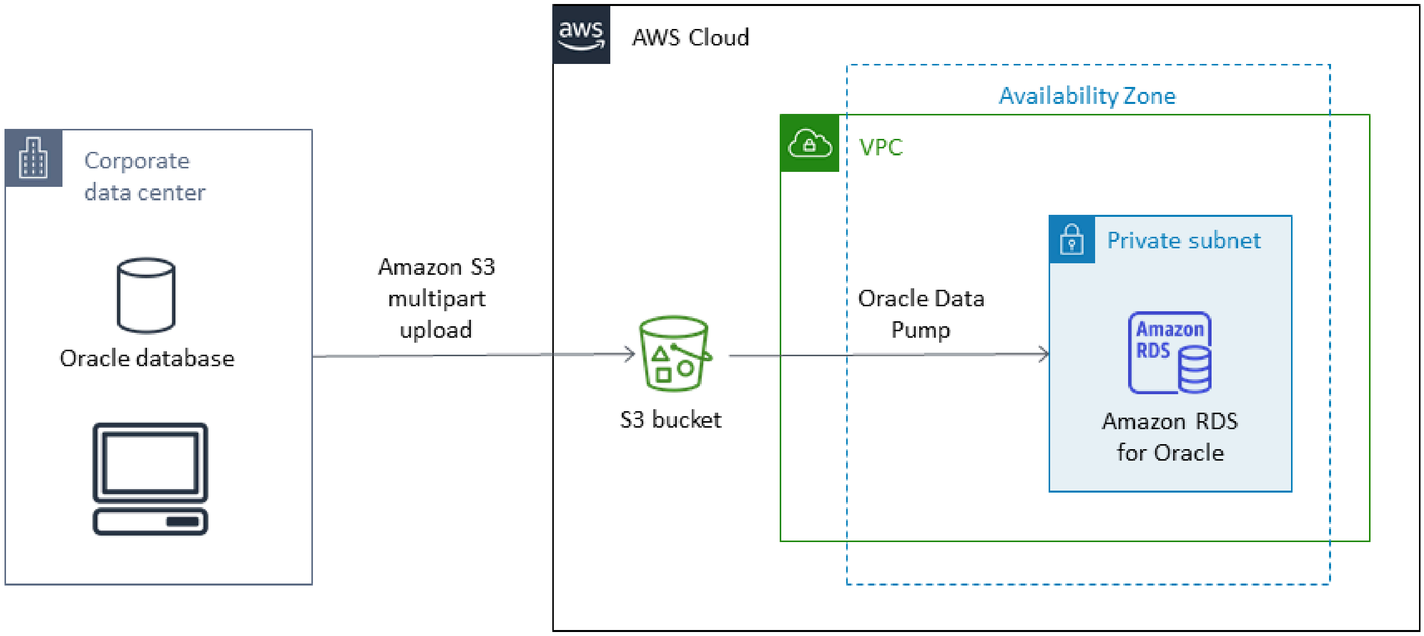
- On-premises Oracle databases

Target technology stack

- Amazon RDS for Oracle

- SQL client (Oracle SQL Developer)
- An S3 bucket

Source and target architecture



Tools

AWS services

- [AWS Identity and Access Management (IAM)](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them. In this pattern, IAM is used to create the roles and policies necessary for migrating data from Amazon S3 to Amazon RDS for Oracle.
- Amazon Relational Database Service (Amazon RDS) for Oracle helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Oracle Data Pump helps you move data and metadata from one database to another at high speeds. In this pattern, Oracle Data Pump is used to export the data dump (.dmp) file to the

Oracle server, and to import it into Amazon RDS for Oracle. For more information, see Importing data into Oracle on Amazon RDS in the Amazon RDS documentation.

- Oracle SQL Developer is an integrated development environment that simplifies the development and management of Oracle databases in both traditional and cloud-based deployments. It interacts with both the on-premises Oracle database and Amazon RDS for Oracle to run the SQL commands required for exporting and importing data.

Epics

Create an S3 bucket

Task	Description	Skills required
Create the bucket.	To create the S3 bucket, follow the instructions in the AWS documentation.	AWS systems administrator

Create the IAM role and assign policies

Task	Description	Skills required
Configure IAM permissions.	To configure permissions, follow the instructions in the AWS documentation.	AWS systems administrator

Create the target Amazon RDS for Oracle DB instance and associate the Amazon S3 integration role

Task	Description	Skills required
Create the target Amazon RDS for Oracle DB instance.	To create the Amazon RDS for Oracle instance, follow the instructions in the AWS documentation.	AWS systems administrator

Task	Description	Skills required
Associate the role with the DB instance.	To associate the role with the instance, follow the instructions in the [AWS documentation]().	DBA

Create the database user on the target database

Task	Description	Skills required

Create the user.	Connect to the target Amazon RDS for Oracle database from Oracle SQL Developer or SQL*Plus, and run the following SQL command to create the user to import the schema into.
	`create user SAMPLE_SCHEMA identified by <PASSWORD>;`
	`grant create session, resource to <USER NAME>;`
	`alter user <USER NAME> quota 100M on users;`
		DBA

Create the export file from the source Oracle database

Task	Description	Skills required
Create a data dump file.	To create a dump file named `sample.dmp` in the `DATA_PUMP_DIR` directory for exporting the SAMPLE_SC	DBA

Task	Description	Skills required

	HEMA user, use the following script.
	
	``
	DECLARE
	    hdnl NUMBER;
	BEGIN
	    hdnl := dbms_data
	pump.open(operation
	 => 'EXPORT',
	
	           job_mode =>
	 'SCHEMA',
	
	           job_name =>
	 NULL);
	
	    dbms_datapump.add_
	file(handle => hdnl,
	
	         filename =>
	 'sample.dmp',
	
	        directory =>
	 'DATA_PUMP_DIR',
	
	         filetype =>
	 dbms_datapump.ku$_
	file_type_dump_file);
	
	    dbms_datapump.add_
	file(handle => hdnl,
	
	         filename =>
	 'export.log',
	
	        directory =>
	 'DATA_PUMP_DIR',
	
	         filetype =>
	 dbms_datapump.ku$_
	file_type_log_file);
	``	

Task	Description	Skills required

	``
	    dbms_datapump.meta
	data_filter(hdnl,
	 'SCHEMA_EXPR', 'IN
	 (''SAMPLE_SCHEMA''
	)');
	
	    dbms_datapump.star
	t_job(hdnl);
	END;
	/
	``
	Review the export details by reviewing the `export.log` file in your local `DATA_PUMP_DIR` directory.	

Upload the dump file to the S3 bucket

Task	Description	Skills required
Upload the data dump file from the source to the S3 bucket.	Using AWS CLI, run the following command. `aws s3 cp sample.dmp s3://<bucket_created_epic_1>/`	DBA

Download the export file from the S3 bucket to the RDS instance

Task	Description	Skills required
Download the data dump file to Amazon RDS	To copy the dump file `sample.dmp` from the S3 bucket to the Amazon RDS	AWS systems administrator

Task	Description	Skills required
	for Oracle database, run the following SQL command. In this example, the `sample.dmp` file is downloaded from the S3 bucket `my-s3-integration1` to the Oracle directory `DATA_PUMP_DIR`. Make sure that you have sufficient disk space allocated to your RDS instance to accommodate both the database and the export file. ```-- If you want to download all the files in the S3 bucket remove the p_s3_prefix line. SELECT rdsadmin.rdsadmin_s3_tasks.download_from_s3(p_bucket_name => 'my-s3-integration', p_s3_prefix => 'sample.dmp', p_directory_name => 'DATA_PUMP_DIR') AS TASK_ID FROM DUAL;``` The previous command outputs a task ID. To review the status of the download by reviewing the data in the task ID, run the following command.	

Task	Description	Skills required

	`SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('BDUMP','dbtask-<task_id>.log'));`
	
	To see the files in the `DATA_PUMP_DIR` directory, run the following command.
	
	`SELECT filename, type,filesize/1024/1024 size_megs,to_char(mtime,'DD-MON-YY HH24:MI:SS') timestamp FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => upper('DATA_PUMP_DIR'))) order by 4;`	

Import the dump file into the target database

Task	Description	Skills required

Restore the schema and data to Amazon RDS.	To import the dump file into the `sample_schema` database schema, run the following SQL command from SQL Developer or SQL*Plus.
	
	`DECLARE`
	`hdnl NUMBER;`
	`BEGIN`
		DBA

Task	Description	Skills required

	``
	hdnl := DBMS_DATA
	PUMP.OPEN(operation
	 => 'IMPORT', job_mode
	 => 'SCHEMA', job_name=
	>null);
	
	DBMS_DATAPUMP.ADD_
	FILE(handle => hdnl,
	 filename => 'sample.d
	mp', directory =>
	 'DATA_PUMP_DIR',
	 filetype => dbms_data
	pump.ku$_file_type
	_dump_file);
	
	DBMS_DATAPUMP.ADD_FILE
	(handle => hdnl,
	 filename => 'import.l
	og', directory =>
	 'DATA_PUMP_DIR',
	 filetype => dbms_data
	pump.ku$_file_type
	_log_file);
	
	DBMS_DATAPUMP.
	METADATA_FILTER(hd
	nl,'SCHEMA_EXPR','
	IN (''SAMPLE_SCHEMA''
	)');
	
	DBMS_DATAPUMP.START_J
	OB(hdnl);
	
	END;
	/
	``
	To see the log file from the import, run the following command.	

Task	Description	Skills required
	```SELECT text FROM table(rdsadmin.rds_file_util.read_text_file('DATA_PUMP_DIR','import.log'));```	

Remove the dump file from the DATA_PUMP_DIR directory

Task	Description	Skills required

List and clean up the export files.	List and remove the export files in the `DATA_PUMP_DIR` directory, run the following commands.
	```-- List the files SELECT filename, type,filesize/1024/1024 size_megs,to_char(mtime,'DD-MON-YY HH24:MI:SS') timestamp FROM TABLE(rdsadmin.rds_file_util.listdir(p_directory => upper('DATA_PUMP_DIR'))) order by 4;```
	```-- Remove the files EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','sample.dmp'); EXEC UTL_FILE.FREMOVE('DATA_PUMP_DIR','import.log');```
		AWS systems administrator

Related resources

- Amazon S3 integration
- Create a DB instance
- Importing data into Oracle on Amazon RDS
- Amazon S3 documentation
- IAM documentation
- Amazon RDS documentation
- Oracle Data Pump documentation
- Oracle SQL Developer

Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical

Rajesh Madiwale, Amazon Web Services

Summary

This pattern outlines steps for migrating a PostgreSQL database (version 9.5 and later) from Amazon Elastic Compute Cloud (Amazon EC2) to Amazon Relational Database Service (Amazon RDS) for PostgreSQL by using the PostgreSQL **pglogical** extension. Amazon RDS now supports the pglogical extension for PostgreSQL version 10.

Prerequisites and limitations

Prerequisites

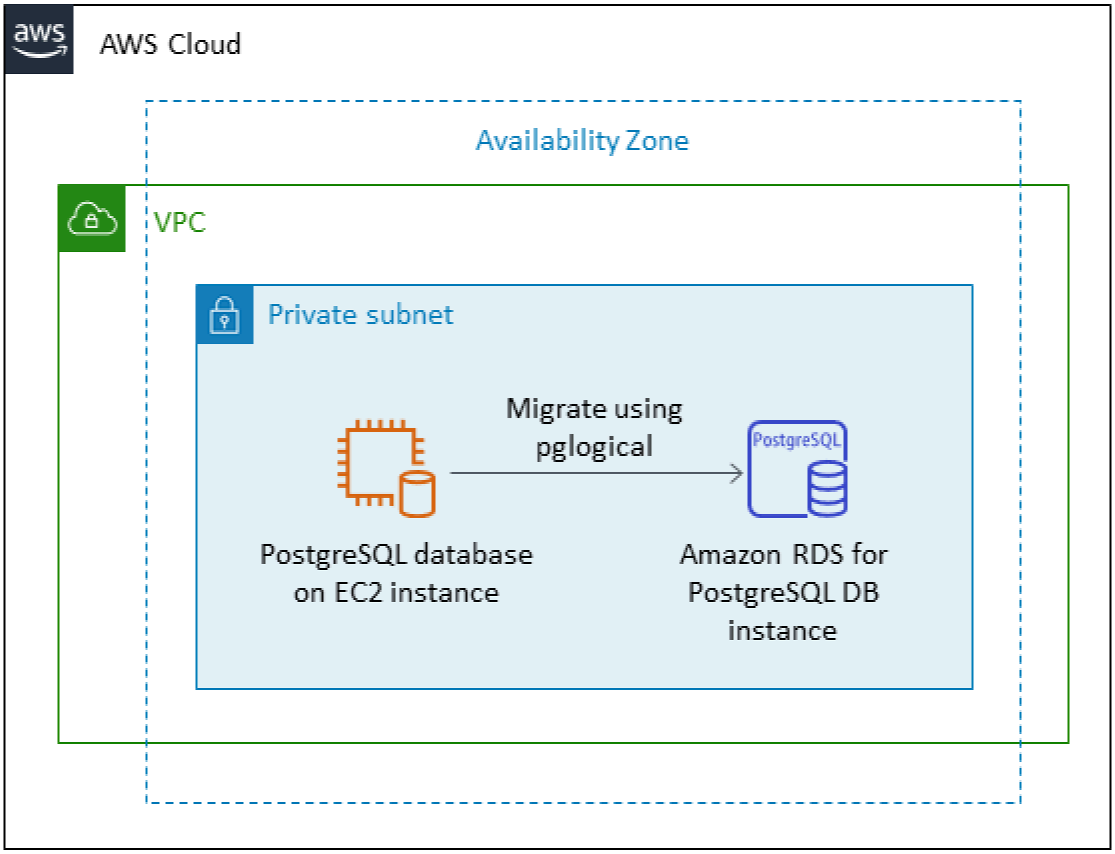
- Choose the right type of Amazon RDS instance. For more information, see Amazon RDS Instance Types.
- Make sure that the source and target versions of PostgreSQL are the same.
- Install and integrate the **pglogical** extension with PostgreSQL on Amazon EC2.

Product versions

- PostgreSQL version 10 and later on Amazon RDS, with the features supported on Amazon RDS (see PostgreSQL on Amazon RDS in the AWS documentation). This pattern was tested by migrating PostgreSQL 9.5 to PostgreSQL version 10 on Amazon RDS, but it also applies to later versions of PostgreSQL on Amazon RDS.

Architecture

Data migration architecture



Tools

- **pglogical** extension
- PostgreSQL native utilities: **pg_dump** and **pg_restore**

Epics

Migrate data by using the pglogical extension

Task	Description	Skills required
Create an Amazon RDS PostgreSQL DB instance.	Set up a PostgreSQL DB instance in Amazon RDS. For instructions, see the Amazon RDS for PostgreSQL documentation.	DBA

Task	Description	Skills required
Obtain a schema dump from the source PostgreSQL database and restore it into the target PostgreSQL database.	1. Use the pg_dump utility with the `-s` option to generate a schema file from the source database. 2. Use the psql utility with the `-f` option to load the schema into the target database.	DBA
Turn on logical decoding.	In the Amazon RDS DB parameter group, set the `rds.logical_replication` static parameter to 1. For instructions, see the Amazon RDS documentation.	DBA

Task	Description	Skills required

Create the pglogical extension on the source and target databases.	1. Create the `pglogical` extension on the source PostgreSQL database:
	`psql -h <amazon-ec2-endpoint> -d target-dbname -U target-dbuser -c "create extension pglogical ;"`
	2. Create the `pglogical` extension on the target PostgreSQL database:
	`psql -h <amazon-rds-endpoint> -d source-dbname -U source-dbuser -c "create extension pglogical ;"`
		DBA

Create a publisher on the source PostgreSQL database.	To create a publisher, run: `psql -d dbname -p 5432 <<EOF SELECT pglogical.create_node( node_name := 'provider1', dsn := 'host=<ec2-endpoint> port=5432 dbname=source-dbname user=source-dbuser' ); EOF`	DBA

Task	Description	Skills required
Create a replication set, add tables and sequences.	To create a replication set on the source PostgreSQL database, and to add tables and sequences to the replicati on set, run: `psql -d dbname -p 5432 <<EOF SELECT pglogical .replication_set_a dd_all_tables('def ault', '{public} '::text[],synchron ize_data := true); EOF`	DBA
Create a subscriber.	To create a subscriber on the target PostgreSQL database, run: `psql -h <rds-endpoint> -d target-dbname -U target-dbuser <<EOF SELECT pglogical .create_node( node_name := 'subscriber1', dsn := 'host=<rd s-endpoint> port=5432 dbname=target-dbna me password=postgres user=target-dbuser' ); EOF`	DBA

Task	Description	Skills required
Create a subscription.	To create a subscription on the target PostgreSQL database, run: ```psql -h <rds-endpoint> -d target -U postgres <<EOF SELECT pglogical .create_subscription(subscription_name := 'subscription1', replication_sets := array['default'], provider_dsn := 'host=<ec2-endpoint> port=5432 dbname=<source-dbname> password=<password> user=source-dbuser');```	DBA

Validate your data

Task	Description	Skills required
Check source and target databases.	Check the source and target databases to confirm that data is being replicated successfully. You can perform basic validation by using `select count(1)` from the source and target tables.	DBA

Related resources

- [Amazon RDS]

- Logical replication for PostgreSQL on Amazon RDS (Amazon RDS documentation)
- pglogical (GitHub repository)
- Limitations of pglogical (GitHub repository README file)
- Migrating PostgreSQL from on-premises or Amazon EC2 to Amazon RDS using logical replication (AWS Database blog)

Migrate an on-premises PostgreSQL database to Aurora PostgreSQL

Baji Shaik and Jitender Kumar, Amazon Web Services

Summary

Amazon Aurora PostgreSQL-Compatible Edition combines the performance and availability of high-end commercial databases with the simplicity and cost-effectiveness of open-source databases. Aurora provides these benefits by scaling storage across three Availability Zones in the same AWS Region, and supports up to 15 read replica instances for scaling out read workloads and providing high availability within a single Region. By using an Aurora global database, you can replicate PostgreSQL databases in up to five Regions for remote read access and disaster recovery in the event of a Region failure. This pattern describes the steps for migrating an on-premises PostgreSQL source database to an Aurora PostgreSQL-Compatible database. The pattern includes two migration options: using AWS Data Migration Service (AWS DMS) or using native PostgreSQL tools (such as pg_dump, pg_restore, and psql) or third-party tools.

The steps described in this pattern also apply to target PostgreSQL databases on Amazon Relational Database Service (Amazon RDS) and Amazon Elastic Compute Cloud (Amazon EC2) instances.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A PostgreSQL source database in an on-premises data center
- An Aurora PostgreSQL-Compatible DB instance or an Amazon RDS for PostgreSQL DB instance

Limitations

- Database size limits are 64 TB for Amazon RDS for PostgreSQL and 128 TB for Aurora PostgreSQL-Compatible.
- If you're using the AWS DMS migration option, review AWS DMS limitations on using a PostgreSQL database as a source.

Product versions

- For PostgreSQL major and minor version support in Amazon RDS, see [Amazon RDS for PostgreSQL updates] in the Amazon RDS documentation.
- For PostgreSQL support in Aurora, see [Amazon Aurora PostgreSQL updates] in the Aurora documentation.
- If you're using the AWS DMS migration option, see [supported PostgreSQL versions] in the AWS DMS documentation.

Architecture

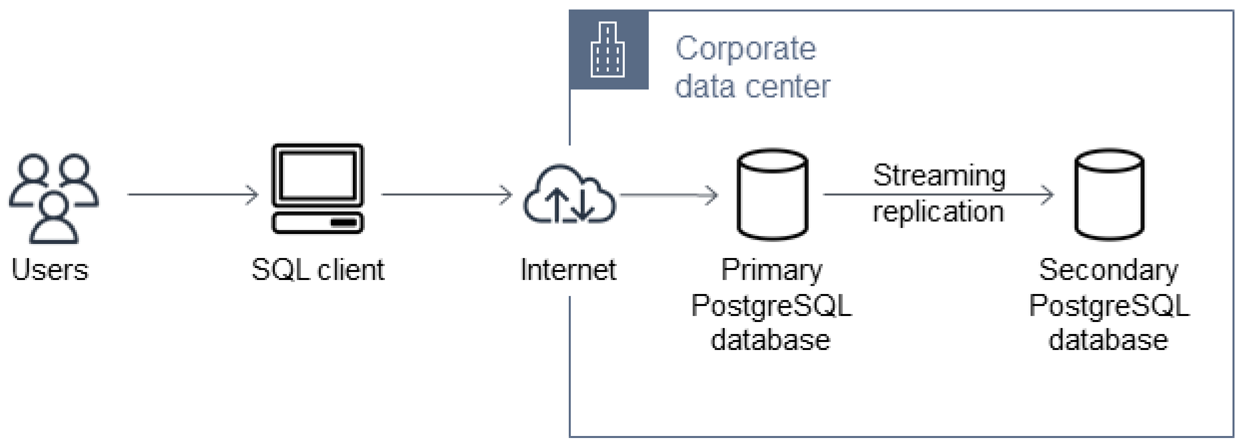
Source technology stack

- On-premises PostgreSQL database

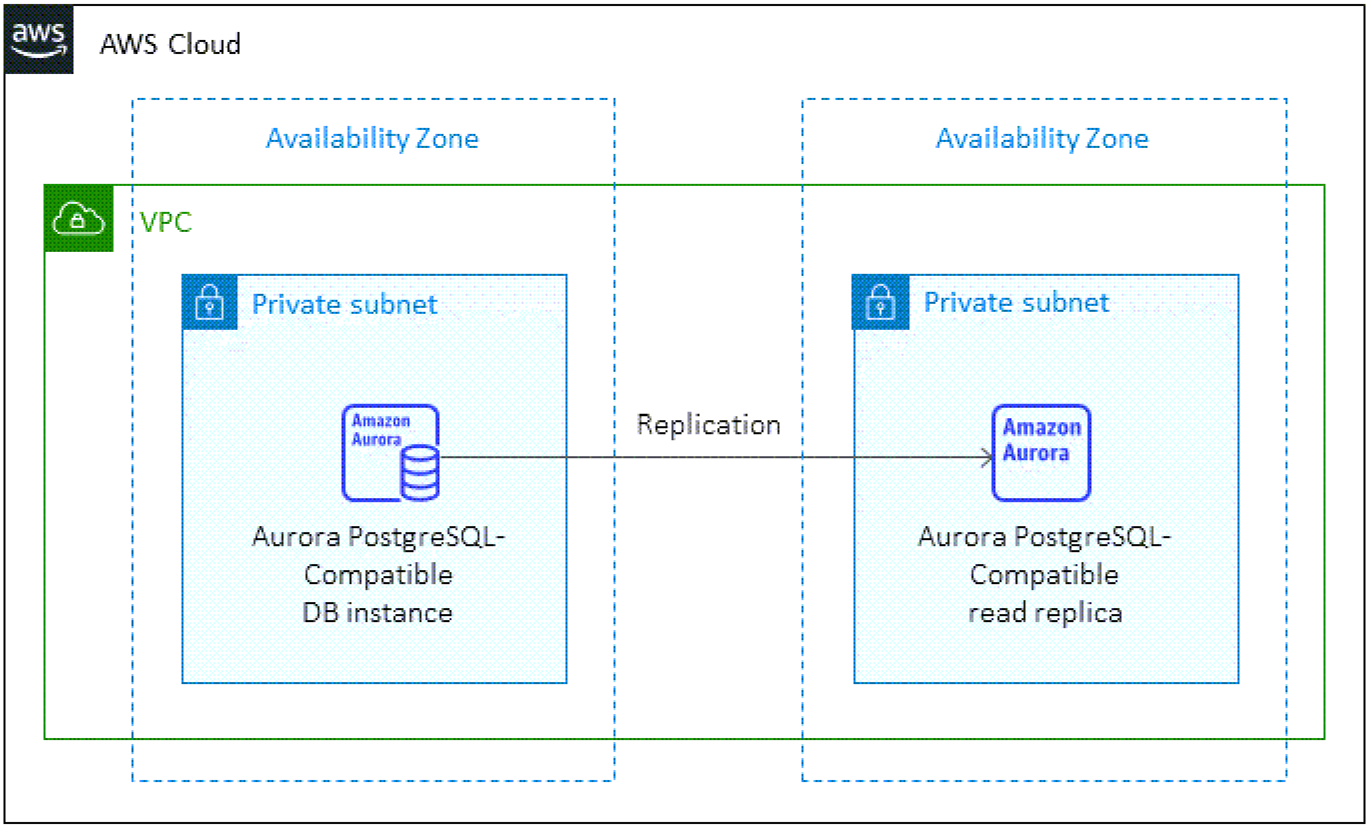
Target technology stack

- Aurora PostgreSQL-Compatible DB instance

Source architecture

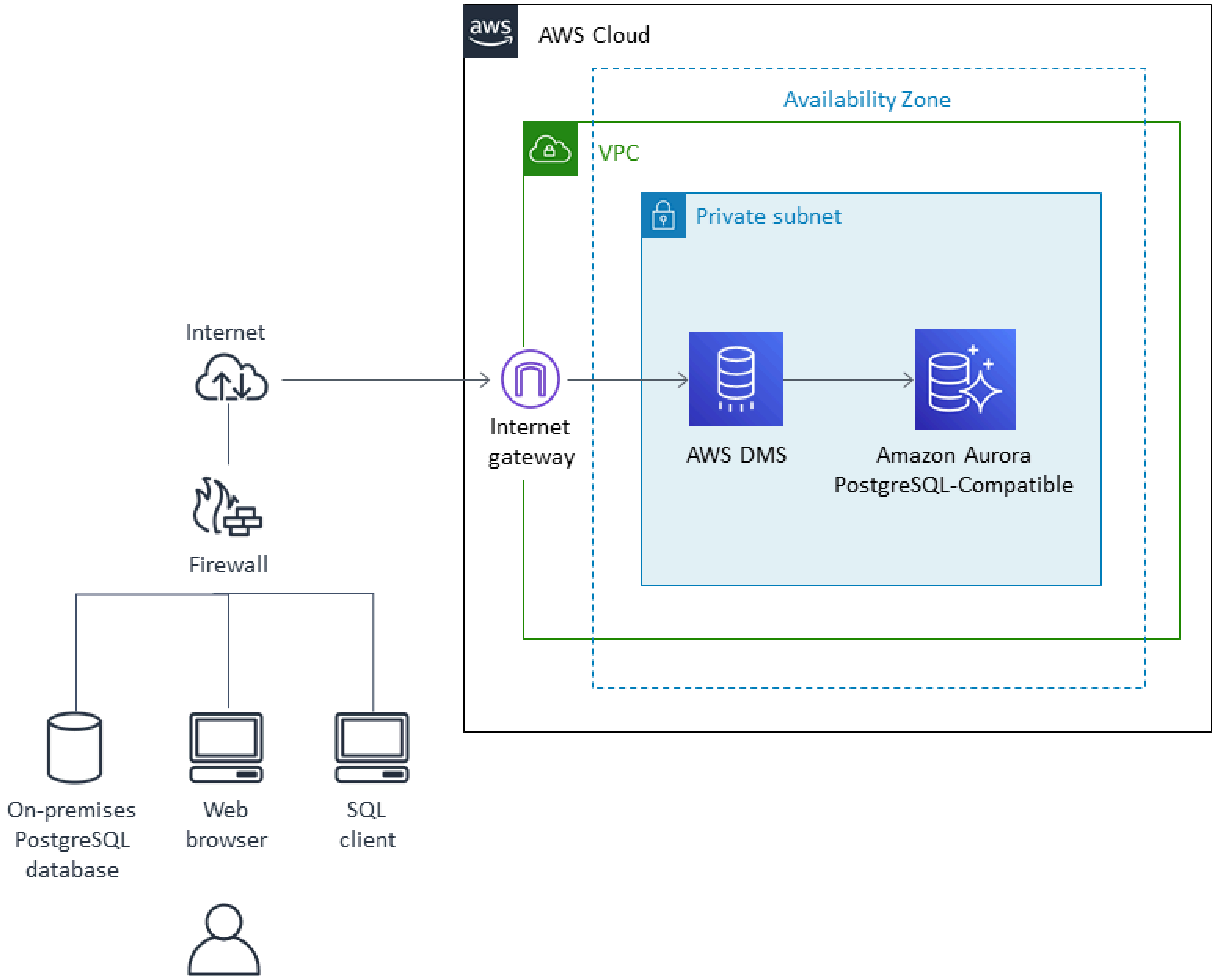


Target architecture

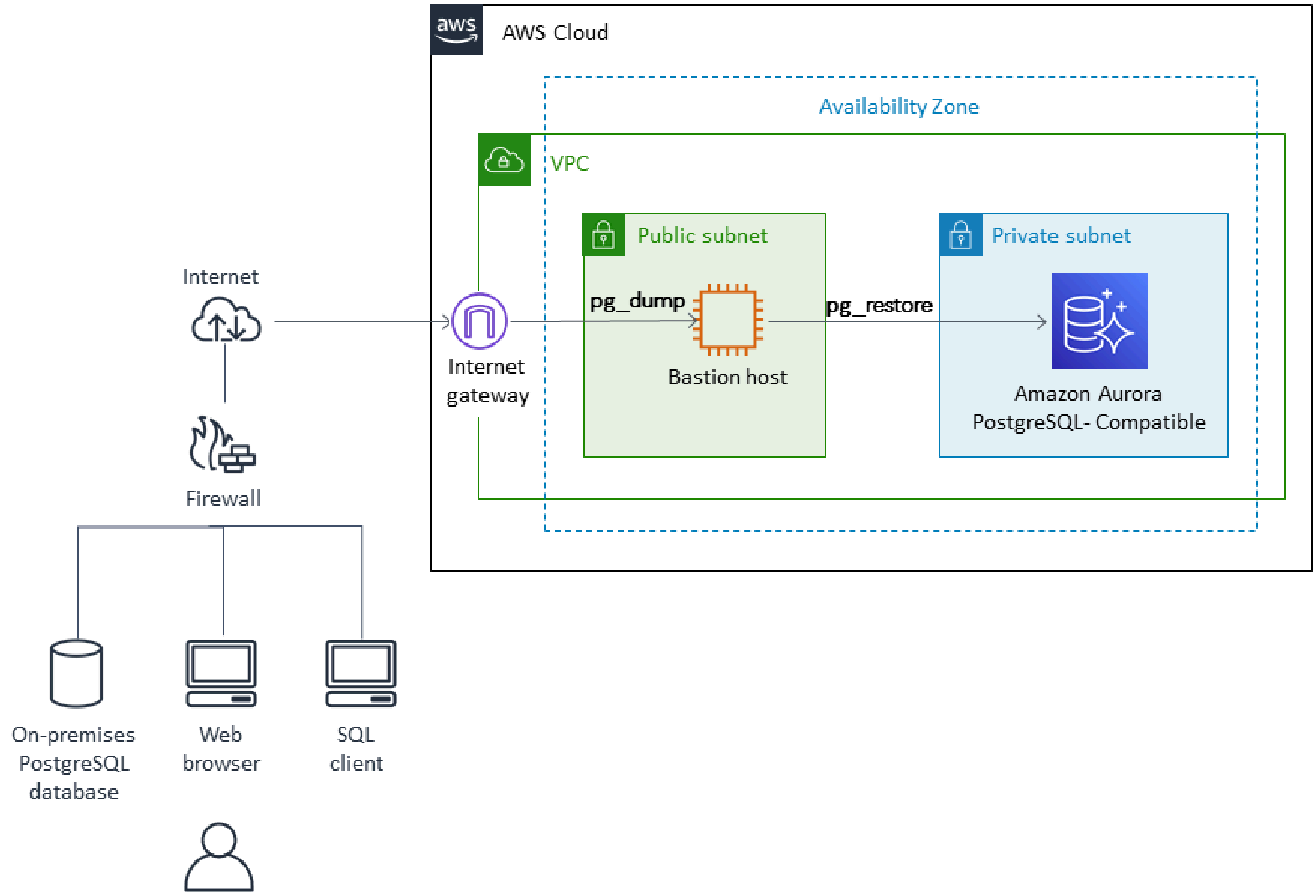


Data migration architecture

Using AWS DMS



Using native PostgreSQL tools



Tools

- AWS Database Migration Service (AWS DMS) helps you migrate data stores into the AWS Cloud or between combinations of cloud and on-premises configurations. This service supports different sources and target databases. For information about how to validate the PostgreSQL source and target database versions and editions supported for use with AWS DMS, see Using a PostgreSQL database as an AWS DMS source. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.
- Native PostgreSQL tools include pg_dump, pg_restore, and psql.

Epics

Analyze the migration

Task	Description	Skills required
Validate the source and target database versions.	If you are using AWS DMS, make sure that you're using a supported version of PostgreSQL.	DBA
Identify the storage type and capacity requirements.	1. Calculate the storage allocated for the source database instance. 2. Gather the historical growth metrics for the source database instance. 3. Anticipate the future growth forecast for the target database instance. 4. Allocate storage by calculating the total number of read and write IOPS on the source database. A General Purpose SSD (gp2) volume provides 3 IOPS for each 1 GB of storage.	DBA, Systems administrator
Choose the proper instance type, capacity, storage features, and network features.	Determine the compute requirements of the target database instance. Review known performance issues that might need additiona l attention. Consider the following factors to	DBA, Systems administrator

Task	Description	Skills required
	determine the appropriate instance type: • CPU utilization of the source database instance • IOPS (read and write operations) for the source database instance • Memory footprint on the source database instance For more information, see Aurora DB instance classes in the Aurora documentation.	
Identify the network access security requirements for the source and target databases.	Determine the appropriate security groups that would enable the application to talk to the database.	DBA, Systems administrator
Identify the application migration strategy.	• Determine the migration cutover strategy based on the complexity of your application. • Determine the recovery time objective (RTO) and recovery point objective (RPO) for the application, and plan for the cutover accordingly.	DBA, App owner, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a VPC.	Create a new virtual private cloud (VPC) for the target database instance.	Systems administrator
Create security groups.	Create a security group within the VPC (as determined in the previous epic) to allow inbound connections to the database instance.	Systems administrator
Configure and start the Aurora DB cluster.	Create the target database instance with the new VPC and security group and start the instance.	Systems administrator

Migrate data – option 1 (using AWS DMS)

Task	Description	Skills required
Complete pre-migration steps.	1. Clean up the data in the source database. 2. Create a replication instance. 3. Create source and target endpoints. 4. Identify the number of available tables and objects to be migrated.	DBA
Complete migration steps.	1. Drop foreign key constrain ts and triggers on the target database.	DBA

Task	Description	Skills required
	2. Drop secondary indexes on the target database. 3. Use a full-load task to migrate data from the source to the target database. 4. Enable foreign keys. 5. If you're using flash-cut migration and your application requires minimal downtime, enable change data capture (CDC) to replicate ongoing changes 6. Enable triggers. 7. Update sequences. 8. Validate the source and target data.	
Validate data.	To ensure that your data was migrated accurately from the source to the target, follow the data validation steps in the AWS DMS documentation.	DBA

Migrate data – option 2 (using pg_dump and pg_restore)

Task	Description	Skills required
Prepare the source database.	1. Create a directory to store pg_dump backup if it doesn't already exist.	DBA

Task	Description	Skills required
	2. Create a migration user that has permissions to run pg_dump on database objects. 3. Connect to the EC2 instance and run pg_dump backup. For more information, see the pg_dump documentation and the walkthrough in the AWS DMS documentation.	
Prepare the target database.	1. Create a migration user that has permissions to use pg_restore on database objects. 2. Import the database dump by using pg_restore. For more information, see the pg_restore documentation and the walkthrough in the AWS DMS documentation.	DBA
Validate data.	1. Compare the database object counts between the source and the target databases. 2. Resolve any discrepancies found between object counts.	DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.	Implement the application migration strategy that you created in the first epic.	DBA, App owner, Systems administrator

Cut over to the target database

Task	Description	Skills required
Switch the application clients over to the new infrastru cture.	1. Stop all application services and client connections that point to the on-premises PostgreSQ L database. 2. Run the AWS DMS tasks. 3. Set up a rollback task (reverse CDC from Aurora PostgreSQL-Compatible to the on-premises PostgreSQ L database) if needed. 4. Validate data. 5. Start the application services on the new target by configuring Amazon Route 53 to the new Aurora PostgreSQL-Compatible DB instance. 6. Add Amazon CloudWatch and Performance Insights monitoring on your new Aurora PostgreSQL-Compatible DB instance.	DBA, App owner, Systems administrator

Task	Description	Skills required
If you need to roll back the migration.	1. Stop all application services that point to the Aurora PostgreSQL-Compatible database. 2. Roll back the changes to the source on-premises PostgreSQL database by using the AWS DMS task you created in the previous story. 3. Stop the AWS DMS tasks running from the on-premises PostgreSQL database to the Aurora PostgreSQL-Compatible database. 4. Configure the application so that it points back to the source on-premises PostgreSQL database. 5. Confirm that all rollback deployment is complete.	DBA, App owner

Close the project

Task	Description	Skills required
Shut down resources.	Shut down the temporary AWS resources.	DBA, Systems administrator
Validate documents.	Review and validate the project documents.	DBA, App owner, Systems administrator

Task	Description	Skills required
Gather metrics.	Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.	DBA, App owner, Systems administrator
Close the project.	Close the project and provide any feedback.	DBA, App owner, Systems administrator

Related resources

References

- AWS Data Migration Service
- VPCs and Amazon Aurora
- Amazon Aurora pricing
- Using a PostgreSQL database as an AWS DMS source
- How to create an AWS DMS replication instance
- How to create source and target endpoints using AWS DMS

Additional resources

- Getting Started with AWS DMS
- Data migration step-by-step walkthroughs
- Amazon Aurora resources

Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux

Tirumala Dasari, Amazon Web Services

Summary

This pattern describes how to migrate from an on-premises Microsoft SQL Server database running on Microsoft Windows, to Microsoft SQL Server on an Amazon Elastic Compute Cloud (Amazon EC2) Linux instance by using backup and restore utilities.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Amazon EC2 Linux AMI (Amazon Machine Image) with Microsoft SQL Server
- AWS Direct Connect between on-premises Windows and Microsoft SQL Server on the Linux EC2 instance

Architecture

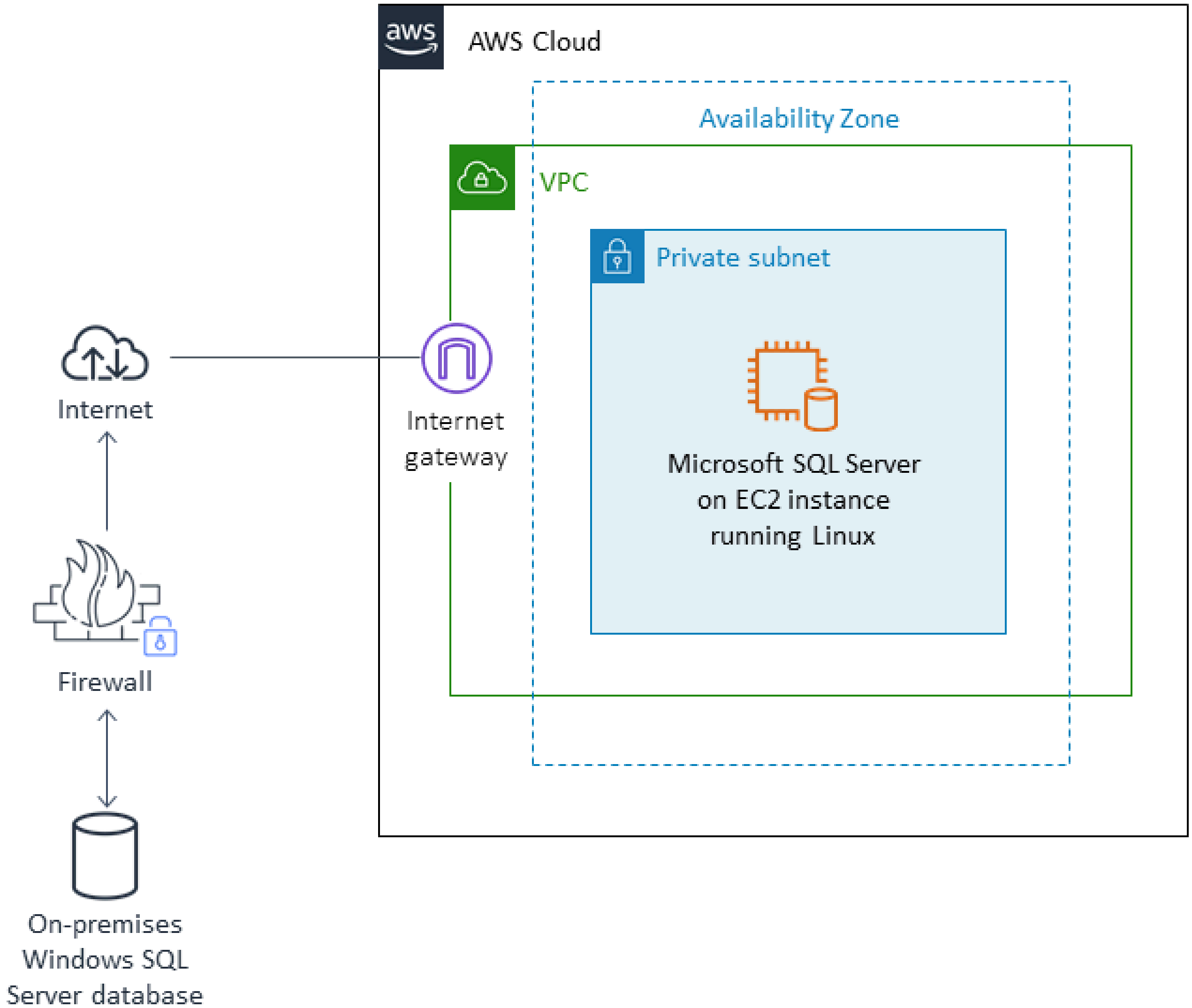
Source technology stack

- On-premises Microsoft SQL Server database

Target technology stack

- Linux EC2 instance with a Microsoft SQL Server database

Database migration architecture



Tools

- **WinSCP** - This tool enables Windows users to easily share files with Linux users.
- **Sqlcmd** - This command-line utility lets you submit T-SQL statements or batches to local and remote instances of SQL Server. The utility is extremely useful for repetitive database tasks such as batch processing or unit testing.

Epics

Prepare the EC2 Linux instance with SQL Server

Task	Description	Skills required
Select an AMI that provides the Linux operating system and includes Microsoft SQL Server.		Sysadmin
Configure the AMI to create an EC2 instance.		Sysadmin
Create inbound and outbound rules for security groups.		Sysadmin
Configure the Linux EC2 instance for a Microsoft SQL Server database.		DBA
Create users and provide permissions as in the source database.		Appowner, DBA
Install SQL Server tools and the sqlcmd utility on the Linux EC2 instance.		DBA

Back up the database and move backup file to Linux EC2 instance

Task	Description	Skills required
Back up the on-premises SQL Server database.		DBA
Install WinSCP on Microsoft SQL Server.		DBA

Task	Description	Skills required
Move the backup file to the Linux EC2 instance running Microsoft SQL Server.		DBA

Restore the database on Linux EC2 instance running SQL Server

Task	Description	Skills required
Restore the database from the database backup file by using the sqlcmd utility.		DBA
Validate database objects and data.		Developer, Test engineer

Cut over from Windows SQL Server to Windows SQL Server on Linux EC2 instance

Task	Description	Skills required
Validate database objects and data.		Developer, Test engineer
Cut over from the on-premis es Microsoft SQL Server database to the Linux EC2 instance running Microsoft SQL Server.		DBA

Related resources

- [How to configure SQL Server 2017 on Amazon Linux and Ubuntu AMIs]
- [Installation of SQL tools on a Linux instance]

- Backup and restoration from an on-premises Microsoft SQL Server database to Microsoft SQL Server on a Linux EC2 instance

Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers

Kevin Yung, Viqash Adwani, and Vishal Singh, Amazon Web Services

Summary

Linked servers enable Microsoft SQL Server to run SQL statements on other instances of database servers. This pattern describes how you can migrate your on-premises Microsoft SQL Server database to Amazon Relational Database Service (Amazon RDS) for Microsoft SQL Server to achieve lower cost and higher availability. Currently, Amazon RDS for Microsoft SQL Server doesn't support connections outside an Amazon Virtual Private Cloud (Amazon VPC) network.

You can use this pattern to achieve the following objectives:

- To migrate Microsoft SQL Server to Amazon RDS for Microsoft SQL Server without breaking linked server capabilities.
- To prioritize and migrate linked Microsoft SQL Server in different waves.

Prerequisites and limitations

Prerequisites

- Check whether Microsoft SQL Server on Amazon RDS supports the features you require.
- Make sure that you can use either Amazon RDS for Microsoft SQL Server with default collations or collations set over database levels.

Architecture

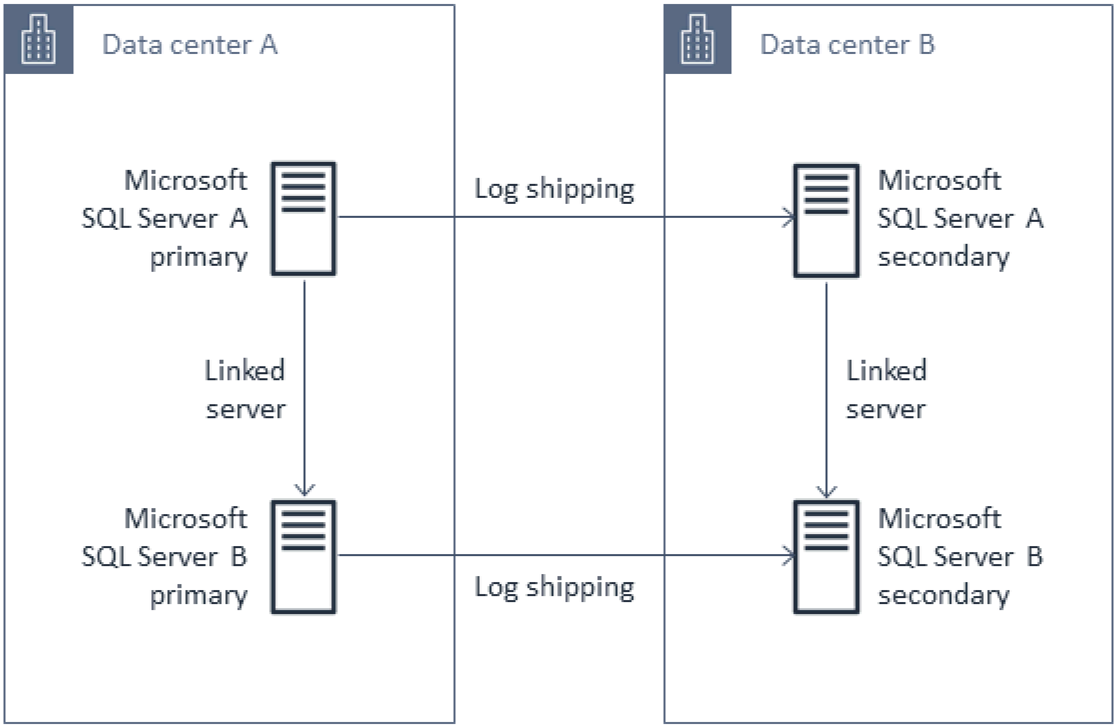
Source technology stack

- On-premises databases (Microsoft SQL Server)

Target technology stack

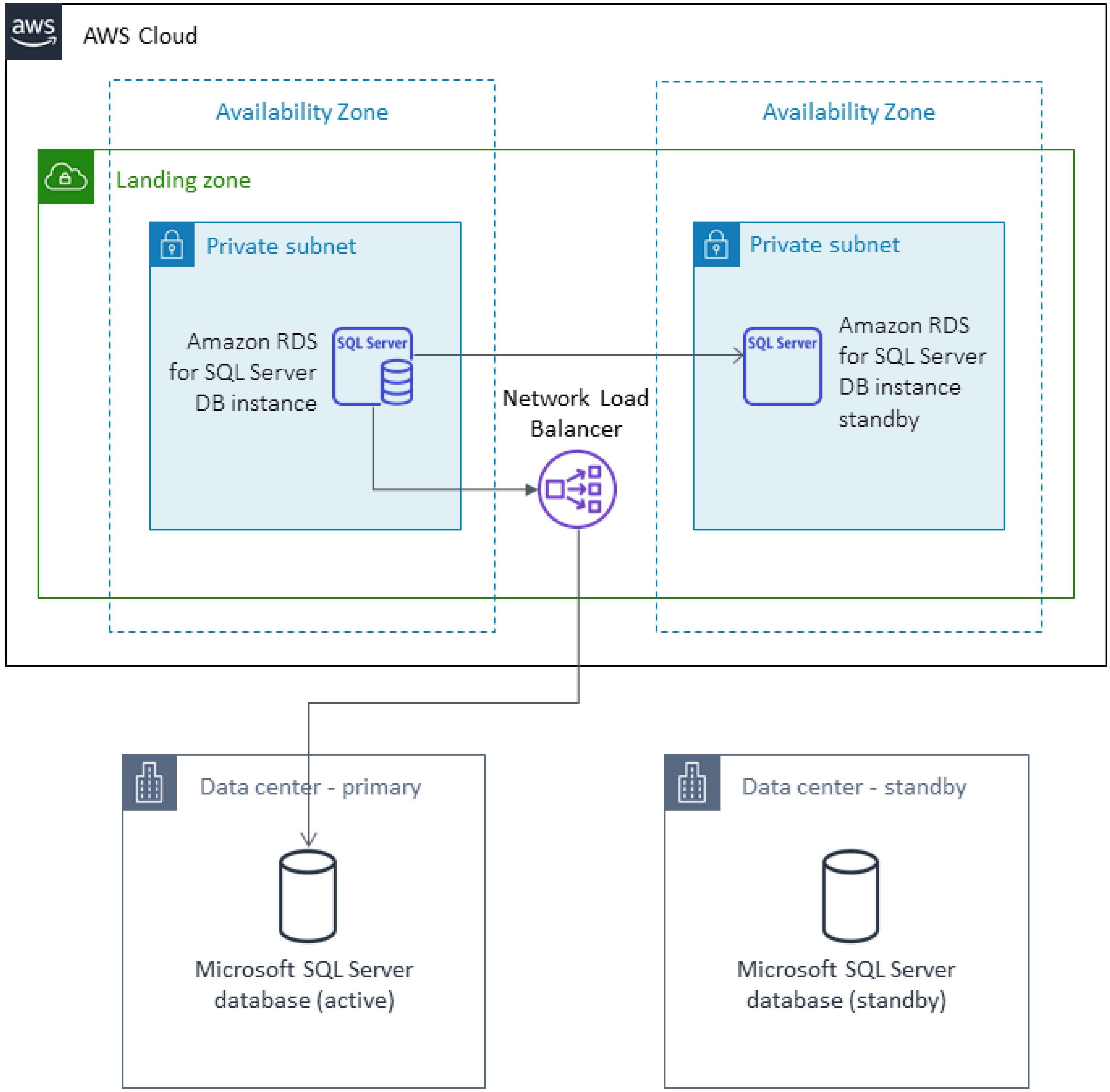
- Amazon RDS for SQL Server

Source state architecture



Target state architecture

In the target state, you migrate Microsoft SQL Server to Amazon RDS for Microsoft SQL Server by using linked servers. This architecture uses a Network Load Balancer to proxy the traffic from Amazon RDS for Microsoft SQL Server to on-premises servers running Microsoft SQL Server. The following diagram shows the reverse proxy capability for the Network Load Balancer.



Tools

- AWS CloudFormation
- Network Load Balancer
- Amazon RDS for SQL Server in multiple Availability Zones (Multi-AZs)

- AWS Database Migration Service (AWS DMS)

Epics

Create a landing zone VPC

Task	Description	Skills required
Create the CIDR allocation.		AWS SysAdmin
Create a virtual private cloud (VPC).		AWS SysAdmin
Create the VPC subnets.		AWS SysAdmin
Create the subnet access control lists (ACLs).		AWS SysAdmin
Create the subnet route tables.		AWS SysAdmin
Create a connection with AWS Direct Connect or AWS Virtual Private Network (VPN).		AWS SysAdmin

Migrate the database to Amazon RDS

Task	Description	Skills required
Create an Amazon RDS for Microsoft SQL Server DB instance.		AWS SysAdmin
Create an AWS DMS replicati on instance.		AWS SysAdmin

Task	Description	Skills required
Create the source and target database endpoints in AWS DMS.		AWS SysAdmin
Create the migration task and set continuous replication to ON after a full load.		AWS SysAdmin
Request a firewall change to allow Amazon RDS for Microsoft SQL Server to access the on-premises SQL Server databases.		AWS SysAdmin
Create a Network Load Balancer.		AWS SysAdmin
Create a target group that targets the database servers in your data center	We recommend that you use hostnames in the target setup to incorporate data center (DC) failover events.	AWS SysAdmin
Run the SQL statement for linked server setup.	Run the SQL statements for adding a linked server by using the Microsoft SQL management tool against the Amazon RDS for Microsoft SQL Server DB instance. In the SQL statement, set @datasrc to use the Network Load Balancer hostname. Add linked server login credentia ls by using the Microsoft SQL management tool against the Amazon RDS for Microsoft SQL Server DB instance.	AWS SysAdmin

Task	Description	Skills required
Test and validate the SQL Server functions.		AWS SysAdmin
Create a cutover.		AWS SysAdmin

Related resources

- Common Management Tasks for Microsoft SQL Server on Amazon RDS
- Collations and Character Sets for Microsoft SQL Server
- Network Load Balancer documentation
- Implement Linked Servers with Amazon RDS for Microsoft SQL Server (blog post)

Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using native backup and restore methods

Tirumala Dasari, David Queiroz, and Vishal Singh, Amazon Web Services

Summary

This pattern describes how to migrate an on-premises Microsoft SQL Server database to an Amazon Relational Database Service (Amazon RDS) for SQL Server DB instance (homogeneous migration). The migration process is based on native SQL Server backup and restore methods. It uses SQL Server Management Studio (SSMS) to create a database backup file, and an Amazon Simple Storage Service (Amazon S3) bucket to store the backup file before restoring it in Amazon RDS for SQL Server.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Identity and Access Management (IAM) role policies to access the S3 bucket and the Amazon RDS for SQL Server DB instance.

Limitations

- The process described in this pattern migrates only the database. SQL logins or database users, including any SQL Server Agent jobs, aren't migrated because they require additional steps.

Product versions

- SQL Server 2012-2017. For the latest list of supported versions and features, see Microsoft SQL Server on Amazon RDS in the AWS documentation.

Architecture

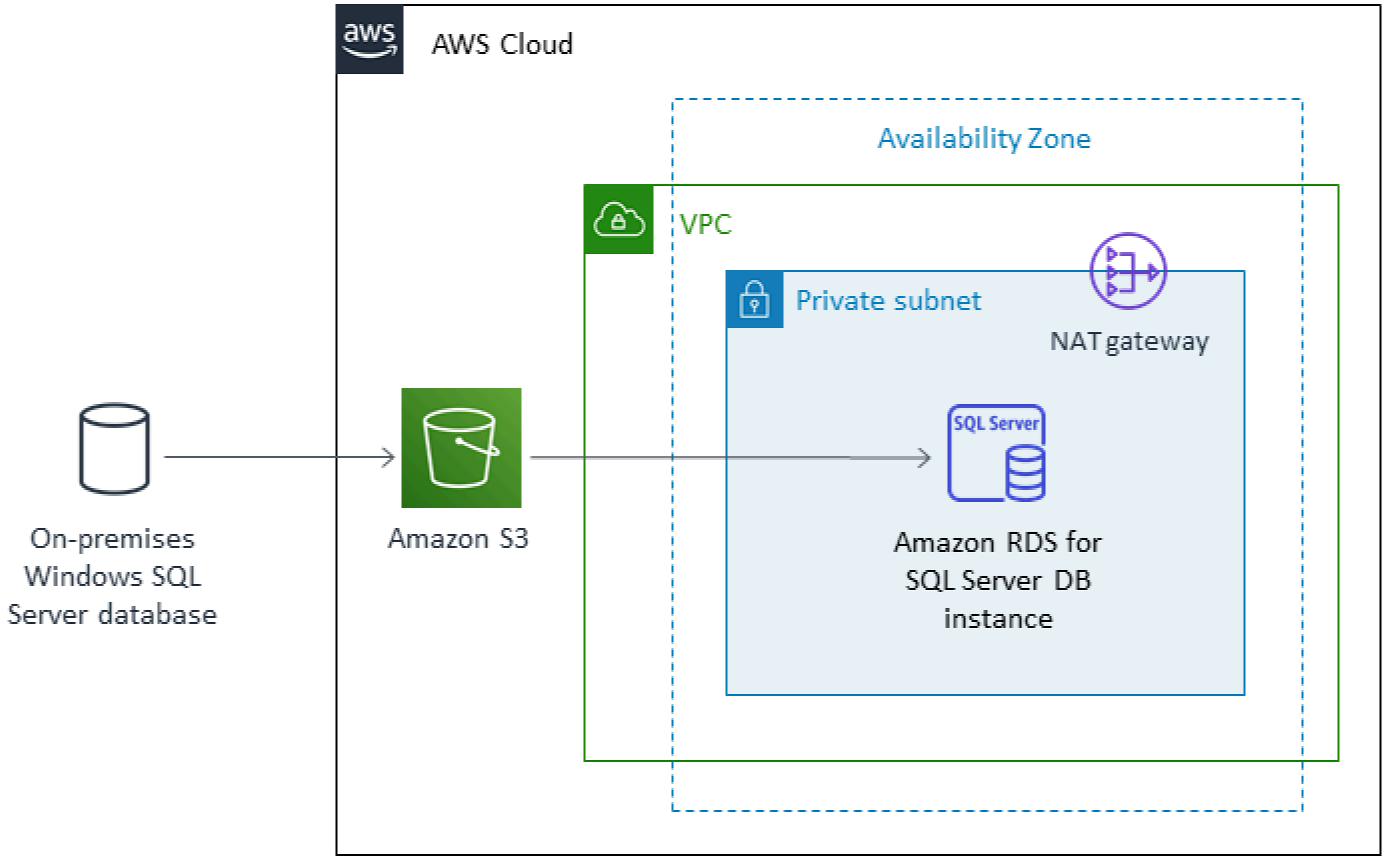
Source technology stack

- An on-premises Microsoft SQL Server database

Target technology stack

- Amazon RDS for SQL Server DB instance

Data migration architecture



Tools

- Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

Epics

Create an Amazon RDS for SQL Server DB instance

Task	Description	Skills required
Select SQL Server as the database engine in Amazon RDS for SQL Server.		DBA
Choose the SQL Server Express Edition.		DBA
Specify database details.	For more information about creating a DB instance, see the Amazon RDS documentation.	DBA, App owner

Create a backup file from the on-premises SQL Server database

Task	Description	Skills required
Connect to the on-premises SQL Server database through SSMS.		DBA
Create a backup of the database.	For instructions, see the SSMS documentation.	DBA, App owner

Upload the backup file to Amazon S3

Task	Description	Skills required
Create a bucket in Amazon S3.	For more information, see the Amazon S3 documentation.	DBA

Task	Description	Skills required
Upload the backup file to the S3 bucket.	For more information, see the [Amazon S3 documentation]().	SysOps administrator

Restore the database in Amazon RDS for SQL Server

Task	Description	Skills required
Add the option group to Amazon RDS.	1. Open the Amazon RDS console at https://console.aws.amazon.com/rds/. 2. In the navigation pane, choose **Option groups**, **Create group**. 3. Complete the information for the option group, and then choose **Create**. 4. Add the `SQLSERVER_BACKUP_RESTORE` option to the option group, and then choose **Add option**. For more information, see the [Amazon RDS documentation]().	SysOps administrator
Restore the database.	1. Connect to Amazon RDS for SQL Server through SSMS. 2. Call the `msdb.dbo.rds_restore_database` stored procedure to restore the database.	DBA

Validate the target database

Task	Description	Skills required
Validate objects and data.	Validate the objects and data between the source database and Amazon RDS for SQL Server. **Note** This task migrates the database only. Logins and jobs will not be migrated.	App owner, DBA

Cut over

Task	Description	Skills required
Redirect application traffic.	After validation, redirect application traffic to the Amazon RDS for SQL Server DB instance.	App owner, DBA

Related resources

- Amazon S3 documentation
- Amazon RDS for SQL Server documentation
- Options for the Microsoft SQL Server Database Engine

Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT

Mark Szalkiewicz and Pavan Pusuluri, Amazon Web Services

Summary

This pattern describes how to migrate a Microsoft SQL Server database that is either on premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance to Amazon Aurora MySQL. The pattern uses AWS Database Migration Service (AWS DMS) and AWS Schema Conversion Tool (AWS SCT) for data migration and schema conversion.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Microsoft SQL Server source database in an on-premises data center or on an EC2 instance
- Java Database Connectivity (JDBC) drivers for AWS SCT connectors, installed on either a local machine or an EC2 instance where AWS SCT is installed

Limitations

- Database size limit: 64 TB

Product versions

- Microsoft SQL Server 2008, 2008R2, 2012, 2014, 2016, and 2017 for the Enterprise, Standard, Workgroup, and Developer editions. The Web and Express editions aren't supported by AWS DMS. For the latest list of supported versions, see Using a Microsoft SQL Server Database as a Source for AWS DMS. We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support. For information about Microsoft SQL Server versions supported by AWS SCT, see the AWS SCT documentation.
- MySQL versions 5.5, 5.6, and 5.7. For the latest list of supported versions, see Using a MySQL-Compatible Database as a Target for AWS DMS.

Architecture

Source technology stack

One of the following:

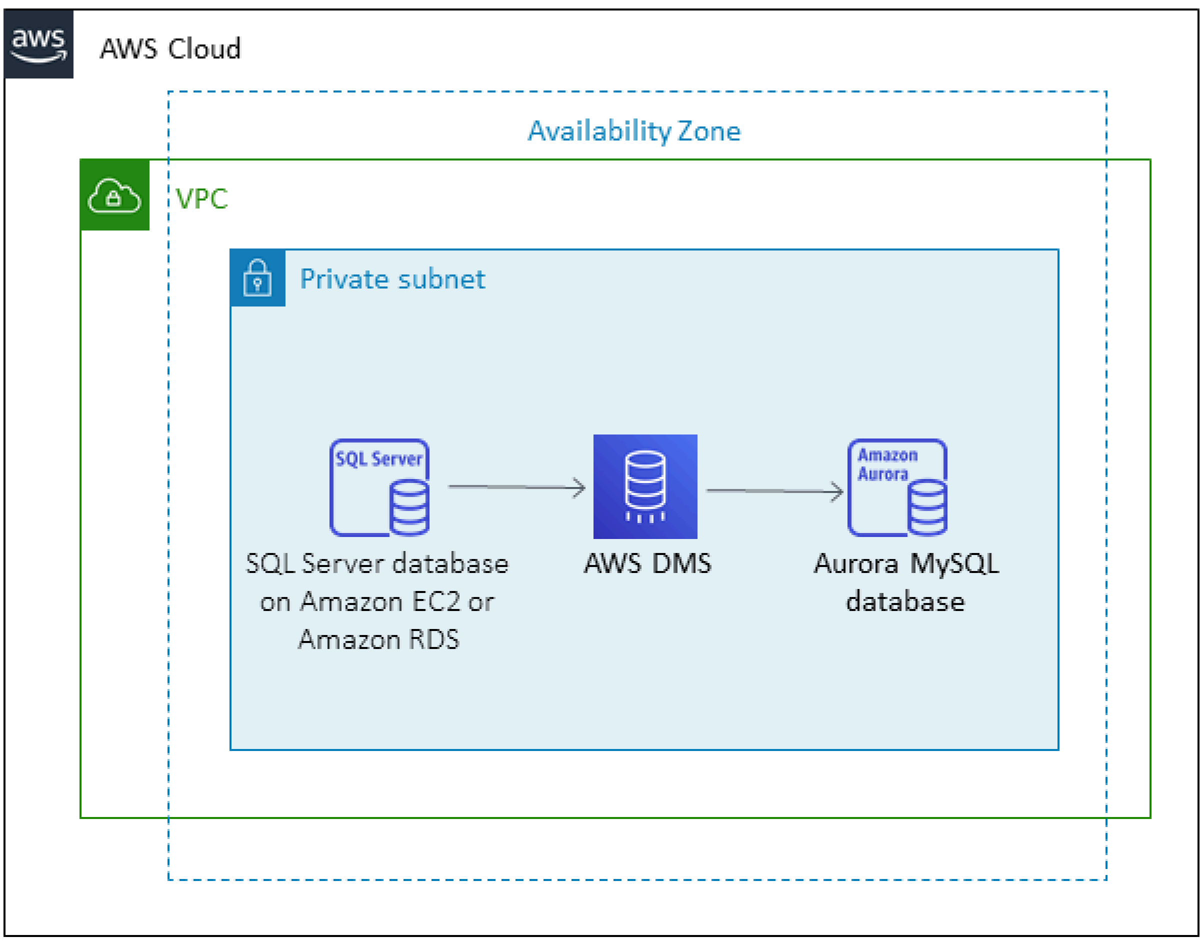
- An on-premises Microsoft SQL Server database
- A Microsoft SQL Server database on an EC2 instance

Target technology stack

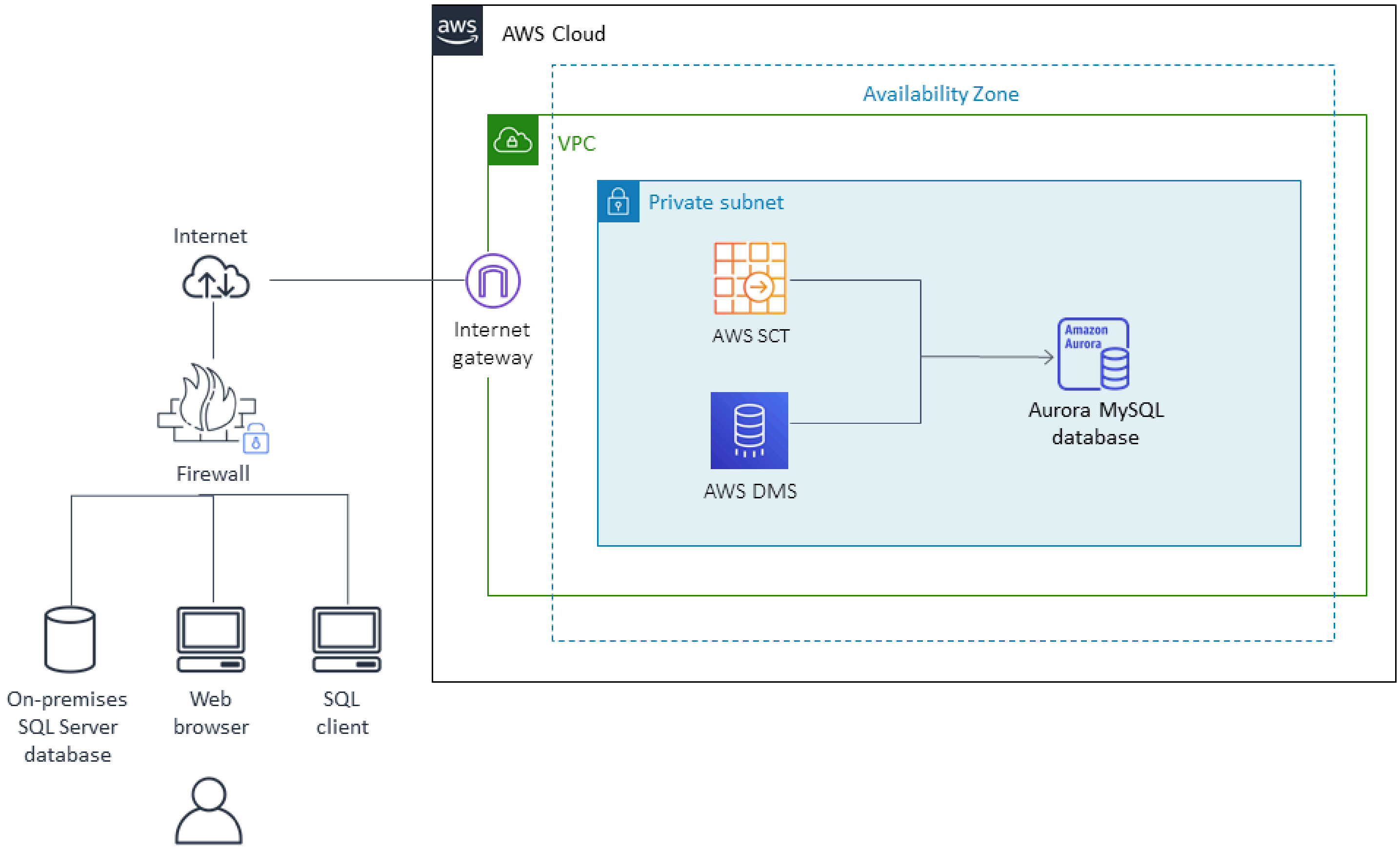
- Aurora MySQL

Data migration architecture

- From a Microsoft SQL Server database running in the AWS Cloud



- From a Microsoft SQL Server database running in an on-premises data center



Tools

- **AWS DMS** - [AWS Data Migration Service](https://docs.aws.amazon.com/dms/) (AWS DMS) helps you migrate your data to and from widely used commercial and open-source databases, including Oracle, SQL Server, MySQL, and PostgreSQL. You can use AWS DMS to migrate your data into the AWS Cloud, between on-premises instances (through an AWS Cloud setup), or between combinations of cloud and on-premises setups.
- **AWS SCT** - [AWS Schema Conversion Tool](https://docs.aws.amazon.com/SchemaConversionTool/) (AWS SCT) makes heterogeneous database migrations easy by automatically converting the source database schema and a majority of the custom code to a format compatible with the target database.

Epics

Prepare for the migration

Task	Description	Skills required
Validate the source and target database version and engine.		DBA
Create an outbound security group for the source and target databases.		SysAdmin
Create and configure an EC2 instance for AWS SCT, if required.		DBA
Download the latest version of AWS SCT and associated drivers.		DBA
Add and validate the prerequisite users and grants in the source database.		DBA
Create an AWS SCT project for the workload and connect to the source database.		DBA
Generate an assessment report and evaluate feasibili ty.		DBA

Prepare the target database

Task	Description	Skills required
Create a target Amazon RDS DB instance, using Amazon Aurora as the database engine.		DBA
Extract the list of users, roles, and grants from the source.		DBA
Map the existing database users to the new database users.		App owner
Create users in the target database.		DBA
Apply roles from the previous step to the target database.		DBA
Review the database options, parameters, network files, and database links in the source database, and then evaluate their applicability to the target database.		DBA
Apply any relevant settings to the target.		DBA

Transfer objects

Task	Description	Skills required
Configure AWS SCT connectiv ity to the target database.		DBA

Task	Description	Skills required
Convert the schema using AWS SCT.	AWS SCT automatically converts the source database schema and most of the custom code to a format that is compatible with the target database. Any code that the tool cannot convert automatically is clearly marked so that you can convert it yourself.	DBA
Review the generated SQL report and save any errors and warnings.		DBA
Apply automated schema changes to the target or save them as a .sql file.		DBA
Validate that AWS SCT created the objects on the target.		DBA
Manually rewrite, reject, or redesign any items that failed to convert automatically.		DBA
Apply the generated role and user grants and review any exceptions.		DBA

Migrate the data

Task	Description	Skills required
Determine the migration method.		DBA
Create a replication instance from the AWS DMS console.	For detailed information on using AWS DMS, see the links in the "Related resources" section.	DBA
Create the source and target endpoints.		DBA
Create a replication task.		DBA
Start the replication task and monitor the logs.		DBA

Migrate the application

Task	Description	Skills required
Use AWS SCT to analyze and convert the SQL items within the application code.	When you convert your database schema from one engine to another, you also need to update the SQL code in your applications to interact with the new database engine instead of the old one. You can view, analyze, edit, and save the converted SQL code. For detailed information on using AWS SCT, see the links in the "Related resources" section.	App owner

Task	Description	Skills required
Create the new application servers on AWS.		App owner
Migrate the application code to the new servers.		App owner
Configure the application server for the target database and drivers.		App owner
Fix any code that's specific to the source database engine in the application.		App owner
Optimize the application code for the target engine.		App owner

Cut over

Task	Description	Skills required
Apply any new users, grants, and code changes to the target.		DBA
Lock the application for any changes.		App owner
Validate that all changes were propagated to the target database.		DBA
Point the new application server to the target database.		App owner
Recheck everything.		App owner

Task	Description	Skills required
Go live.		App owner

Close the project

Task	Description	Skills required
Shut down the temporary AWS resources (AWS DMS replication instance and EC2 instance used for AWS SCT).		DBA, App owner
Update feedback on the AWS DMS process for internal teams.		DBA, App owner
Revise the AWS DMS process and improve the template if necessary.		DBA, App owner
Review and validate the project documents.		DBA, App owner
Gather metrics around time to migrate, percent of manual versus tool cost savings, and so on.		DBA, App owner
Close the project and provide any feedback.		DBA, App owner

Related resources

References

- AWS DMS User Guide

- AWS SCT User Guide
- Amazon Aurora Pricing

Tutorials and videos

- Getting Started with AWS Database Migration Service
- Getting Started with the AWS Schema Conversion Tool
- Amazon RDS resources
- AWS DMS Step-by-Step Walkthroughs

Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools

Shyam Sunder Rakhecha, Amazon Web Services

Summary

This pattern provides guidance for migrating an on-premises MariaDB database to Amazon Relational Database Service (Amazon RDS) for MariaDB by using native tools. If you have MySQL tools installed, you can use **mysql** and **mysqldump**. If you have MariaDB tools installed, you can use **mariadb** and **mariadb-dump**. MySQL and MariaDB tools have the same origin, but there are minor differences in MariaDB version 10.6 and later.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A MariaDB source database in an on-premises data center

Limitations

- Database size limit: 64 TB

Product versions

- MariaDB versions 10.0-10.6 (for the latest list of supported versions, see [MariaDB on Amazon RDS](#) in the AWS documentation)

Architecture

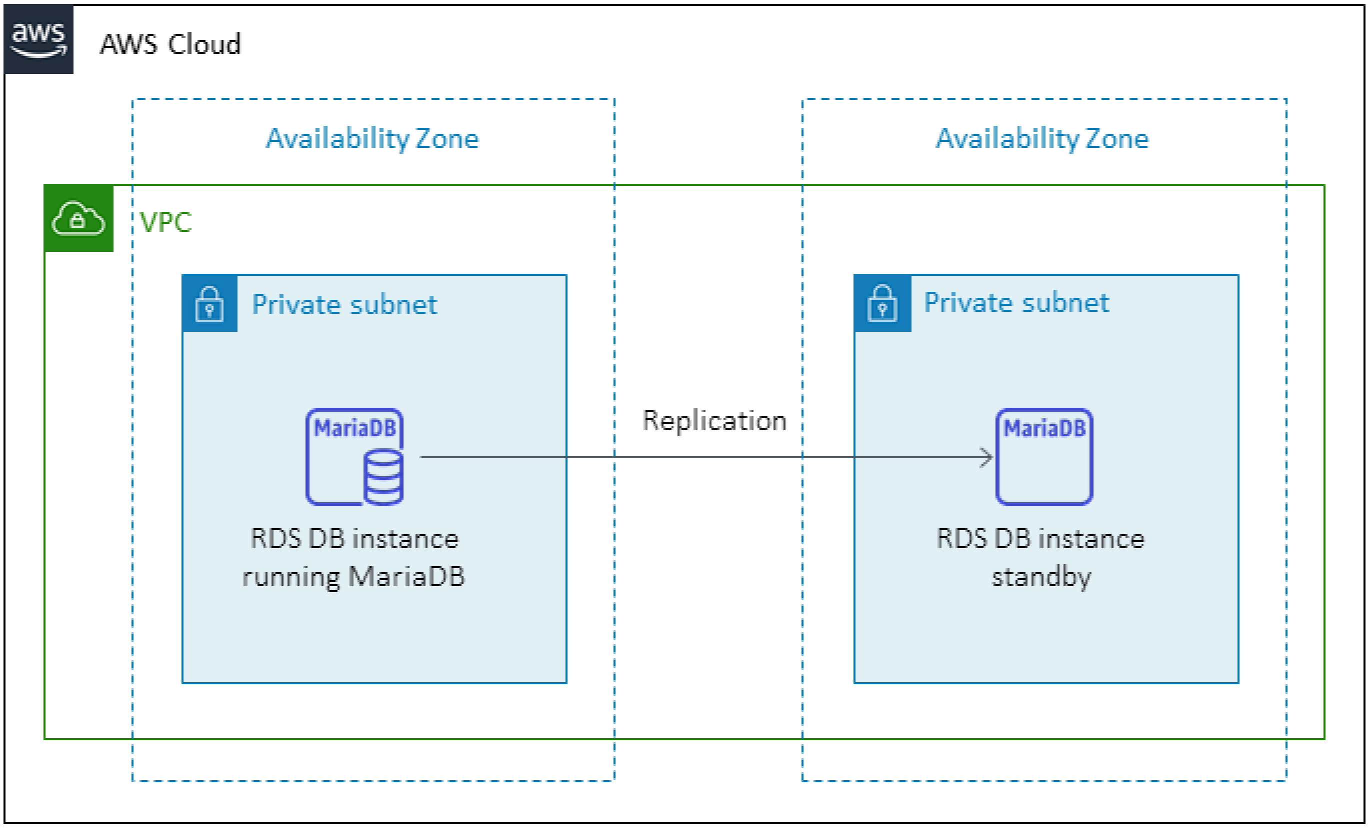
Source technology stack

- MariaDB database in an on-premises data center

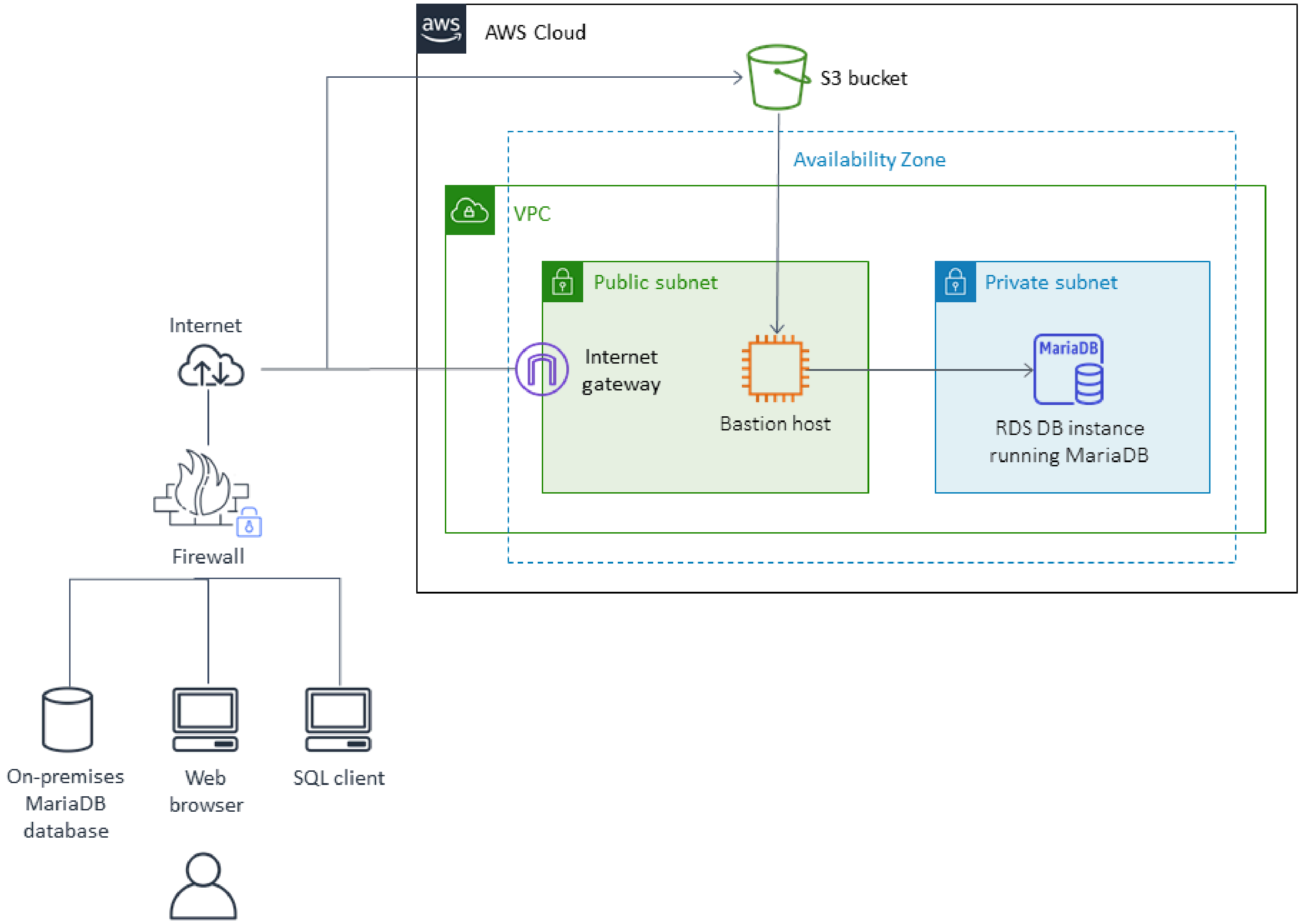
Target technology stack

- Amazon RDS for MariaDB DB instance

Target architecture



Data migration architecture



Tools

- Native MySQL tools: **mysql** and **mysqldump**
- Native MariaDB tools: **mariadb** and **mariadb-dump**

Epics

Plan the migration

Task	Description	Skills required
Validate source and target database versions and engines.		DBA

Task	Description	Skills required
Identify hardware requireme nts for the target server instance.		DBA, Systems administrator
Identify storage requirements (storage type and capacity).		DBA, Systems administrator
Choose the proper instance type based on capacity, storage features, and network features.		DBA, Systems administrator
Identify the network access security requirements for source and target databases.		DBA, Systems administrator
Identify the application migration strategy.		DBA, App owner, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC).		Systems administrator
Create security groups.		Systems administrator
Configure and start an Amazon RDS DB instance running MariaDB.		Systems administrator

Migrate data

Task	Description	Skills required
Use native tools to migrate database objects and data.	In the source database, use **mysqldump** or **mariadb-dump** to create an output file that contains database objects and data. In the target database, use **mysql** or **mariadb** to restore the data.	DBA
Validate the data.	Check the source and target databases to confirm that the data migration was successfu l.	DBA

Migrate the application

Task	Description	Skills required
Follow the application migration strategy.		DBA, App owner, Systems administrator

Cut over

Task	Description	Skills required
Switch the application clients over to the new infrastru cture.		DBA, App owner, Systems administrator

Close the project

Task	Description	Skills required
Shut down the temporary AWS resources.		Systems administrator
Review and validate the project documents.		DBA, App owner, Systems administrator
Gather metrics around time to migrate, cost savings provided by tools, and so on.		DBA, App owner, Systems administrator
Close out the project and provide feedback.		DBA, App owner, Systems administrator

Related resources

Amazon RDS references

- Amazon RDS for MariaDB
- Amazon Virtual Private Cloud VPCs and Amazon RDS
- Amazon RDS Multi-AZ Deployments
- Amazon RDS Pricing

MySQL and MariaDB references

- mariadb-dump/mysqldump
- mysql Command-line Client

Tutorials and videos

- Getting Started with Amazon RDS

Migrate an on-premises MySQL database to Aurora MySQL

Igor Obradovic, Amazon Web Services

Summary

This pattern explains how to migrate an on-premises MySQL source database to Amazon Aurora MySQL-Compatible Edition. It describes two options for migration: using AWS Database Migration Service (AWS DMS) or using native MySQL tools such as **mysqldbcopy** and **mysqldump**.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A source MySQL database in an on-premises data center

Limitations

- Database size limit: 128 TB

Product versions

- MySQL version 8.0 (Aurora MySQL version 3) is available under standard support.
- MySQL version 5.7 (Aurora MySQL version 2) is available under extended support, for an additional cost.

For the latest list of supported versions, see [Amazon Aurora versions] in the AWS documentation. If you're using AWS DMS, see also [Using a MySQL-Compatible Database as a Target for AWS DMS] for MySQL versions supported by AWS DMS.

Architecture

Source technology stack

- An on-premises MySQL database

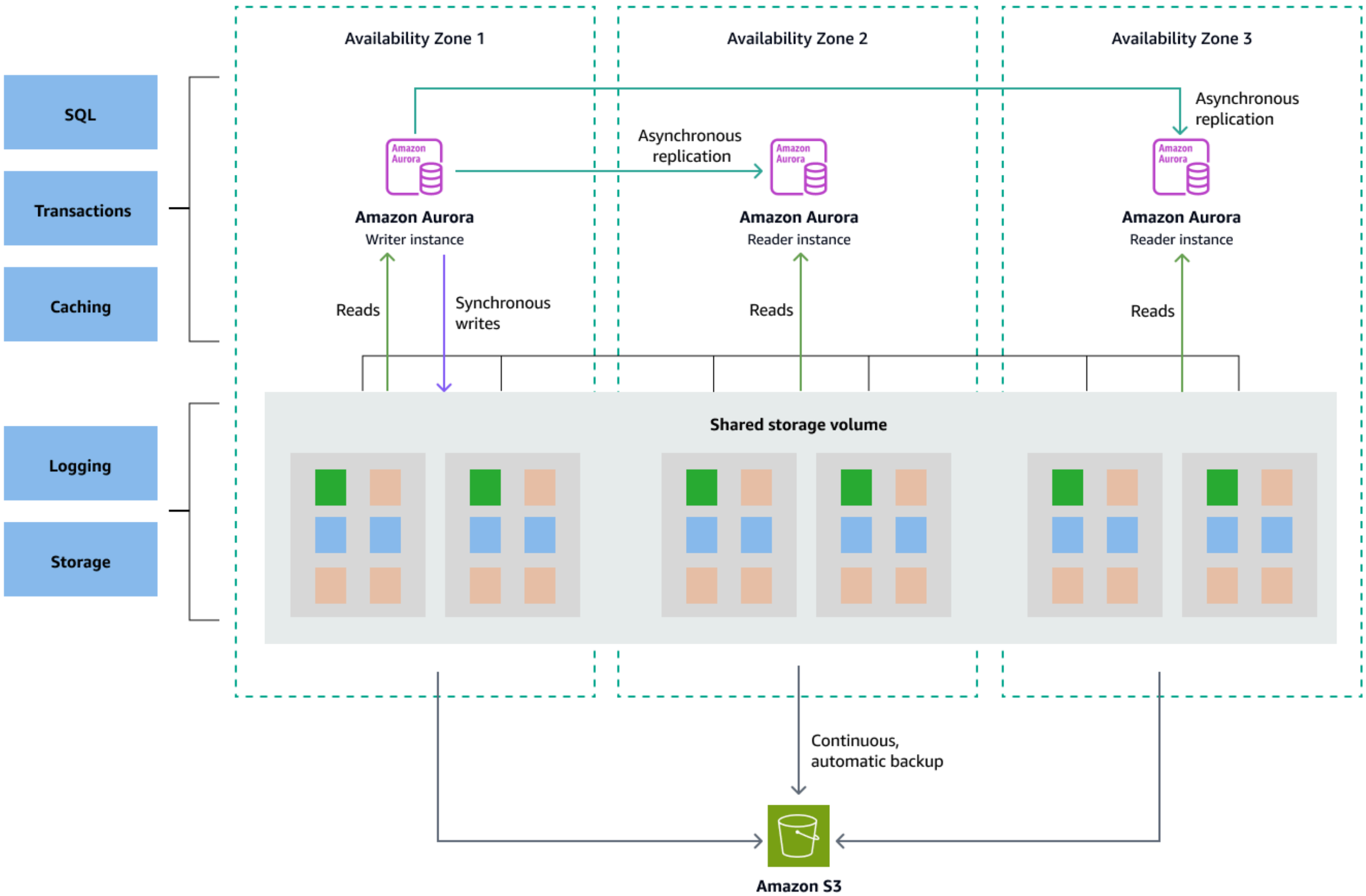
Target technology stack

- Amazon Aurora MySQL-Compatible Edition

Target architecture

Aurora data is stored in a cluster volume, which is a single, virtual volume that uses solid state drives (SSDs). A cluster volume consists of copies of the data across three Availability Zones in a single AWS Region. Because the data is automatically replicated across Availability Zones, it is highly durable with less possibility of data loss.

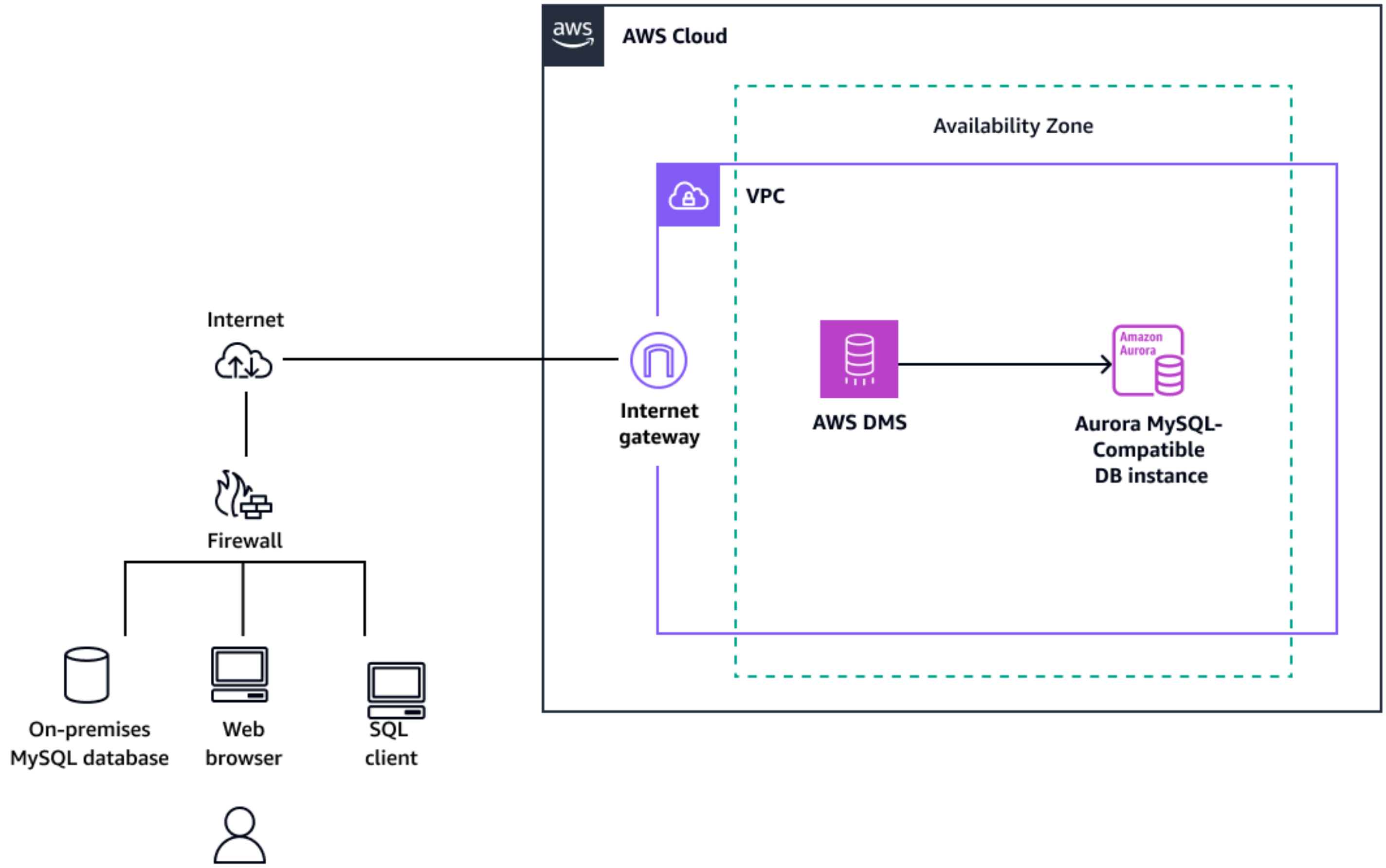
Aurora automatically divides your database volume into 10 GB segments spread across many disks. Each 10 GB chunk of your database volume is replicated six ways, across three Availability Zones. The following diagram illustrates the relationship between the cluster volume, the writer DB instance, and reader DB instances in an Aurora DB cluster, and the separation of compute capacity and storage. For more information about this architecture, see the [Aurora documentation](#) and [FAQ](#).



Data migration architecture

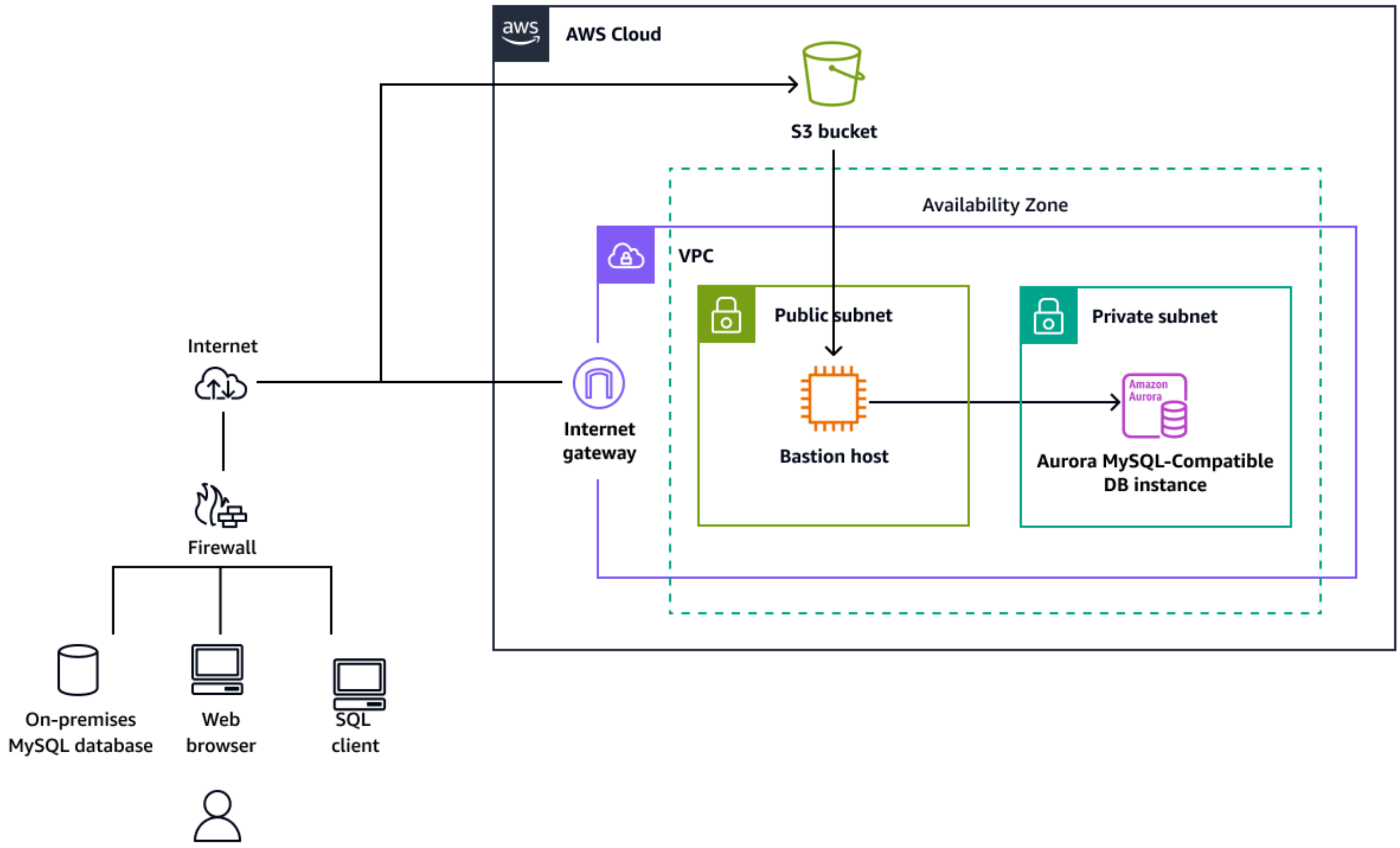
Using AWS DMS:

The following diagram illustrates the migration of an on-premises MySQL database to an Aurora MySQL-Compatible cluster in the AWS Cloud, using AWS DMS.



Using native MySQL tools:

The following diagram illustrates the migration of an on-premises MySQL database to an Aurora MySQL-Compatible cluster in the AWS Cloud, using native MySQL tools such as **mysqldbcopy** and **mysqldump**.



Tools

- [AWS Database Migration Service (AWS DMS)](#) supports several source and target database engines. For information about MySQL source and target databases supported by AWS DMS, see [Migrating MySQL-Compatible Databases to AWS](#). We recommend that you use the latest version of AWS DMS for the most comprehensive version and feature support.
- [mysqldbcopy](#) is a MySQL utility that copies a MySQL database on a single server or between servers.
- [mysqldump](#) is a MySQL utility that creates a dump file from a MySQL database for backup or migration purposes.

Epics

Plan the migration

Task	Description	Skills required
Validate the version and engine.	Validate the database version and engine for the source and target databases.	DBA
Identify hardware requireme nts.	Identify hardware requireme nts for the target server instance.	DBA, Systems administrator
Identify storage requireme nts.	Identify storage requirements (storage type and capacity).	DBA, Systems administrator
Choose the instance type.	Choose the proper instance type based on your compute, storage, and network requirements.	DBA, Systems administrator
Determine network access security requirements.	Identify the network access security requirements for the source and target databases.	DBA, Systems administrator
Determine strategy.	Identify the application migration strategy.	DBA, App owner, Systems administrator

Configure the infrastructure

Task	Description	Skills required
Create a virtual private cloud (VPC).	For instructions, see Create a VPC in the Amazon Virtual Private Cloud (Amazon VPC) documentation.	Systems administrator

Task	Description	Skills required
Create security groups.	For instructions, see Create a security group for your VPC in the Amazon VPC documenta tion.	Systems administrator
Configure and start an Aurora MySQL-Compatible DB cluster in your AWS account.	For instructions, see Creating an Amazon Aurora DB cluster in the Aurora documentation.	Systems administrator

Migrate data - option 1

Task	Description	Skills required
Use native MySQL tools or third-party tools to migrate database objects and data.	For instructions, see the documentation for MySQL tools such as mysqldbcopy and mysqldump.	DBA

Migrate data - option 2

Task	Description	Skills required
Migrate data with AWS DMS.	For instructions, see Using a MySQL-compatible database as a source and Using a MySQL-compatible database as a target in the AWS DMS documentation.	DBA

Migrate the application

Task	Description	Skills required
Follow the strategy.	Follow the application migration strategy.	DBA, App owner, Systems administrator

Cut over

Task	Description	Skills required
Switch application clients.	Switch the application clients over to connect to the new Aurora cluster endpoint.	DBA, App owner, Systems administrator

Close the project

Task	Description	Skills required
Shut down resources.	Shut down temporary AWS resources.	DBA, Systems administrator
Review documentation.	Review and validate the project documents.	DBA, App owner, Systems administrator
Collect metrics.	Gather metrics around time to migrate, percentage of manual steps versus tool usage, cost savings, and so on.	DBA, App owner, Systems administrator
Complete migration project.	Close out the project and provide feedback.	App owner, DBA, Systems administrator

Related resources

References

- Migrating data to Amazon Aurora MySQL DB cluster
- AWS DMS website
- AWS DMS documentation
- Amazon Aurora Pricing
- Creating and connecting to an Aurora MySQL DB cluster
- Amazon VPC and Amazon RDS
- Amazon Aurora documentation

Tutorials and videos

- Getting Started with AWS DMS
- Getting Started with Amazon Aurora

Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3

Rohan Jamadagni, Udayasimha Theepireddy, and sajith menon, Amazon Web Services

Summary

This pattern describes how to migrate large, on-premises MySQL databases efficiently to Amazon Aurora MySQL by using Percona XtraBackup. Percona XtraBackup is an open-source, non-blocking backup utility for MySQL-based servers. The pattern shows how to use Amazon Elastic File System (Amazon EFS) to reduce the time to upload the backup to Amazon Simple Storage Service (Amazon S3) and to restore the backup to Amazon Aurora MySQL. The pattern also provides details on how to make incremental Percona backups to minimize the number of binary logs to be applied to the target Aurora MySQL database.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Permissions to create AWS Identity and Access Management (IAM) roles and policies
- Network connectivity between the on-premises MySQL database and the virtual private cloud (VPC) on AWS

Limitations

- The source servers must be Linux-based systems that can install a Network File System (NFS) client (nfs-utils/nfs-common).
- The S3 bucket used for uploading backup files supports server-side encryption (SSE-S3/SSE-KMS) only.
- Amazon S3 limits the size of the backup files to 5 TB. If your backup file exceeds 5 TB, you can split it into multiple, smaller files.
- The number of source files uploaded to the S3 bucket cannot exceed one million files.
- The pattern supports Percona XtraBackup full backup and incremental backup only. It doesn't support partial backups that use `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude`, or `--databases-file`.

- Aurora doesn't restore users, functions, stored procedures, or time zone information from the source MySQL database.

Product versions

- The source database must be MySQL version 5.5, 5.6, or 5.7.
- For MySQL 5.7, you must use Percona XtraBackup 2.4.
- For MySQL 5.6 and 5.6, you must use Percona XtraBackup 2.3 or 2.4.

Architecture

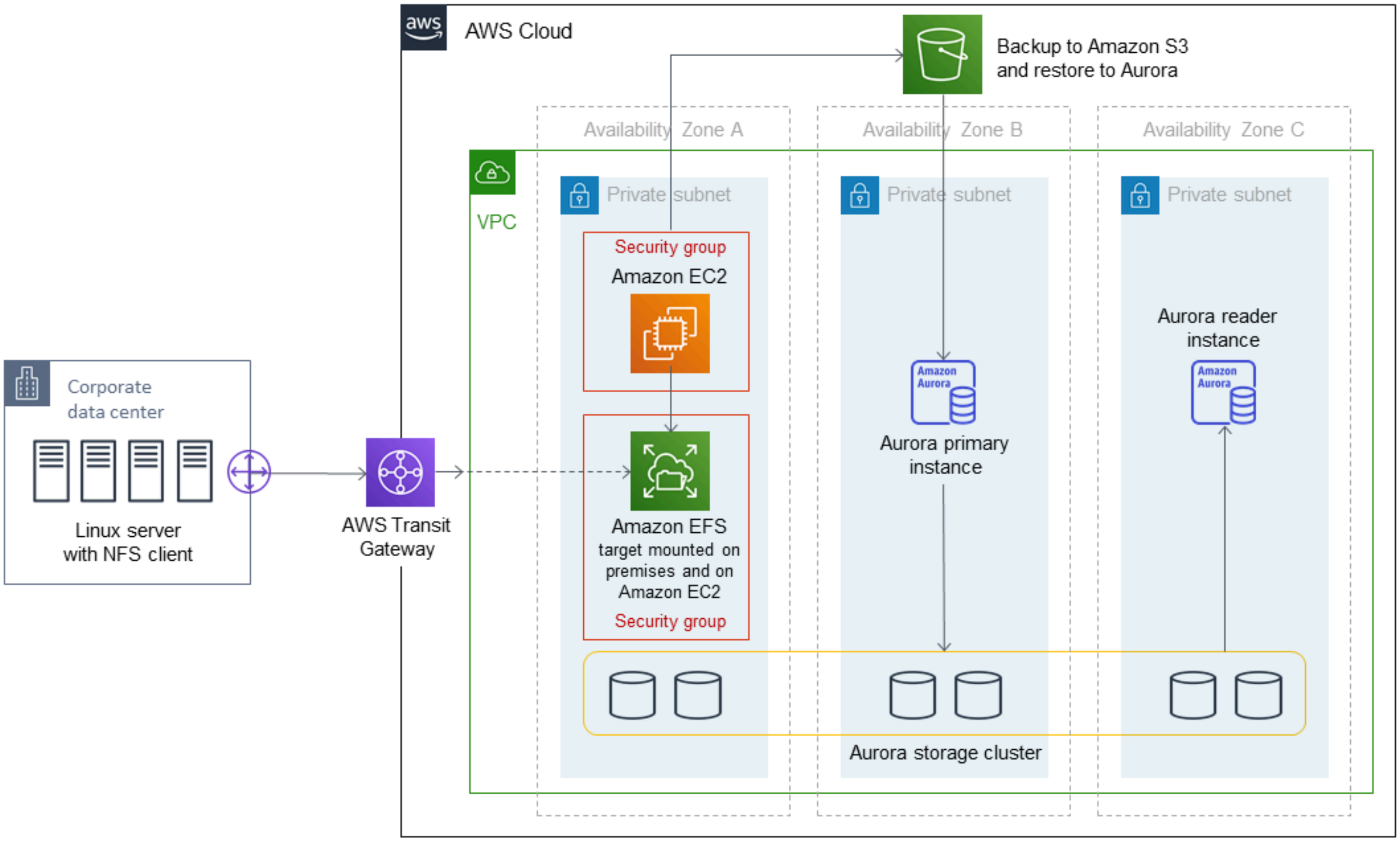
Source technology stack

- Linux-based operating system
- MySQL server
- Percona XtraBackup

Target technology stack

- Amazon Aurora
- Amazon S3
- Amazon EFS

Target architecture



Tools

AWS services

- [Amazon Aurora](#) is a fully managed relational database engine that makes it simple and cost-effective to set up, operate, and scale MySQL deployments. Aurora MySQL is a drop-in replacement for MySQL.
- [Amazon Elastic File System (Amazon EFS)](#) helps you create and configure shared file systems in the AWS Cloud.
- [Amazon Simple Storage Service (Amazon S3)](#) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- [Percona XtraBackup](#) is an open-source utility that performs streaming, compressed, and incremental backups of MySQL databases without disrupting or blocking your databases.

Epics

Create an Amazon EFS file system

Task	Description	Skills required
Create a security group to associate with Amazon EFS mount targets.	Create a security group in the VPC that is configured with a VPN attachment to the on-premises database over AWS Transit Gateway. For more information about the commands and steps described in this and other stories, see the links in the "Related resources" section at the end of this pattern.	AWS DevOps/database administrator
Edit the security group rules.	Add an inbound rule, using type NFS, port 2049, and the IP range of the on-premises database server as the source. By default, the outbound rule allows all the traffic to leave. If this is not the case , add an outbound rule to open a connection for the NFS port. Add two more inbound rules: port 2049 (source: security group ID of this same security group) and port 22 (source: IP range from where you will connect to an EC2 instance).	AWS DevOps/database administrator
Create a file system.	In the mount targets, use the VPC and security group you created in the previous story. Choose the throughput	AWS DevOps/database administrator

Task	Description	Skills required
	mode and performance based on the I/O requirements of the on-premises database. Optionally, enable encryption at rest.	

Mount the file system

Task	Description	Skills required
Create an IAM instance profile role to be associated with an EC2 instance.	Create an IAM role that has permissions to upload and access objects in Amazon S3. Choose the S3 bucket where the backup will be stored as a policy resource.	AWS DevOps
Create an EC2 instance.	Launch an Linux-based EC2 instance and attach the IAM instance profile role that you created in the previous step, and the security group you created earlier.	AWS DevOps
Install the NFS client.	Install the NFS client on the on-premises database server and on the EC2 instance. For installation instructions, see the "Additional information" section.	DevOps
Mount the Amazon EFS file system.	Mount the Amazon EFS file system on premises and on the EC2 instance. On each server, create a directory	DevOps

Task	Description	Skills required
	for storing the backup, and mount the file system by using the mount target endpoint. For an example, see the "Additional information" section.	

Make a backup of the MySQL source database

Task	Description	Skills required
Install Percona XtraBackup.	Install Percona XtraBacku p 2.3 or 2.4 (depending on the version of your MySQL database) on the on-premises database server. For installat ion links, see the "Related resources" section.	Database administrator
Count the schemas and tables in the source database.	Gather and note the number of schemas and objects in the source MySQL database. You will use these counts to validate the Aurora MySQL database after migration.	Database administrator
(Optional) Note the latest binary log sequence from the source database.	Perform this step if you want to establish binary log replication between the source database and Aurora MySQL to minimize downtime. log-bin must be enabled, and server_id must be unique. Note the current binary log sequence from the	Database administrator

Task	Description	Skills required
	source database, just before initiating a backup. Perform this step just before full backup if you're planning to use only full backup. If you're planning to make increment al backups after a full backup, perform this step just before the final incremental backup that you will restore on the Aurora MySQL DB instance.	
Start a full backup of the source MySQL database.	Make a full backup of the MySQL source database using Percona XtraBackup. For example commands for full and incremental backups, see the "Additional information" section.	Database administrator

Task	Description	Skills required
(Optional) Make increment al backups using Percona XtraBackup.	Incremental backups can be used to reduce the amount of binary logs you need to apply to sync the source database with Aurora MySQL. Large-size and transaction-heavy databases might generate a large number of binary logs during backups. By taking incremental backups and storing them on a shared Amazon EFS file system, you can significantly reduce the time for backing up and uploading your database. For details, see the "Addition al information" section. Continue to make incremental backups until you're ready to begin the migration process to Aurora.	Database administrator

Task	Description	Skills required
Prepare backups.	In this step, transactional logs are applied to the backup for transactions that were in flight during the backup. Continue to apply transactional logs (--apply-log-only) to each incremental backup to merge the backups, except for the last backup. For examples, see the "Additional information" section. After this step, the complete, merged backup will be in ~/<efs_mount_name>/fullbackup.	Database administrator
Zip and split the final merged backup.	After you prepare the final, merged backup, use tar, zip, and split commands to create smaller zipped files of the backup. For examples, see the "Additional information" section.	Database administrator

Restore the backup to an Aurora MySQL DB cluster

Task	Description	Skills required
Upload the backup to Amazon S3.	The Amazon EFS file system where the backup files are stored is mounted on both the on-premises database and an EC2 instance, so the backup files are readily	AWS DevOps

Task	Description	Skills required
	available to the EC2 instance. Connect to the EC2 instance by using Secure Shell (SSH) and upload the zipped backup files to a new or existing S3 bucket; for example: aws s3 sync ~/ <efs_mount_name>/fullback up s3://<bucket_name>/ fullbackup. For additional details, see the links in the "Related resources" section.	
Create a service role for Aurora to access Amazon S3.	Create an IAM role with trust "rds.amazonaws.com" and a policy that will enable Aurora to access the S3 bucket where the backup files are stored. The required permissions are ListBucket, GetObject, and GetObjectVersion.	AWS DevOps

Task	Description	Skills required
Create the networking configuration for Aurora.	Create a cluster DB subnet group with at least two Availability Zones and a subnet route table configura tion that allows outbound connectivity to the source database. Create a security group that allows outbound connections to the on-premis es database, and allows administrators to connect to the Aurora DB cluster. For more information, see the links in the "Related resources " section.	AWS DevOps/database administrator
Restore the backup to an Aurora MySQL DB cluster.	Restore your data from the backup that you uploaded to Amazon S3. Specify the MySQL version of your source database, provide the S3 bucket name and folder path prefix where you uploaded the backup file (for example, "fullbackup" for the examples in the "Additional informati on" section), and provide the IAM role you created to authorize Aurora to access Amazon S3.	AWS DevOps/database administrator

Task	Description	Skills required
Validate the Aurora MySQL database.	Validate the count of schema and objects in the restored Aurora DB cluster against the count you obtained from the source database.	Database administrator
Set up binlog replication.	Use the binary log sequence that you noted earlier, before making the last backup that was restored to the Aurora DB cluster. Create a replication user on the source database, and follow the instructions in the "Additional informati on" section to provide the appropriate privileges, to enable replication on Aurora, and to confirm that the replication is in sync.	AWS DevOps/database administrator

Related resources

Creating an Amazon EFS file system

- Creating a security group (Amazon VPC documentation)
- Transit gateway VPN attachments (Amazon VPC documentation)
- Scaling VPN throughput using AWS Transit Gateway (Networking & Content Delivery blog)
- Creating an Amazon EFS file system (Amazon EFS documentation)
- Creating mount targets (Amazon EFS documentation)
- Encrypting data at rest (Amazon EFS documentation)

Mounting the file system

- IAM roles for Amazon EC2 (Amazon EC2 documentation)

- Launching an Amazon EC2 Linux instance (Amazon EC2 documentation)
- Installing the NFS client (Amazon EFS documentation)
- Mounting the Amazon EFS file system on your on-premises client (Amazon EFS documentation)
- Mounting EFS File Systems (Amazon EFS documentation)

Making a backup of the MySQL source database

- Installing Percona XtraBackup 2.3 (Percona XtraBackup documentation)
- Installing Percona XtraBackup 2.4 (Percona XtraBackup documentation)
- Setting the replication master configuration (MySQL documentation)
- Migrating data from an external MySQL database to an Aurora MySQL DB cluster (Aurora documentation)
- Incremental backup (Percona XtraBackup documentation)

Restoring the backup to Amazon Aurora MySQL

- Creating a bucket (Amazon S3 documentation)
- Connecting to your Linux instance using SSH (Amazon Ec2 documentation)
- Configuring the AWS CLI (AWS CLI documentation)
- sync command (AWS CLI command reference)
- Creating an IAM policy to access Amazon S3 resources (Aurora documentation)
- DB cluster prerequisites (Aurora documentation)
- Working with DB subnet groups (Aurora documentation)
- Creating a VPC security group for a private DB instance (Aurora documentation)
- Restoring an Aurora MySQL DB cluster from an S3 bucket (Aurora documentation)
- Setting up replication with MySQL or another Aurora DB cluster (Aurora documentation)
- mysql.rds_set_external_master procedure (MySQL on Amazon RDS SQL reference)
- mysql.rds_start_replication procedure (MySQL on Amazon RDS SQL reference)

Additional references

- Migrating data from an external MySQL database to an Aurora MySQL DB cluster (Aurora documentation)

- MySQL server downloads (Oracle website)

Tutorials and videos

- Migrating MySQL data to an Aurora MySQL DB cluster using Amazon S3 (AWS Knowledge Center)
- Amazon EFS setup and mount (video)

Additional information

Installing an NFS client

- If you are using Red Hat or a similar Linux operating system, use the command:

```
$ sudo yum -y install nfs-utils
```

- If you are using Ubuntu or a similar Linux operating system, use the command:

```
$ sudo apt-get -y install nfs-common
```

For more information, see the walkthrough in the Amazon EFS documentation.

Mounting the Amazon EFS file system

Use the commands:

```
mkdir ~/<efs_mount_name>
$ sudo mount -t nfs -o
 nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport mount-target-IP:/ ~/<efs_mount_name>
```

For more information, see the walkthrough and Mounting EFS File Systems in the Amazon EFS documentation.

Making backups of the MySQL source database

Full backups

Use a command like the following, which takes the backup, zips it, and splits it into smaller chunks of 1 GB each:

```
xtrabackup --backup --user=dbuser --password=<password> --binlog-info=AUTO --stream=tar
 --target-dir=~/<efs_mount_name>/fullbackup | gzip - | split -d --bytes=1024MB - ~/
<efs_mount_name>/fullbackup/backup.tar.gz &
```

If you're planning to make subsequent incremental backups after the full backup, do not zip and split the backup. Instead, use a command similar to the following:

```
xtrabackup --backup --user=dbuser --password=<password> --target-dir=~/
<efs_mount_name>/fullbackup/
```

Incremental backups

Use the full backup path for the `--incremental-basedir` parameter; for example:

```
xtrabackup --backup --user=dbuser --password=<password> --target-dir=~/
<efs_mount_name>/incremental/backupdate --incremental-basedir=~/<efs_mount_name>/
fullbackup
```

where *basedir* is the path to the full backup and the xtrabackup_checkpoints file.

For more information about making backups, see Migrating Data from an External MySQL Database to an Amazon Aurora MySQL DB Cluster in the Aurora documentation.

Preparing backups

To prepare a full backup:

```
xtrabackup --prepare --apply-log-only --target-dir=~/<efs_mount_name>/fullbackup
```

To prepare an incremental backup:

```
xtrabackup --prepare --apply-log-only --target-dir=~/<efs_mount_name>/fullbackup --
incremental-dir=~/<efs_mount_name>/incremental/06062020
```

To prepare the final backup:

```
xtrabackup --prepare --target-dir=~/<efs_mount_name>/fullbackup --incremental-dir=~/
<efs_mount_name>/incremental/06072020
```

For more information, see Incremental backups in the Percona XtraBackup documentation.

Zipping and splitting the merged backup

To zip the merged backup at ~/<efs_mount_name>/fullbackup:

```
tar -zcvf <backupfilename.tar.gz> ~/<efs_mount_name>/fullbackup
```

To split the backup:

```
split -d -b1024M --verbose  <backupfilename.tar.gz> <backupfilename.tar.gz>
```

Setting up binlog replication

To create a replication user on the source database and provide the appropriate privileges:

```
CREATE USER 'repl_user'@'' IDENTIFIED BY ''; GRANT REPLICATION CLIENT, REPLICATION
 SLAVE ON *.* TO 'repl_user'@'';
```

To enable replication on Aurora by connecting to the Aurora DB cluster, enable binary logs in the DB cluster parameter group. Set `binlog_format = mixed` (mixed mode is preferred). This change requires that you restart the instance to apply the update.

```
CALL mysql.rds_set_external_master ('sourcedbinstanceIP', sourcedbport, 'repl_user',
 '', 'binlog_file_name', binlog_file_position, 0); CALL mysql.rds_start_replication;
```

To confirm that the replication is in sync:

```
SHOW Slave Status \G;
```

The **Seconds behind master** field shows how far behind Aurora is from the on-premises database.

Migrate on-premises Java applications to AWS using AWS App2Container

Dhananjay Karanjkar, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

AWS App2Container (A2C) is a command line tool that helps transform existing applications running in virtual machines into containers, without needing any code changes. A2C discovers applications running on a server, identifies dependencies, and generates relevant artifacts for seamless deployment to Amazon Elastic Container Service (Amazon ECS) and Amazon Elastic Kubernetes Service (Amazon EKS).

This pattern provides the steps for remotely migrating on-premises Java applications deployed on an application server to AWS Fargate or Amazon EKS by using App2Container through the worker machine.

The worker machine can be used in the following use cases:

- Docker installation is not allowed or not available on the application servers where the Java applications are running.
- You must manage the migration of multiple applications deployed on different physical or virtual servers.

This pattern uses AWS CodeCommit, AWS CodePipeline, and AWS CodeBuild.

Prerequisites and limitations

Prerequisites

- An application server with a Java application running on a Linux server
- A worker machine with a Linux operating system
- A worker machine with at least 20 GB of available disk space

Limitations

- Not all applications are supported. For more information, see Supported applications for Linux.

Architecture

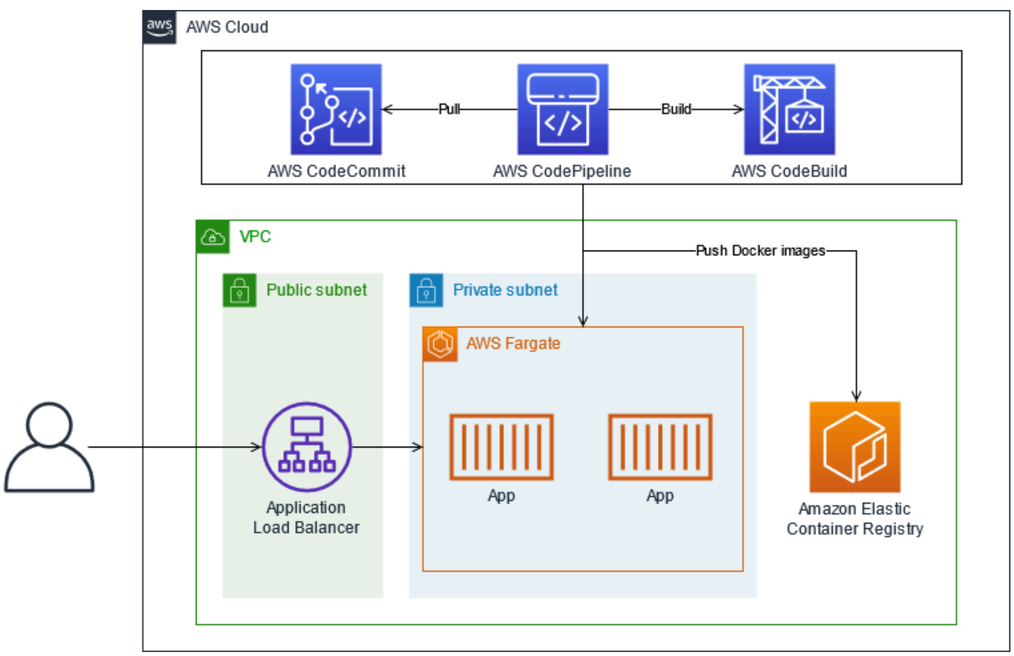
Source technology stack

- Java applications running on Linux server

Target technology stack

- AWS CodeBuild
- AWS CodeCommit
- AWS CodeDeploy
- AWS CodePipeline
- Amazon Elastic Container Registry
- AWS Fargate

Target architecture



Tools

Tools

- AWS App2Container – AWS App2Container (A2C) is a command line tool to help you lift and shift applications that run in your on-premises data centers or on virtual machines, so that they run in containers that are managed by Amazon ECS or Amazon EKS.
- AWS CodeBuild – AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy.
- AWS CodeCommit – AWS CodeCommit is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as documents, source code, and binary files) in the cloud.
- AWS CodePipeline – AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that for running, stopping, and managing containers on a cluster.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is an AWS managed container image registry service that is secure, scalable, and reliable.
- Amazon EKS – Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.
- AWS Fargate – AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon Elastic Compute Cloud (Amazon EC2) instances. With Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers.

Epics

Set up credentials

Task	Description	Skills required
Create a secret to access the application server.	To access the application server remotely from the worker machine, create a secret in AWS Secrets Manager. For your secret, you can use either the SSH private key or the Certificate and the	DevOps, Developer

Task	Description	Skills required
	SSH private key. For more information, see [Manage secrets for AWS App2Container]().	

Set up the worker machine

Task	Description	Skills required
Install the tar file.	Run `sudo yum install -y tar`.	DevOps, Developer
Install the AWS CLI.	To install the Amazon Command Line Interface (AWS CLI), run `curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"` . Unzip `awscliv2.zip` . Run `sudo ./aws/install` .	DevOps, Developer
Install App2Container.	Run the following commands: `curl -o AWSApp2Container-installer-linux.tar.gz https://app2container-release-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/AWSApp2Con`	DevOps, Developer

Task	Description	Skills required

	`tainer-installer-linux.tar.gz`
	
	`sudo tar xvf AWSApp2Container-installer-linux.tar.gz`
	
	`sudo ./install.sh`	

Configure the profiles.	To configure the AWS default profile, run `sudo aws configure` . To configure the named AWS default profile, run `sudo aws configure --profile <profile name>`.	DevOps, Developer

Install Docker.	Run the following commands.
	
	`sudo yum install -y docker`
	
	`sudo systemctl enable docker & sudo systemctl restart docker`	

Task	Description	Skills required
Initialize App2Container.	To initialize App2Container, you need the following information: • `workspace` : To store application containerization artifacts. We recommend providing a directory path that has at least 20 GB of free disk space. • `awsProfile` : AWS profile configured on the server. This is required to upload artifacts to Amazon S3, run the `containerize` command, and generate AWS artifacts for deployment on Amazon ECS or Amazon EKS. • `s3Bucket`: To extract and store AWS artifacts. • `metricsReportPermission` : To collect and store metrics reported. • `dockerContentTrust` : To sign the Docker image. Run `sudo app2container init`.	DevOps, Developer

Configure the worker machine

Task	Description	Skills required
Configure the worker machine to remotely connect and run App2Container commands on the application server.	To configure the worker machine, the following information is required: • `Server FQDN`: The fully qualified domain name of the application server. • `Server IP address`: The IP address of the applicati on server. Either the FQDN or the IP address is sufficien t. • `SecretARN` : The Amazon Resource Name (ARN) of the secret that is used to connect to the applicati on server and is stored in Secrets Manager. • `AuthMethod` : The `key` or `cert` authentication method. Run `sudo app2container remote configure` .	DevOps, Developer

Discover, analyze, and extract applications on the worker machine

Task	Description	Skills required
Discover the on-premises Java applications.	To remotely discover all the applications running on the	Developer, DevOps

Task	Description	Skills required
	application server, run the following command. `sudo app2container remote inventory --target <FQDN/IP of App server>` This command generates a list of deployed applications in `inventory.json` .	
Analyze the discovered applications.	To remotely analyze each application by using the application-id obtained in the inventory stage, run the following command. `sudo app2container remote analyze --application-id <java-app-id> --target <FQDN/IP of App Server>` This generates `analysis.json` file in the workspace location. After this file is generated, you can alter the containerization parameters based on your needs.	Developer, DevOps

Task	Description	Skills required
Extract the analyzed applicati ons.	To generate an applicati on archive for the analyzed application, remotely run the following command, which will generate the tar bundle in the workspace location. `sudo app2container remote extract --application-id <application id> --target <FQDN/IP of App Server>` Extracted artifacts can be generated on the local worker machine.	Developer, DevOps

Containerize the extracted artifacts on the worker machine

Task	Description	Skills required
Containerize the extracted artifacts.	Containerize the artifacts extracted in the previous step by running the following command. `sudo app2container containerize --input-archive <tar bundle location on worker machine>`	Developer, DevOps
Finalize the target.	To finalize the target, open `deployment.json` ,	Developer, DevOps

Task	Description	Skills required
	which is created when the `containerize` command runs. To specify AWS Fargate as the target, set `createEcsArtifacts` to `true`. To set Amazon EKS as the target, set `createEksArtifacts` to true.	

Generate and provision AWS artifacts

Task	Description	Skills required
Generate AWS deployment artifacts on the worker machine.	To generate deployment artifacts, run the following command. `sudo app2container generate app-deployment --application-id <application id>` This generates the `ecs-master.yml` AWS CloudFormation template in the workspace.	DevOps
Provision the artifacts.	To further provision the generated artifacts, deploy the AWS CloudFormation template by running the following command. `aws cloudformation deploy --template-`	DevOps

Task	Description	Skills required
	`file <path to ecs-master.yml> --capabilities CAPABILITY_NAMED_IAM --stack-name <application id>-ECS`	
Generate the pipeline.	Modify `pipeline.json` , which was created in the previous story, based on your needs. Then run the `generate pipeline` command to generate the pipeline deployment artifacts.	DevOps

Related resources

- What is App2Container?
- AWS App2Container blog post
- AWS CLI configuration basics
- Docker basics for Amazon ECS
- Docker commands

Migrate shared file systems in an AWS large migration

Amit Rudraraju, Sam Apa, Bheemeswararao Balla, Wally Lu, and Sanjeev Prakasam, Amazon Web Services

Summary

Migrating 300 or more servers is considered a *large migration*. The purpose of a large migration is to migrate workloads from their existing, on-premises data centers to the AWS Cloud, and these projects typically focus on application and database workloads. However, shared file systems require focused attention and a separate migration plan. This pattern describes the migration process for shared file systems and provides best practices for migrating them successfully as part of a large migration project.

A *shared file system* (SFS), also known as a *network* or *clustered* file system, is a file share that is mounted to multiple servers. Shared file systems are accessed through protocols such as Network File System (NFS), Common Internet File System (CIFS), or Server Message Block (SMB).

These systems are not migrated with standard migration tools such as AWS Application Migration Service because they are neither dedicated to the host being migrated nor represented as a block device. Although most host dependencies are migrated transparently, the coordination and management of the dependent file systems must be handled separately.

You migrate shared file systems in the following phases: discover, plan, prepare, cut over, and validate. Using this pattern and the attached workbooks, you migrate your shared file system to an AWS storage service, such as Amazon Elastic File System (Amazon EFS), Amazon FSx for NetApp ONTAP, or Amazon FSx for Windows File Server. To transfer the file system, you can use AWS DataSync or a third-party tool, such as NetApp SnapMirror.

> Note: This pattern is part of an AWS Prescriptive Guidance series about large migrations to the AWS Cloud. This pattern includes best practices and instructions for incorporating SFSs into your wave plans for servers. If you are migrating one or more shared file systems outside of a large migration project, see the data transfer instructions in the AWS documentation for Amazon EFS, Amazon FSx for Windows File Server, and Amazon FSx for NetApp ONTAP.

Prerequisites and limitations

Prerequisites

Prerequisites can vary depending on your source and target shared file systems and your use case. The following are the most common:

- An active AWS account.
- You have completed application portfolio discovery for your large migration project and started developing wave plans. For more information, see Portfolio playbook for AWS large migrations.
- Virtual private clouds (VPCs) and security groups that allow ingress and egress traffic between the on-premises data center and your AWS environment. For more information, see Network-to-Amazon VPC connectivity options and AWS DataSync network requirements.
- Permissions to create AWS CloudFormation stacks or permissions to create Amazon EFS or Amazon FSx resources. For more information, see the CloudFormation documentation, Amazon EFS documentation, or Amazon FSx documentation.
- If you're using AWS DataSync to perform the migration, you need the following permissions:
  - Permissions for AWS DataSync to send logs to an AWS CloudWatch Logs log group. For more information, see Allowing DataSync to upload logs to CloudWatch log groups.
  - Permissions to access the CloudWatch Logs log group. For more information, see Overview of managing access permissions to your CloudWatch Logs resources.
  - Permissions to create agents and tasks in DataSync. For more information, see Required IAM permissions for using AWS DataSync.

Limitations

- This pattern is designed to migrate SFSs as part of a large migration project. It includes best practices and instructions for incorporating SFSs into your wave plans for migrating applications. If you are migrating one or more shared file systems outside of a large migration project, see the data transfer instructions in the AWS documentation for Amazon EFS, Amazon FSx for Windows File Server, and Amazon FSx for NetApp ONTAP.
- This pattern is based on commonly used architectures, services, and migration patterns. However, large migration projects and strategies can vary between organizations. You might need to customize this solution or the provided workbooks based on your requirements.

Architecture

Source technology stack

One or more of the following:

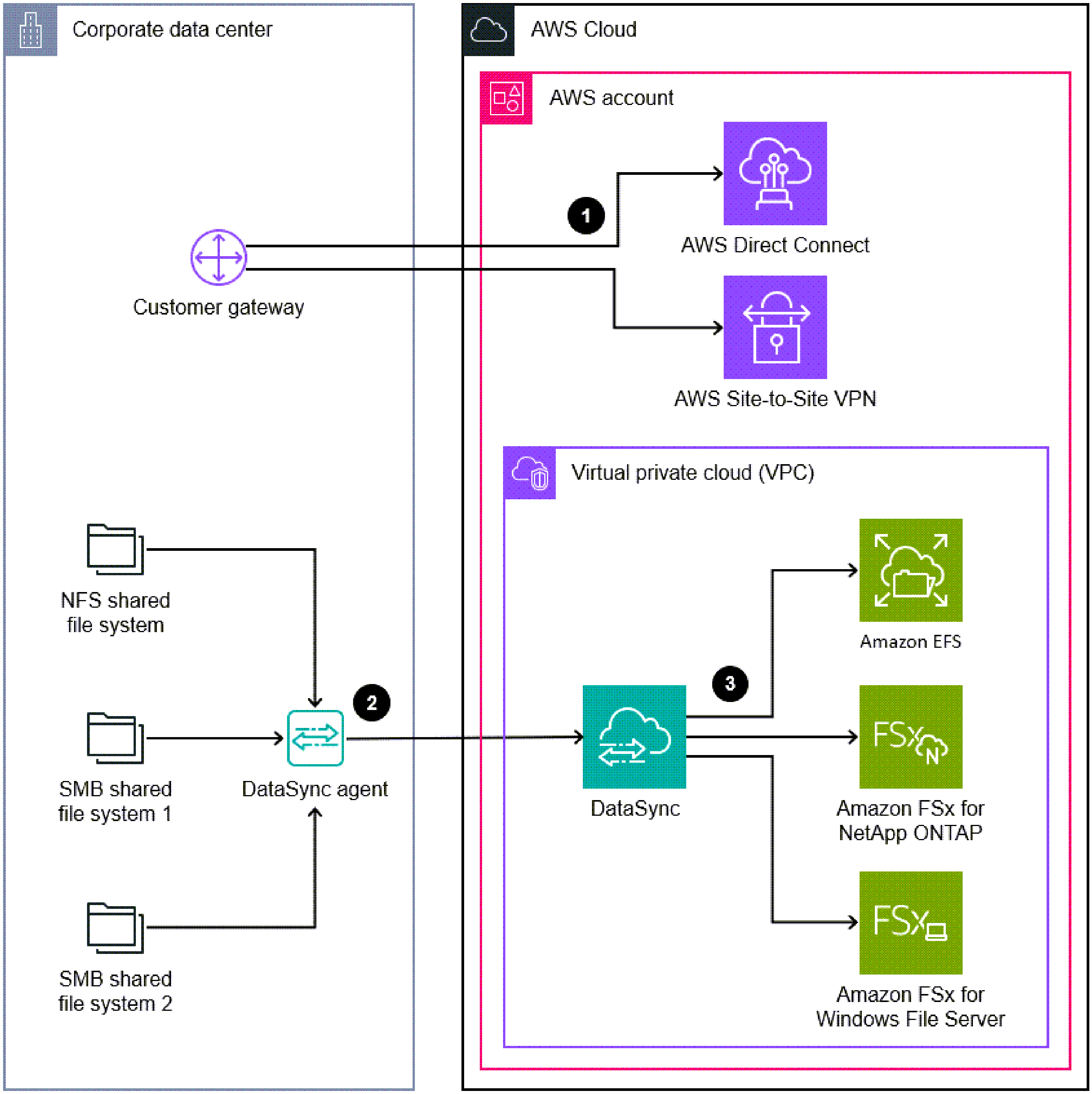
- Linux (NFS) file server
- Windows (SMB) file server
- NetApp storage array
- Dell EMC Isilon storage array

Target technology stack

One or more of the following:

- Amazon Elastic File System
- Amazon FSx for NetApp ONTAP
- Amazon FSx for Windows File Server

Target architecture



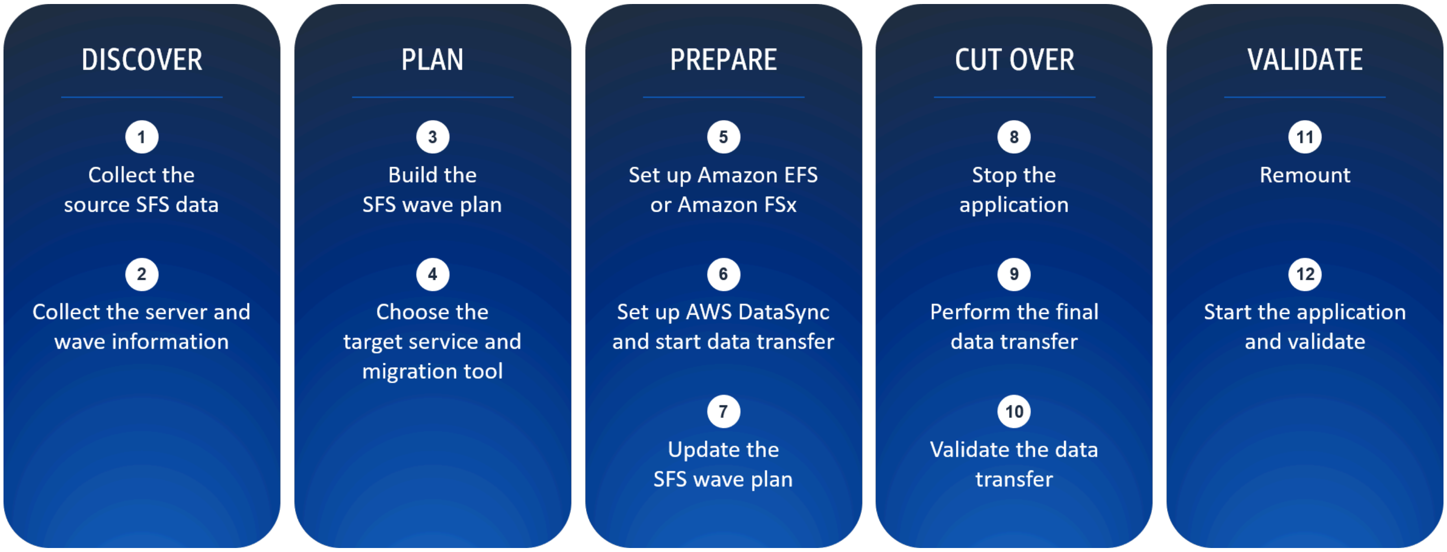
The diagram shows the following process:

1. You establish a connection between the on-premises data center and the AWS Cloud by using an AWS service such as AWS Direct Connect or AWS Site-to-Site VPN.
2. You install the DataSync agent in the on-premises data center.

3. According to your wave plan, you use DataSync to replicate data from the source shared file system to the target AWS file share.

Migration phases

The following image shows the phases and high-level steps for migrating an SFS in a large migration project.



The Epics section of this pattern contains detailed instructions for how to complete the migration and use the attached workbooks. The following is a high-level overview of the steps in this phased approach.

Phase	Steps
Discover	1. Using a discovery tool, you collect data about the shared file system, including servers, mount points, and IP addresses. 2. Using a configuration management database (CMDB) or your migration tool, you collect details about the server, including information about the migration wave, environment, application owner, IT service management (ITSM) service name, organizat ional unit, and application ID.

Plan	3. Using the collected information about the SFSs and the servers, create the SFS wave plan. 4. Using the information in the build worksheet, for each SFS, choose a target AWS service and a migration tool.
Prepare	5. Set up the target infrastructure in Amazon EFS, Amazon FSx for NetApp ONTAP, or Amazon FSx for Windows File Server. 6. Set up the data transfer service, such as DataSync, and then start the initial data sync. When the initial sync is complete, you can set up reoccurring syncs to run on a schedule, 7. Update the SFS wave plan with informati on about the target file share, such as the IP address or path.
Cut over	8. Stop applications that actively access the source SFS. 9. In the data transfer service, perform a final data sync. 10. When the sync is complete, validate that it was completely successfully by reviewing the log data in CloudWatch Logs.
Validate	11. On the servers, change the mount point to the new SFS path. 12. Restart and validate the applications.

Tools

AWS services

- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- AWS DataSync is an online data transfer and discovery service that helps you move files or object data to, from, and between AWS storage services.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.
- Amazon FSx provides file systems that support industry-standard connectivity protocols and offer high availability and replication across AWS Regions.

Other tools

- SnapMirror is a NetApp data replication tool that replicates data from specified source volumes or qtrees to target volumes or qtrees, respectively. You can use this tool to migrate a NetApp source file system to Amazon FSx for ONTAP.
- Robocopy, which is short for *Robust File Copy*, is a command-line directory and command for Windows. You can use this tool to migrate a Windows source file system to Amazon FSx for Windows File Server.

Best practices

Wave planning approaches

When planning waves for your large migration project, consider latency and application performance. When the SFS and dependent applications are operating in different locations, such as one in the cloud and one in the on-premises data center, this can increase latency and affect application performance. The following are the available options when creating wave plans:

1. **Migrate the SFS and all dependent servers within the same wave –** This approach prevents performance issues and minimizes rework, such as reconfiguring mount points multiple times. It is recommended when very low latency is required between the application and the SFS. However, wave planning is complex, and the goal is typically to remove variables from dependency groupings, not add to them. In addition, this approach isn't recommended if many servers access the same SFS because it makes the wave too large.
2. **Migrate the SFS after the last dependent server has been migrated –** For example, if an SFS is accessed by multiple servers and those servers are scheduled to migrate in waves 4, 6, and 7, schedule the SFS to migrate in wave 7.

This approach is often the most logical for large migrations and is recommended for latency-sensitive applications. It reduces costs associated with data transfer. It also minimizes the period of latency between the SFS and higher-tier (such as production) applications because higher-tier applications are typically scheduled to migrate last, after development and QA applications.

However, this approach still requires discovery, planning, and agility. You might need to migrate the SFS in an earlier wave. Confirm that the applications can withstand the additional latency for the period of time between the first dependent wave and the wave containing the SFS. Conduct a discovery session with the application owners and migrate the application in same wave the most latency-sensitive application. If performance issues are discovered after migrating a dependent application, be prepared to pivot quickly to migrate the SFS as quickly as possible.

3. **Migrate the SFS at the end of the large migration project –** This approach is recommended if latency is not a factor, such as when the data in the SFS is infrequently accessed or not critical to application performance. This approach streamlines the migration and simplifies cutover tasks.

You can blend these approaches based on the latency-sensitivity of the application. For example, you can migrate latency-sensitive SFSs by using approaches 1 or 2 and then migrate the rest of the SFSs by using approach 3.

Choosing an AWS file system service

AWS offers several cloud services for file storage. Each offers different benefits and limitations for performance, scale, accessibility, integration, compliance, and cost optimization. There are some logical default options. For example, if your current on-premises file system is operating Windows Server, then Amazon FSx for Windows File Server is the default choice. Or if the on-premises file system is operating NetApp ONTAP, then Amazon FSx for NetApp ONTAP is the default choice. However, you might choose a target service based on the requirements of your application or to realize other cloud operating benefits. For more information, see Choosing the right AWS file storage service for your deployment (AWS Summit presentation).

Choosing a migration tool

Amazon EFS and Amazon FSx support use of AWS DataSync to migrate shared file systems to the AWS Cloud. For more information about supported storage systems and services, benefits, and use cases, see What is AWS DataSync. For an overview of the process of using DataSync to transfer your files, see How AWS DataSync transfers work.

There are also several third-party tools that are available, including the following:

- If you choose Amazon FSx for NetApp ONTAP, you can use NetApp SnapMirror to migrate the files from the on-premises data center to the cloud. SnapMirror uses block-level replication, which can be faster than DataSync and reduce the duration of the data transfer process. For more information, see Migrating to FSx for ONTAP using NetApp SnapMirror.
- If you choose Amazon FSx for Windows File Server, you can use Robocopy to migrate files to the cloud. For more information, see Migrating existing files to FSx for Windows File Server using Robocopy.

Epics

Discover

Task	Description	Skills required
Prepare the SFS discovery workbook.	1. Download the workbooks in the Attachments section of this pattern. This contains two files, **SFS-Discovery-Workbook.xlsx** and **SFS-Wave-Plan-Workbook.xlsx**. 2. Open the **SFS-Discovery-Workbook** file in Microsoft Excel. 3. On the **Dashboard** worksheet, do the following: • In column **A**, update the environment name. • In column **B**, update the order of the environments to put them in order of lowest (1) priority to highest priority. • In columns **D–E**, update the wave schedule.	Migration engineer, Migration lead

Task	Description	Skills required
	• In columns **C** and **K**, update the AWS account names. • In column **L**, update the VPC IDs. • In columns **M–O**, update the subnet IDs. 4. Review the rest of the workbook template and update any other values necessary for your organization or use case. 5. Save the workbook.	

Task	Description	Skills required
Collect information about the source SFS.	1. Using your preferred discovery tool, identify all of the SFS mounts across all of the applicable storage devices, Linux servers, and Windows servers. Typically, you need to collect the following information: • Client devices • Client IP address • SFS details • Mount point **Note** You can add mount point details to your migration runbook for remounting the SFS after the migration. 2. Open the **SFS-Discovery-Workbook** file. 3. On the **Wave-Sheet** worksheet, do the following: • In the **Server location** (D) column, in the formula, confirm that the format	Migration engineer, Migration lead

Task	Description	Skills required
	of the CIDR range for the on-premises source works for your range. For example, if your CIDR range is `10.0.0.0/8` , enter `10.*.*.*`. • In the **SFS location** (E) column, in the formula, confirm that the format of the CIDR range for the target VPC works for your range. For example, if your CIDR range is `176.16.0.0/16` , enter `176.16.*.*` . 4. On the **SFS-Data** worksheet, do the following: • In the **Server name** (A) column, enter the name of the server where the SFS is mounted. • In the **SFS path** (B) column, enter the name of the SFS. • In the **IP address** (C) column, enter the IP address of the server. • Add any other relevant information that you collected during discovery, such as the mount point and SFS size. You can use this	

Task	Description	Skills required
	data later to modify the wave planning calculati ons. 5. Save the workbook.	

Task	Description	Skills required
Collect information about the servers.	1. Using your CMDB or the data recorded in your migration tool, identify all of the following informati on about the servers that have SFS mounts: • Server name • IP address • Wave • Organizational unit (OU) • Server environment, such as `DEV`, `QA`, or `PROD` • Application name • Application owner and contact information 2. Open the **SFS-Discovery-Workbook** file. 3. On the **Server-Data** worksheet, in columns **A–H**, enter the information that you collected about the source servers. Note the following: • In the **Wave #** (C) column, enter the wave name (such as `Wave1`), out-of-scope (`OOS`), or `Retire`. • If the **App owner contact** (H) column, verify the email address is correct. This email	Migration engineer, Migration lead

Task	Description	Skills required
	address is automatically generated based on the name you provided in the **App owner** (G) column. If necessary, manually update the value to reflect the correct email address. • Don't modify columns **I–J**, which contain formulas. 4. Save the workbook.	

Plan

Task	Description	Skills required
Build the SFS wave plan.	1. Open the **SFS-Discovery-Workbook** file. 2. Verify all of the information collected in the discovery phase is accurate and current. 3. On the **Wave-Sheet** worksheet, filter the **SFS wave** (K) column on the value 1. This is a list of all SFSs in the first wave. **Note** A value of `0` in this column indicates that the SFS is out	Build lead, Cutover lead, Migration engineer, Migration lead

Task	Description	Skills required
	of scope of the migration. This might be because the SFS is already hosted on AWS or because the servers that access the share are out of scope of the migration. 4. Verify that you want to migrate these SFSs in this wave. For more information about how to assign SFSs to waves, see *Wave planning approaches* in the [Best Practices](#) section. 5. Select and copy the cells containing the filtered values. Do not copy the header row containing the column titles. 6. Open the **SFS-Wave-Plan-Workbook** file that you previously downloaded. 7. On the **Export-from-Discovery** worksheet, select cell **A2**. 8. Paste the copied data. 9. Save the **SFS-Discovery-Workbook** and **SFS-Wave-Plan-Workbook** files.	

Task	Description	Skills required
Choose the target AWS service and migration tool.	1. In the **SFS-Wave-Plan-Workbook** file, on the **Exported-from-Discovery** worksheet, select and copy the values in the **Old path** (C) column. 2. On the **Build-Wave** worksheet, select cell **A2**. 3. Paste the copied data. Columns B–M in this worksheet automatically update to reflect other data associated with this path. 4. Remove any duplicate values in column **A**. For instructions, see [Remove duplicate values](#) (Microsoft Support website). 5. In the **Target pattern or service** (F) column, review the recommended target AWS service and update as needed. For more informati on, see *Choosing an AWS file system service* in the [Best practices](#) section of this pattern. 6. In the **Migration method** (G) column, review the recommended migration tool and update as needed. For more	Migration engineer, Migration lead

Task	Description	Skills required
	information, see *Choosing a migration tool* in the [Best practices](#) section of this pattern. 7. Save the **SFS-Discovery-Workbook** file. You have finished creating a wave plan for this wave. 8. Repeat these instructions to prepare a wave plan for each wave. Because wave plans are subject to change during the migration, we recommend that you plan no more than 5 waves in advance.	

Prepare

Task	Description	Skills required
Set up the target file system.	According to the details recorded in your wave plan, set up the target file systems in the target AWS account, VPC, and subnets. For instructions, see the following AWS documentation: • Amazon EFS • Amazon FSx for NetApp ONTAP	Migration engineer, Migration lead, AWS administrator

Task	Description	Skills required
	• Amazon FSx for Windows File Server	

Task	Description	Skills required

Set up the migration tool and transfer data.	1. If you're using AWS DataSync, configure logging for DataSync tasks. For instructions, see [Logging your AWS DataSync task activities]().
	2. Set up the migration tool and perform an initial data transfer according to the instructions for your selected tool:
	• For Amazon EFS, see the following:
	  ◦ [Transfer files to Amazon EFS using AWS DataSync]()
	• For Amazon FSx for ONTAP, see the following :
	  ◦ [Migrating to FSx for ONTAP using NetApp SnapMirror]()
	  ◦ [Migrating to FSx for ONTAP using AWS DataSync]()
	• For Amazon FSx for Windows File Server, see the following:
	  ◦ [Migrating existing files to FSx for Windows File Server using AWS DataSync]()
		AWS administrator, Cloud administrator, Migration engineer, Migration lead

Task	Description	Skills required
	- Migrating existing files to FSx for Windows File Server using Robocopy 3. Changes to the source SFS might occur during or after the initial transfer. Set up recurring data transfers between the source and target file systems to keep data synchronized: - If you're using DataSync, see Scheduling your AWS DataSync task. DataSync transfers only the modified or new files in the source SFS. - If you're using a third-party tool, see the documentation for your selected tool.	

Task	Description	Skills required
Update the wave plan.	1. Open the **SFS-Wave-Plan-Workbook** file for the current wave. 2. On the **Build–Wave** worksheet, in the **New path IP address** (N) column, enter the IP address of the target file system. Do one of the following to locate the IP address: • For FSx for Windows File Server, on the Amazon FSx console, choose **File systems**, choose your file system, and then view the **Network & Security** section. • For FSx for ONTAP, see Mounting volumes. • For Amazon EFS, see Mounting with an IP address. 3. In the **New path** (O) column, enter the new mount path. The mount path is the DNS name of the file system. Do one of the following to locate the mount path: • For FSx for Windows File Server, on the Amazon FSx console, choose **File**	Migration engineer, Migration lead

Task	Description	Skills required
	systems, choose your file system, and then choose **Attach**. • For FSx for ONTAP, see the **File system details** page. For instructions, see [Mounting volumes](#). • For Amazon EFS, see [Gather Information](#). 4. On the **Remount-Summary** worksheet, confirm that the **New path** (C) and **New path IP address** (D) columns reflect the updated values. 5. Confirm that your organization has prepared runbooks for remounting the Linux and Windows file systems after cutover. For general instructions, see the following: • [Mounting EFS file systems](#) • [Accessing FSx for Windows File Server file shares](#) • [Mounting FSx for ONTAP volumes](#) 6. If any dependent servers are not included in this wave, record them on the **App-Team-Communica**	

Task	Description	Skills required
	tion worksheet. Inform the respective application or server owners because they might not be included in the standard wave communications. 7. If SFSs are removed from the wave after completin g the wave plan, track these on the **Descoped** worksheet.	

Cut over

Task	Description	Skills required
Stop applications.	If applications or clients are actively performing read and write operations in the source SFS, stop them before you perform the final data sync. For instructions, see the application documenta tion or your internal processes for stopping read and write activities. For example, see Start or Stop the Web Server (IIS 8) (Microsoft documentation) or Managing system services with systemctl (Red Hat documentation).	App owner, App developer

Task	Description	Skills required
Perform the final data transfer.	1. In the migration tool, manually run a final data transfer task or job to synchronize the target file system with the source SFS. For instructions, see [Starting your DataSync task]() or see the documentation for your selected third-party migration tool. 2. Wait for the data transfer task to complete. For more information, see AWS [Monitoring AWS DataSync activity with Amazon CloudWatch]() and [Monitoring your DataSync task from the command line]().	Migration engineer, Migration lead

Task	Description	Skills required
Validate the data transfer.	If you're using AWS DataSync, do the following to validate the final data transfer completed successfully: 1. In the AWS DataSync console, make a note of the task and execution ID, such as `task-0000-exec-1111` . 2. Navigate to the **Task Logging** section of the DataSync task. 3. Choose the **CloudWatch log group** link. 4. In the logs, search for the task and execution ID. 5. Make note of any transfer errors. For more informati on, see [Common Errors](#) in the DataSync documenta tion. 6. Validate the following: • Compare the file lists from the source and target SFSs to confirm that all data has been transferred • Compare the file access permissions between the source and target SFSs.	Migration engineer, Migration lead

Task	Description	Skills required
	If you're using a third-party tool, see the data transfer validation instructions in the documentation for the selected migration tool.	

Validate

Task	Description	Skills required
Remount the file system and validate application function and performance.	1. If dependent servers were migrated in this wave, in the **SFS-Wave-Plan-Workbook** file, on the **Remount-Summary** worksheet, enter the new IP address of the server in the **New server IP address** (F) column. 2. On all servers, update the mount point for the file system from the old path to the new path. Use your organization's runbook for remounting previously discussed in the *Prepare* phase. 3. Confirm that the file system is mounted properly and accessible by checking the mounts and verifying files are present. The infrastructure team typically performs these activities.	AWS systems administrator, App owner

Task	Description	Skills required
	4. Restart the applications and engage the applicati on owners or QA team to complete functional and performance testing on the application, as needed for the application.	

Troubleshooting

Issue	Solution
Cell values in Microsoft Excel don’t update.	Copy the formulas in the sample rows by dragging the fill handle. For more informati on, see instructions for Windows or for Mac (Microsoft Support website).

Related resources

AWS documentation

- AWS DataSync documentation
- Amazon EFS documentation
- Amazon FSx documentation
- Large migrations to the AWS Cloud
  - Guide for AWS large migrations
  - Portfolio playbook for AWS large migrations

Troubleshooting

- Troubleshooting AWS DataSync issues
- Troubleshooting Amazon EFS
- Troubleshooting Amazon FSx for Windows File Server

- [Troubleshooting Amazon FSx for NetApp ONTAP]

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Migrate an Oracle database to Amazon RDS for Oracle by using Oracle GoldenGate flat file adapters

Dhairya Jindani and Baji Shaik, Amazon Web Services

Summary

Oracle GoldenGate is a real-time data capture and replication service for heterogeneous databases and IT environments. However, this service doesn't currently support Amazon Relational Database Service (Amazon RDS) for Oracle. For a list of supported databases, see Oracle GoldenGate for Heterogeneous Databases (Oracle documentation). This pattern describes how to use Oracle GoldenGate and Oracle GoldenGate flat file adapters to generate flat files from the source Oracle database, which can be on-premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance. You can then import those flat files to an Amazon RDS for Oracle database instance.

In this pattern, you use Oracle GoldenGate to extract the trail files from your source Oracle database. The data pump copies the trail files to an integration server, which is an Amazon EC2 instance. On the integration server, Oracle GoldenGate uses the flat file adapter to generate a series of sequential flat files based on the transational data capture of the trail files.Oracle GoldenGate formats the data as either delimiter-separated values or length-delimited values. You then use Oracle SQL*Loader to import the flat files into the target Amazon RDS for Oracle database instance.

Intended audience

This pattern is intended for those who have experience with and knowledge of an Oracle GoldenGate's fundamental building blocks. For more information, see Overview of the Oracle GoldenGate Architecture (Oracle documentation).

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An Oracle GoldenGate license.
- A separate license for an Oracle GoldenGate adapter.
- A source Oracle database, either running on-premises or on an Amazon EC2 instance.
- An Amazon EC2 Linux instance that is used as the integration server. For more information, see Get started with Amazon EC2 Linux instances (Amazon EC2 documentation).

- A target Amazon RDS for Oracle database instance. For more information, see [Creating an Oracle DB instance](#) (Amazon RDS documentation).

Product versions

- Oracle Database Enterprise Edition version 10g, 11g, 12c, or later
- Oracle GoldenGate version 12.2.0.1.1 or later

Architecture

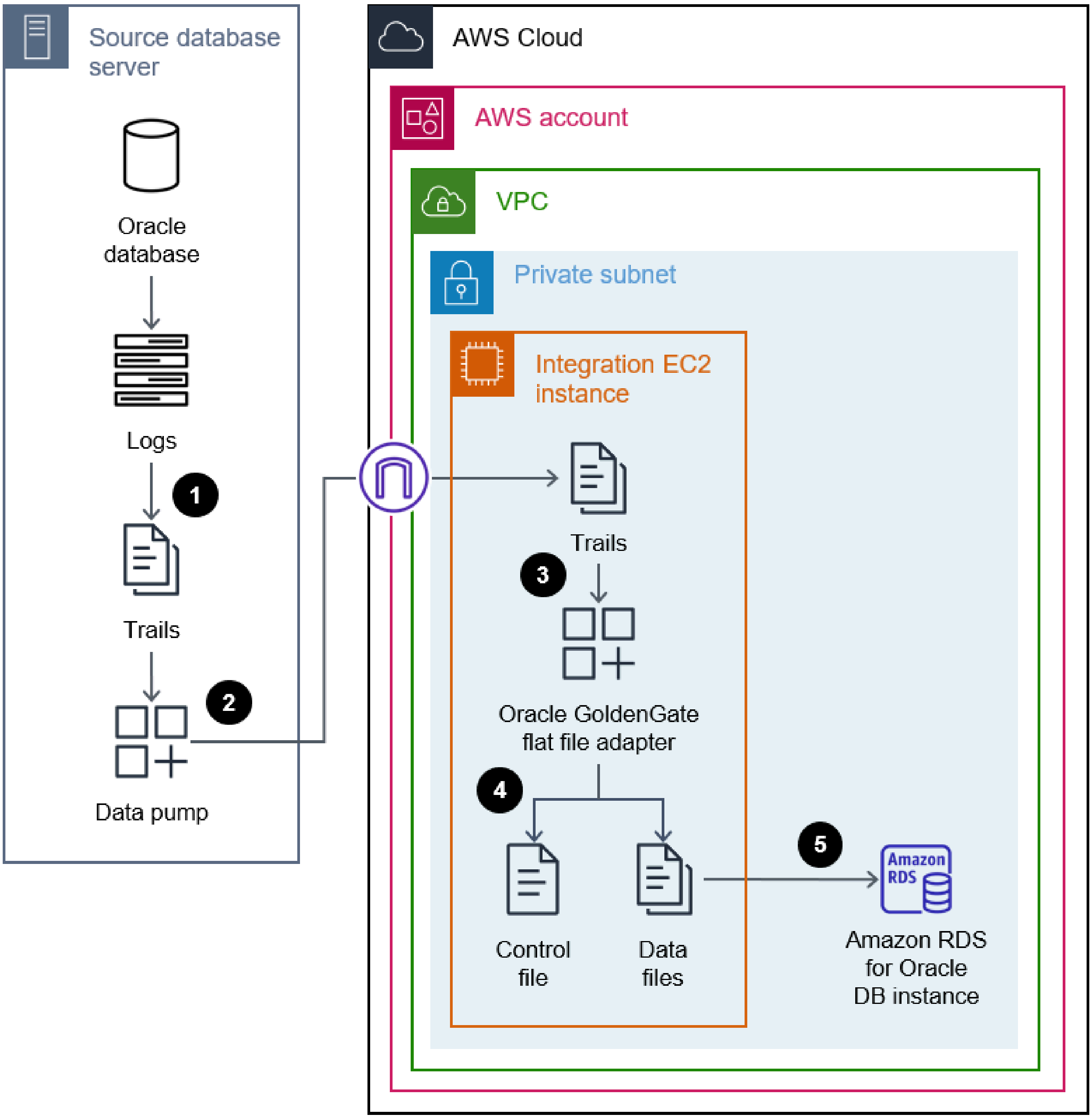
Source technology stack

An Oracle database (on premises or on an Amazon EC2 instance)

Target technology stack

Amazon RDS for Oracle

Source and target architecture



1. Oracle GoldenGate extracts trails from the source database logs.
2. The data pump extracts the trails and migrates them to an integration server.
3. The Oracle GoldenGate flat file adapter reads the trails, source definitions, and extract parameters.

4. You exit the extraction, which generates a control file and flat data files.
5. You migrate the flat data files to an Amazon RDS for Oracle database instance in the AWS Cloud.

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Relational Database Service (Amazon RDS) for Oracle helps you set up, operate, and scale an Oracle relational database in the AWS Cloud.

Other services

- Oracle GoldenGate is a service that helps you to replicate, filter, and transform data from one database to another heterogeneous database or to another target topology, such as flat files.
- Oracle GoldenGate application adapters enable Oracle GoldenGate to produce a series of sequential flat files and control files from transactional data captured in the trail files of a source database. These adapters are widely used for extract, transform, and load (ETL) operations in data warehouse applications and proprietary or legacy applications. Oracle GoldenGate performs this capture and applies it in near real-time across heterogeneous databases, platforms, and operating systems. The adapters support different formats for the output files, such as CSV or Apache Parquet. You can load these generated files in order to load the data into different heterogeneous databases.

Epics

Set up Oracle GoldenGate on the source database server

Task	Description	Skills required
Download Oracle GoldenGat e.	On the source database server, download Oracle GoldenGate version 12.2.0.1. 1 or later. For instructi ons, see Downloading	DBA

Task	Description	Skills required
	Oracle GoldenGate (Oracle documentation).	
Install Oracle GoldenGate.	For instructions, see Installing Oracle GoldenGate (Oracle documentation).	DBA
Set up Oracle GoldenGate.	For instructions, see Preparing the Database for Oracle GoldenGate (Oracle documentation).	DBA

Set up Oracle GoldenGate on the integration server

Task	Description	Skills required
Download Oracle GoldenGate.	On the integration server, download Oracle GoldenGate version 12.2.0.1.1 or later. For instructions, see Downloading Oracle GoldenGate (Oracle documentation).	DBA
Install Oracle GoldenGate.	Create directories, set up the manager process, and create the `defgen` file for a heterogeneous environment. For instructions, see Installing Oracle GoldenGate (Oracle documentation).	DBA

Change the Oracle GoldenGate data capture configuration

Task	Description	Skills required
Prepare the Oracle GoldenGate adapters.	On the integration server, set up the Oracle GoldenGate adapter software. Do the following: 1. From Oracle Software Delivery Cloud, download **ggs_Adapters_Linux_x64.zip**. 2. Unzip **ggs_Adapters_Linux_x64.zip**. 3. Run the following command to install the adapters: `tar -xvf ggs_Adapters_Linux_x64.tar`	DBA
Configure the data pump.	On the source server, configure the data pump to transfer the trail file from the source server to the integration server. Create the data pump parameter file and trails file directory. For instructions, see Configuring the Flat File Adapter (Oracle documentation).	DBA

Generate and migrate the flat files

Task	Description	Skills required
Generate the flat files.	Create the extract file and control file, and then start the extraction process on the integration server. This extracts the database changes and writes the source database to the flat files. For instructions, see Using the Flat File Adapter (Oracle documentation).	DBA
Load the flat files to the target database.	Load the flat files into the target Amazon RDS for Oracle database instance. For more information, see Importing using Oracle SQL*Loader (Amazon RDS documentation).	DBA

Troubleshooting

Issue	Solution
The Oracle GoldenGate flat file adapter generates an error.	For a description of the adapter errors, see Locating Error Messages (Oracle documentation). For troubleshooting instructions, see Troubleshooting the Flat File Adapter (Oracle documentation).

Related resources

- Installing Oracle GoldenGate (Oracle documentation)

- Configuring Oracle GoldenGate (Oracle documentation)
- Understanding Oracle GoldenGate Adapters (Oracle documentation)
- Configuring the Flat File Adapter (Oracle documentation)

Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition

Dwarika Patra and Deepesh Jayaprakash, Amazon Web Services

Summary

This pattern describes changes to application repositories that might be required when you migrate databases from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition. The pattern assumes that these applications are Python-based or Perl-based, and provides separate instructions for these scripting languages.

Migrating SQL Server databases to Aurora PostgreSQL-Compatible involves schema conversion, database object conversion, data migration, and data loading. Because of the differences between PostgreSQL and SQL Server (relating to data types, connection objects, syntax, and logic), the most difficult migration task involves making the necessary changes to the code base so that it works correctly with PostgreSQL.

For a Python-based application, connection objects and classes are scattered throughout the system. Also, the Python code base might use multiple libraries to connect to the database. If the database connection interface changes, the objects that run the application's inline queries also require changes.

For a Perl-based application, changes involve connection objects, database connection drivers, static and dynamic inline SQL statements, and how the application handles complex dynamic DML queries and results sets.

When you migrate your application, you can also consider possible enhancements on AWS, such as replacing the FTP server with Amazon Simple Storage Service (Amazon S3) access.

The application migration process involves the following challenges:

- Connection objects. If connection objects are scattered in the code with multiple libraries and function calls, you might have to find a generalized way to change them to support PostgreSQL.
- Error or exception handling during record retrieval or updates. If you have conditional create, read, update, and delete (CRUD) operations on the database that return variables, results sets, or data frames, any errors or exceptions might result in application errors with cascading effects. These should be handled carefully with proper validations and save points. One such save point is to call large inline SQL queries or database objects inside `BEGIN...EXCEPTION...END` blocks.

- Controlling transactions and their validation. These includes manual and automatic commits and rollbacks. The PostgreSQL driver for Perl requires you to always explicitly set the auto-commit attribute.
- Handling dynamic SQL queries. This requires a strong understanding of the query logic and iterative testing to ensure that queries work as expected.
- Performance. You should ensure that code changes don't result in degraded application performance.

This pattern explains the conversion process in detail.

Prerequisites and limitations

Prerequisites

- Working knowledge of Python and Perl syntax.
- Basic skills in SQL Server and PostgreSQL.
- Understanding of your existing application architecture.
- Access to your application code, SQL Server database, and PostgreSQL database.
- Access to Windows or Linux (or other Unix) development environment with credentials for developing, testing, and validating application changes.
- For a Python-based application, standard Python libraries that your application might require, such as **Pandas** to handle data frames, and **psycopg2** or **SQLAlchemy** for database connections.
- For a Perl-based application, required Perl packages with dependent libraries or modules. The Comprehensive Perl Archive Network (CPAN) module can support most application requirements.
- All required dependent customized libraries or modules.
- Database credentials for read access to SQL Server and read/write access to Aurora.
- PostgreSQL to validate and debug application changes with services and users.
- Access to development tools during application migration such as Visual Studio Code, Sublime Text, or **pgAdmin**.

Limitations

- Some Python or Perl versions, modules, libraries, and packages aren't compatible with the cloud environment.

- Some third-party libraries and frameworks used for SQL Server cannot be replaced to support PostgreSQL migration.
- Performance variations might require changes to your application, to inline Transact-SQL (T-SQL) queries, database functions, and stored procedures.
- PostgreSQL supports lowercase names for table names, column names, and other database objects.
- Some data types, such as UUID columns, are stored in lowercase only. Python and Perl applications must handle such case differences.
- Character encoding differences must be handled with the correct data type for the corresponding text columns in the PostgreSQL database.

Product versions

- Python 3.6 or later (use the version that supports your operating system)
- Perl 5.8.3 or later (use the version that supports your operating system)
- Aurora PostgreSQL-Compatible Edition 4.2 or later (see details)

Architecture

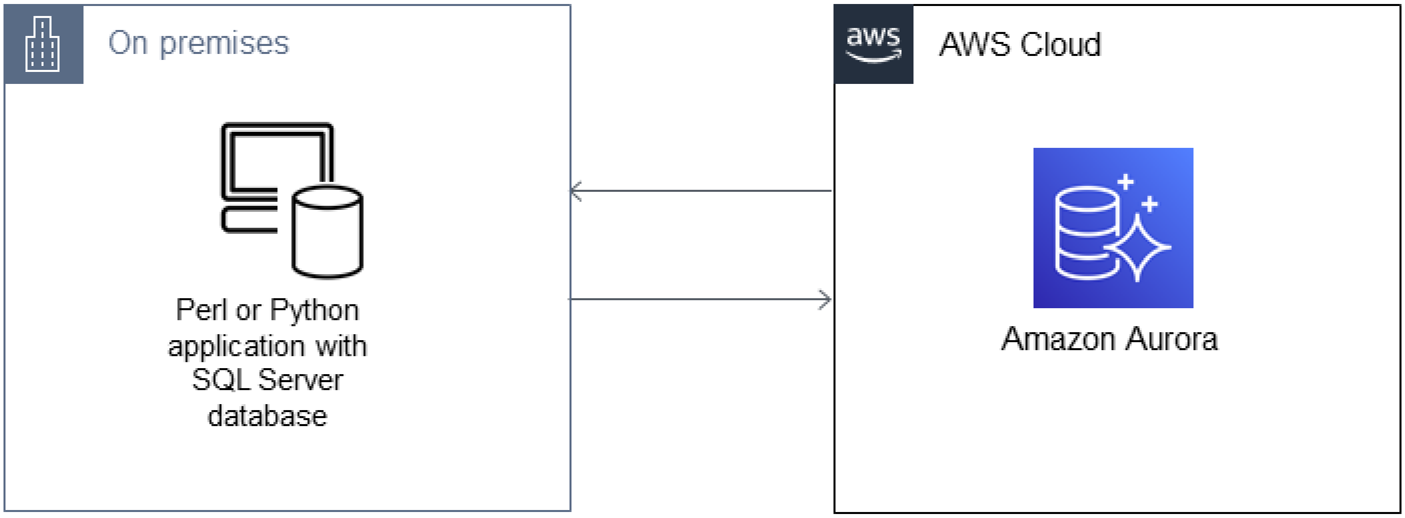
Source technology stack

- Scripting (application programming) language: Python 2.7 or later, or Perl 5.8
- Database: Microsoft SQL Server version 13
- Operating system: Red Hat Enterprise Linux (RHEL) 7

Target technology stack

- Scripting (application programming) language: Python 3.6 or later, or Perl 5.8 or later
- Database: Aurora PostgreSQL-Compatible 4.2
- Operating system: RHEL 7

Migration architecture



Tools

AWS services and tools

- Aurora PostgreSQL–Compatible Edition is a fully managed, PostgreSQL-compatible, and ACID-compliant relational database engine that combines the speed and reliability of high-end commercial databases with the cost-effectiveness of open-source databases. Aurora PostgreSQL is a drop-in replacement for PostgreSQL and makes it easier and more cost-effective to set up, operate, and scale your new and existing PostgreSQL deployments.
- AWS Command Line Interface (AWS CLI) is an open-source tool that enables you to interact with AWS services by using commands in your command-line shell.

Other tools

- Python and PostgresSQL database connection libraries such as psycopg2 and SQLAlchemy
- Perl and its DBI modules
- PostgreSQL interactive terminal (psql)

Epics

Migrate your application repository to PostgreSQL – high-level steps

Task	Description	Skills required
Follow these code conversion steps to migrate your application to PostgreSQL.	1. Set database-specific ODBC drivers and libraries for PostgreSQL. For example, you can use one of the CPAN modules for Perl and **pyodbc**, **psycopg2**, or **SQLAlchemy** for Python. 2. Convert database objects by using these libraries to connect to Aurora PostgreSQL-Compatible. 3. Apply code changes in existing application modules to get compatible T-SQL statements. 4. Rewrite database-specific function calls and stored procedures in application code. 5. Handle changes to your application's variables and their data types that are used for inline SQL queries. 6. Handle incompatible database-specific functions. 7. Complete end-to-end testing of converted	App developer

Task	Description	Skills required
	application code for database migration. 8. Compare results from Microsoft SQL Server against the application you migrated to PostgreSQL. 9. Perform application performance benchmark ing between Microsoft SQL Server and PostgreSQL. 10Revise stored procedures or inline T-SQL statements called by the application to improve performance. The following epics provide detailed instructions for some of these conversion tasks for Python and Perl applications.	

Task	Description	Skills required
Use a checklist for each step of the migration.	Add the following to your checklist for each step of application migration, including the final step: • Review PostgreSQL documentation to ensure that all your changes are compatible with the PostgreSQL standard. • Check for integer and floating values for columns. • Identify the number of rows inserted, updated, and extracted, along with the column names and date/ time stamps. You can use a diff utility or write a script to automate these checks. • Complete performance checks for large inline SQL statements, and check the overall performance of the application. • Check for correct error handling for database operations and graceful program exit by using multiple try/catch blocks. • Check to ensure that proper logging processes are in place.	App developer

Analyze and update your application – Python code base

Task	Description	Skills required
Analyze your existing Python code base.	Your analysis should include the following to facilitate the application migration process: • Identify all connection objects in the code. • Identify all incompatible inline SQL queries (such as T-SQL statements and stored procedures) and analyze required changes. • Review the documentation for your code and track the control flow to understan d code functionality. This will be helpful later when you test the application for performance or load comparisons. • Understand the purpose of the application so you can test it effectively after database conversion. Most Python applications that are candidates for conversio n with database migration s are either feeds that load data from other sources into database tables, or extractors that retrieve data from the tables and transform them into	App developer

Task	Description	Skills required
	different output formats (such as CSV, JSON, or flat files) that are suitable for creating reports or for making API calls to perform validations.	

Task	Description	Skills required

Convert your database connections to support PostgreSQL.	Most Python applications use the **pyodbc** library to connect with SQL Server databases as follows.
	``
	import pyodbc
	....
	try:
	    conn_string =
	 "Driver=ODBC Driver 17
	 for SQL
	    Server;UID={};PWD=
	{};Server={};Datab
	ase={}".format
	 (conn_user, conn_pass
	word,
	    conn_server,
	 conn_database)
	    conn = pyodbc.co
	nnect(conn_string)
	    cur = conn.cursor()
	    result = cur.execu
	te(query_string)
	    for row in result:
	    print (row)
	except Exception as e:
	    print(str(e))
	``
	Convert the database connection to support PostgreSQL as follows.
	``
	import pyodbc
	import psycopg2
	....
	try:
	    conn_string =
	 'postgresql+psycop
	g2://'+
	``
		App developer

Task	Description	Skills required

	``
	    conn_user+':'+conn_password+'@'+conn_server+'/'+conn_database
	    conn = pyodbc.connect(conn_string, connect_args={'options':'-csearch_path=dbo'})
	    cur = conn.cursor()
	    result = cur.execute(query_string)
	    for row in result:
	    print (row)
	except Exception as e:
	    print(str(e))
	``	

Task	Description	Skills required

Change inline SQL queries to PostgreSQL.	Convert your inline SQL queries to a PostgreSQL-compatible format. For example, the following SQL Server query retrieves a string from a table.
	```dtype = "type1" stm = ‘"SELECT TOP 1 searchcode FROM TypesTable (NOLOCK) WHERE code="’ + "’" + str(dtype) + "’" # For Microsoft SQL Server Database Connection engine = create_engine(‘mssql+pyodbc:///?odbc_connect=%s’ % urllib.parse.quote_plus(conn_string), connect_args={‘connect_timeout’:login_timeout}) conn = engine_connect() rs = conn.execute(stm) for row in rs: print(row)```
	After conversion, the PostgreSQL-compatible inline SQL query looks like the following.
	```dtype = "type1" stm = ‘"SELECT searchcode FROM TypesTable```
		App developer

Task	Description	Skills required

	``
	WHERE code="' + "'" +
	 str(dtype) + "' LIMIT
	 1"
	# For PostgreSQL
	 Database Connection
	engine = create_en
	gine('postgres+psy
	copg2://%s' %conn_str
	ing, connect_a
	rgs={'connect_time
	out':login_timeout})
	conn = engine.connect()
	rs = conn.execute(stm)
	for row in rs:
	    print(row)
	``	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Handle dynamic SQL queries.</td><td>Dynamic SQL can be present in one script or in multiple Python scripts. Earlier examples showed how to use Python’s string replace function to insert variables for constructing dynamic SQL queries. An alternate approach is to append the query string with variables wherever applicable.

In the following example, the query string is constructed on the fly based on the values returned by a function.

<pre>query = ‘"SELECT id from
 equity e join issues i
 on e.permId=i.permId
 where e.id’"
query += get_id_fi
lter(ids) + " e.id is
 NOT NULL</pre>
These types of dynamic queries are very common during application migration . Follow these steps to handle dynamic queries:
Check the overall syntax (for example, the syntax for the <code>SELECT</code> statement with a <code>JOIN</code> clause).</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
	• Verify all variables or column names used in the query, such as `i` and `id`. • Check the functions, arguments, and return values used in the query (for example, `get_id_filter` and its argument `ids`).	

Task	Description	Skills required

Handle results sets, variables, and data frames.	For Microsoft SQL Server, you use Python methods such as `fetchone()` or `fetchall()` to retrieve the results set from the database. You can also use `fetchmany(size)` and specify the number of records to return from the results set. To do this, you can use the **pyodbc** connection object as shown in the following example.
	
	pyodbc (Microsoft SQL Server)
	
	`import pyodbc`
	`server = 'tcp:myserver.database.windows.net'`
	`database = 'exampledb'`
	`username = 'exampleusername'`
	`password = 'examplepassword'`
	`conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)`
	`cursor = conn.cursor()`
	`cursor.execute("SELECT * FROM ITEMS")`
	`row = cursor.fetchone()`
	`while row:`
		App developer

Task	Description	Skills required

	``
	    print(row[0])
	    row = cursor.fe
	tchone()
	``
	
	In Aurora, to perform similar tasks such as connectin g to PostgreSQL and fetching results sets, you can use either **psycopg2** or **SQLAlchemy**. These Python libraries provide the connection module and cursor object to traverse through the PostgreSQL database records, as shown in the following example.
	
	psycopg2 (Aurora PostgreSQ L-Compatible)
	
	``
	import psycopg2
	query = "SELECT * FROM
	 ITEMS;"
	//Initialize
	 variables
	host=dbname=user=
	password=port=sslm
	ode=connect_timeou
	t=""
	connstring = "host='{h
	ost}' dbname='{
	dbname}' user='{user}'
	 \
	password='{passw
	ord}'port='{port}'
	".format(host=host
	,dbname=dbname,\
	user=user,password=
	password,port=port)
	``	

Task	Description	Skills required

	``` conn = psycopg2. connect(connstring) cursor = conn.cursor() cursor.execute(query) column_names = [column[0] for column in cursor.description] print("Column Names: ", column_names) print("Column values: " for row in cursor: print("itemid :", row[0]) print("itemdescrpt ion :", row[1]) print("it emprice :", row[3])) ```
	 SQLAlchemy (Aurora PostgreSQL-Compatible)
	 ``` from sqlalchemy import create_engine from pandas import DataFrame conn_string = 'postgres ql://core:database @localhost:5432/ex ampledatabase' engine = create_en gine(conn_string) conn = engine.co nnect() dataid = 1001 result = conn.exec ute("SELECT * FROM ITEMS") df = DataFrame (result.fetchall()) ```	

Task	Description	Skills required
	``` df.columns = result.ke ys() df = pd.DataFrame() engine.connect() df = pd.read_sql_query(sql_query, engine, coerce_float=False) print("df=", df) ```	
Test your application during and after migration.	Testing the migrated Python application is an ongoing process. Because the migration includes connection object changes (**psycopg2** or **SQLAlchemy**), error handling, new features (data frames), inline SQL changes, bulk copy functionalities (`bcp` instead of `COPY`) and similar changes, it must be tested carefully during and after application migration. Check for: • Error conditions and handling • Any record mismatches after migration • Record updates or deletions • Time required to run the application	App developer

Analyze and update your application – Perl code base

Task	Description	Skills required
Analyze your existing Perl code base.	Your analysis should include the following to facilitate the application migration process. You should identify: • Any INI or configuration-based code • Database-specific standard Open Database Connectiv ity (ODBC) Perl drivers or any customized drivers • Code changes required for inline and T-SQL queries • Interactions among various Perl modules (for example, a single Perl ODBC connection object that is called or used by multiple functional components) • Dataset and results set handling • External, dependent Perl libraries • Any APIs that are used in the application • Perl version compatibility and driver compatibility with Aurora PostgreSQL-Compatible	App developer

Task	Description	Skills required

Convert the connections from the Perl application and DBI module to support PostgreSQL.	Perl-based applications generally use the Perl DBI module, which is a standard database access module for the Perl programming language. You can use the same DBI module with different drivers for SQL Server and PostgreSQL.
	
	For more information about required Perl modules, installations, and other instructions, see the [DBD::Pg documentation](). The following example connects to Aurora PostgreSQL-Compatible at `exampletest-aurorapg-database.cluster-sampleclusture.us-east-.rds.amazonaws.com` .
	
	`#!/usr/bin/perl`
	`use DBI;`
	`use strict;`
	`my $driver = "Pg";`
	`my $hostname = "exampletest-aurorapg-database-sampleclusture.us-east.rds.amazonaws.com"`
	`my $dsn = "DBI:$driver: dbname = $hostname ;host = 127.0.0.1;port = 5432";`
		App developer

Task	Description	Skills required

	<pre>my $username = "postgres
	";
	my $password = "pass123"
	;
	$dbh = DBI->conn
	ect("dbi:Pg:dbname
	=$hostname;host=$h
	ost;port=$port;opt
	ions=$options",
	      $username,
	      $password,
	      {AutoCommit =>
	 0, RaiseError => 1,
	 PrintError => 0}
	      );</pre>	

Task	Description	Skills required

Change Inline SQL queries to PostgreSQL.	Your application might have inline SQL queries with `SELECT`, `DELETE`, `UPDATE`, and similar statements that include query clauses that PostgreSQL doesn't support. For example, query keywords such as `TOP` and `NOLOCK` aren't supported in PostgreSQL. The following examples show how you can handle `TOP`, `NOLOCK`, and Boolean variables.
	
	In SQL Server:
	
	`$sqlStr = $sqlStr . "WHERE a.student_id in (SELECT TOP $numofRecords c_student_id \ FROM active_student_record b WITH (NOLOCK) \ INNER JOIN student_contributor c WITH (NOLOCK) on c.contributor_id = b.c_st)`
	
	For PostgreSQL, convert to:
	
	`$sqlStr = $sqlStr . "WHERE a.student_id in (SELECT TOP $numofRecords c_student_id \ FROM active_student_record b INNER JOIN`
		App developer

Task	Description	Skills required
	``` student_contributor c \ on c.contributor_id = b.c_student_contr_id WHERE b_current_1 is true \ LIMIT $numofRecords)" ```	

Task	Description	Skills required
Handle dynamic SQL queries and Perl variables.	Dynamic SQL queries are SQL statements that are built at application runtime. These queries are construct ed dynamically when the application is running, depending on certain conditions, so the full text of the query isn’t known until runtime. An example is a financial analytics applicati on that analyzes the top 10 shares on a daily basis, and these shares change every day. The SQL tables are created based on top performers, and the values aren’t known until runtime. Let’s say that the inline SQL queries for this example are passed to a wrapper function to get the results set in a variable, and then a variable uses a condition to determine whether the table exists: • If the table exists, don't create it; do some processin g. • If the table doesn’t exist, create the table and also do some processing.	App developer

Task	Description	Skills required

	Here’s an example of variable handling, followed by the SQL Server and PostgreSQL queries for this use case.
	`my $tableexists = db_read( arg 1, $sql_qry, undef, 'writer'); my $table_already_exists = $tableexists->[0]{table_exists}; if ($table_already_exists){ # do some thing } else { # do something else }`
	SQL Server:
	`my $sql_qry = "SELECT OBJECT_ID('$backendTable', 'U') table_exists", undef, 'writer')";`
	PostgreSQL:
	`my $sql_qry = "SELECT TO_REGCLASS('$backendTable', 'U') table_exists", undef, 'writer')";`
	The following example uses a Perl variable in	

Task	Description	Skills required

	inline SQL, which runs a `SELECT` statement with a `JOIN` to fetch the primary key of the table and position of the key column.
	
	SQL Server:
	
	`my $sql_qry = "SELECT column_name', character_maximum_length \ FROM INFORMATION_SCHEMA.COLUMNS \ WHERE TABLE_SCHEMA='$example_schemaInfo' \ AND TABLE_NAME='$example_table' \ AND DATA_TYPE IN ('varchar','nvarchar');";`
	
	PostgreSQL:
	
	`my $sql_qry = "SELECT c1.column_name, c1.ordinal_position \ FROM information_schema.key_column_usage AS c LEFT \ JOIN information_schema.table_constraints AS t1 \ ON t1.constraint_name = c1.constraint_name \ WHERE t1.table_name = $example_schemaInfo'.'$example_table’ \`	

Task	Description	Skills required
	`AND t1.constraint_type = 'PRIMARY KEY' ;";`	

Make additional changes to your Perl-based or Python-based application to support PostgreSQL

Task	Description	Skills required
Convert additional SQL Server constructs to PostgreSQL.	The following changes apply to all applications, regardless of programming language. • Qualify database objects that your application uses with new and appropriate schema names. • Handle LIKE operators for case-sensitive matching with the collation feature in PostgreSQL. • Handle unsupported database specific functions such as `DATEDIFF`, `DATEADD`, `GETDATE`, `CONVERT`, and `CAST` operators. For equivalent PostgreSQL-compatible functions, see *Native or built-in SQL functions* in the Additional information section. • Handle Boolean values in comparison statements.	App developer

Task	Description	Skills required
	• Handle return values from functions. These could be record sets, data frames, variables, and Boolean values. Handle these according to the requirements of your application and to support PostgreSQL. • Handle anonymous blocks (such as `BEGIN TRAN`) with new, user-defined PostgreSQL functions. • Convert bulk inserts for rows. The PostgreSQL equivalent of the SQL Server bulk copy (`bcp`) utility, which is called from inside the application, is `COPY`. • Convert column concatenation operators. SQL Server uses + for string concatenation, but PostgreSQL uses `||`.	

Improve performance

Task	Description	Skills required
Take advantage of AWS services to make performance enhancements.	When you migrate to the AWS Cloud, you can refine your application and database design to take advantage of AWS services. For example,	App developer, Cloud architect

Task	Description	Skills required
	if the queries from your Python application, which is connected to an Aurora PostgreSQL-Compatible database server, is taking more time than your original Microsoft SQL Server queries, you could consider creating a feed of historical data directly to an Amazon Simple Storage Service (Amazon S3) bucket from the Aurora server, and use Amazon Athena-based SQL queries to generate reports and analytic data queries for your user dashboards.	

Related resources

- Perl
- Perl DBI Module
- Python
- psycopg2
- SQLAlchemy
- Bulk Copy - PostgreSQL
- Bulk Copy - Microsoft SQL Server
- PostgreSQL
- Working with Amazon Aurora PostgreSQL

Additional information

Both Microsoft SQL Server and Aurora PostgreSQL-Compatible are ANSI SQL-complaint. However, you should still be aware of any incompatibilities in syntax, column data types, native database-specific functions, bulk inserts, and case sensitivity when you migrate your Python or Perl application from SQL Server to PostgreSQL.

The following sections provide more information about possible inconsistencies.

Data type comparison

Data type changes from SQL Server to PostgreSQL can lead to significant differences in the resulting data that applications operate on. For a comparison of data types, see the table on the Sqlines website.

Native or built-in SQL functions

The behavior of some functions differs between SQL Server and PostgreSQL databases. The following table provides a comparison.

Microsoft SQL Server	Description	PostgreSQL
`CAST`	Converts a value from one data type to another.	PostgreSQL `type :: operator`
`GETDATE()`	Returns the current database system date and time, in a `YYYY-MM-DD hh:mm:ss.mmm` format.	`CLOCK_TIMESTAMP`
`DATEADD`	Adds a time/date interval to a date.	`INTERVAL` expression
`CONVERT`	Converts a value to a specific data format.	`TO_CHAR`
`DATEDIFF`	Returns the difference between two dates.	`DATE_PART`
`TOP`	Limits the number of rows in a SELECT results set.	`LIMIT/FETCH`

Anonymous blocks

A structured SQL query is organized into sections such as declaration, executables, and exception handling. The following table compares the Microsoft SQL Server and PostgreSQL versions of a simple anonymous block. For complex anonymous blocks, we recommend that you call a custom database function within your application.

Microsoft SQL Server	PostgreSQL

`my $sql_qry1=`
`my $sql_qry2 =`
`my $sqlqry = "BEGIN TRAN`
`$sql_qry1 $sql_qry2`
`if @\@error !=0 ROLLBACK`
`TRAN`
`else COMIT TRAN";`
	`my $sql_qry1=`
	`my $sql_qry2 =`
	`my $sql_qry = " DO \$\$`
	`BEGIN`
	`$header_sql $content_sql`
	`END`
	`\$\$";`

Other differences

- **Bulk inserts of rows:** The PostgreSQL equivalent of the Microsoft SQL Server bcp utility is COPY.
- **Case sensitivity:** Column names are case-sensitive in PostgreSQL, so you have to convert your SQL Server column names to lowercase or uppercase. This becomes a factor when you extract or compare data, or place column names in results sets or variables. The following example identifies columns where values might be stored in uppercase or lowercase.

```
my $sql_qry = "SELECT $record_id FROM $exampleTable WHERE LOWER($record_name) =
 \'failed transaction\'";
```

- **Concatenation:** SQL Server uses + as an operator for string concatenation, whereas PostgreSQL uses ||.
- **Validation:** You should test and validate inline SQL queries and functions before you use them in application code for PostgreSQL.

- **ORM Library inclusion :** You can also look for including or replace existing database connection library with Python ORM libraries such as SQLAlchemy and PynomoDB. This will help to easily query and manipulate data from a database using an object-oriented paradigm.

Stream data from IBM Db2, SAP, Sybase, and other databases to MongoDB Atlas on AWS

Battulga Purevragchaa and Igor Alekseev, Amazon Web Services

Babu Srinivasan, MongoDB

Summary

This pattern describes the steps for migrating data from IBM Db2 and other databases such as mainframe databases and Sybase to MongoDB Atlas on the AWS Cloud. It uses AWS Glue to help accelerate the data migration to MongoDB Atlas.

The pattern accompanies the guide Migrating to MongoDB Atlas on AWS on the AWS Prescriptive Guidance website. It provides the implementation steps for one of the migration scenarios that are discussed in that guide. For additional migration scenarios, see the following patterns on the AWS Prescriptive Guidance website:

- Migrate a self-hosted MongoDB environment to MongoDB Atlas on AWS
- Migrate relational databases to MongoDB Atlas on AWS

The pattern is intended for AWS Managed Services Partners and AWS users.

Prerequisites and limitations

Prerequisites

- A source database such as SAP, Sybase, IBM Db2, and others to migrate to MongoDB Atlas.
- Familiarity with databases such as SAP, Sybase, IBM Db2, MongoDB Atlas, and AWS services.

Product versions

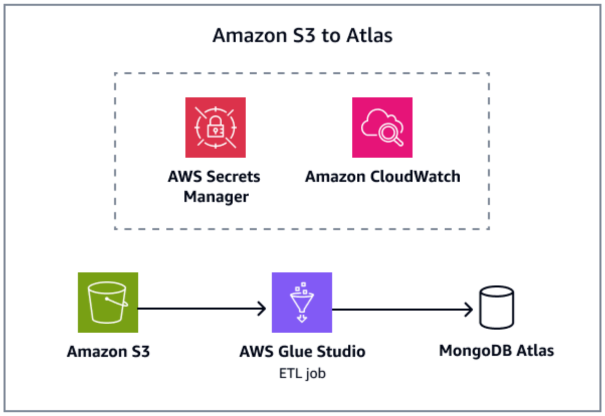
- MongoDB version 5.0 or later.

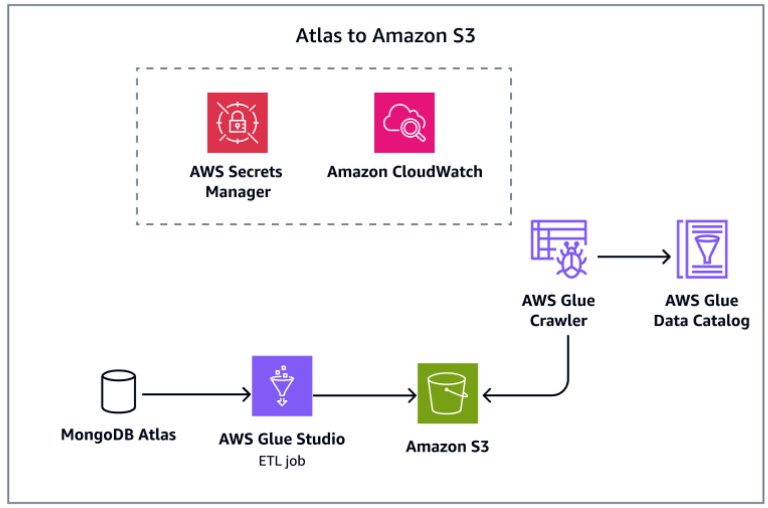
Architecture

The following diagram illustrates batch data load and data streaming by using AWS Glue Studio, Amazon Kinesis Data Streams, and MongoDB Atlas.

This reference architecture uses AWS Glue Studio to create extract, transform, and load (ETL) pipelines to migrate data to MongoDB Atlas. An AWS Glue crawler integrates with MongoDB Atlas to facilitate data governance. The data can be either ported in batch or streamed to MongoDB Atlas by using Amazon Kinesis Data Streams.

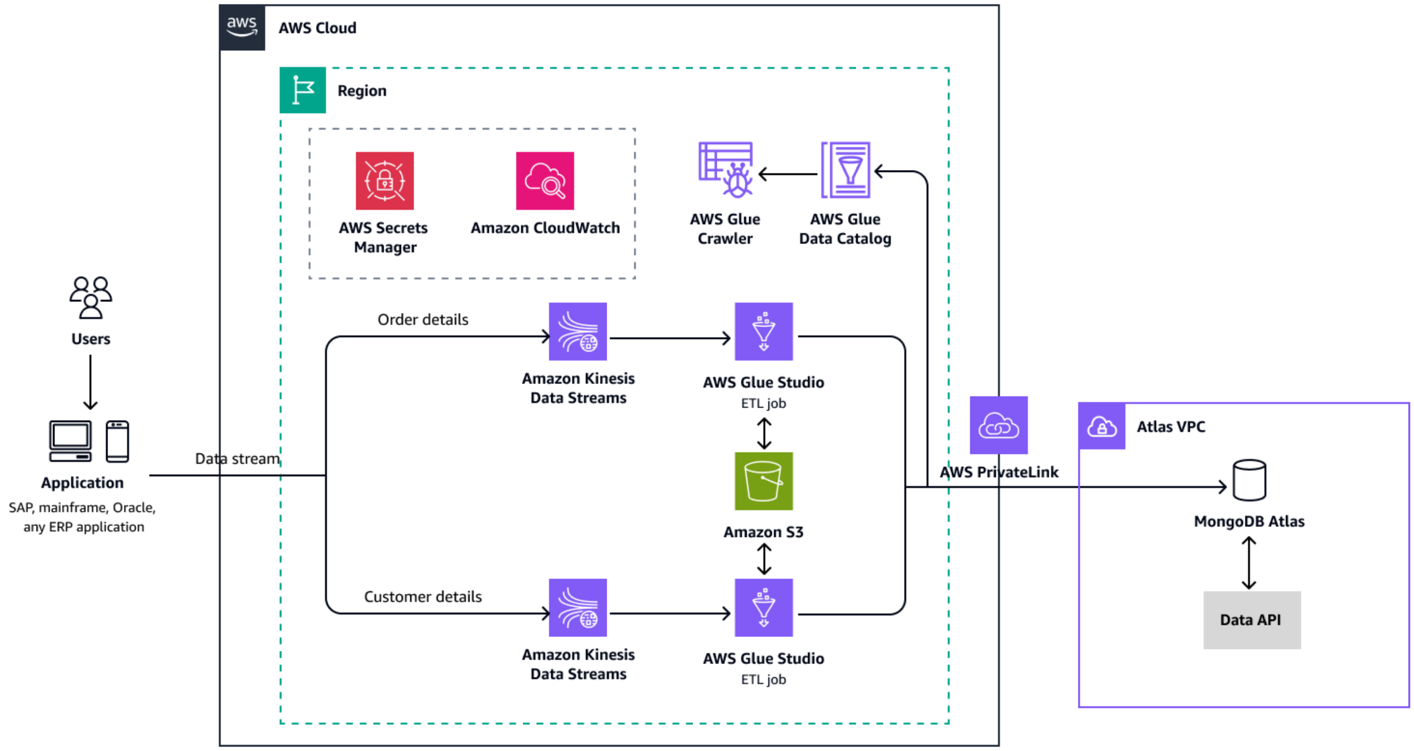
Batch data load





For more information about the batch data migration, see the AWS blog post [Compose your ETL jobs for MongoDB Atlas with AWS Glue](#).

Data streaming



For MongoDB Atlas reference architectures that support different usage scenarios, see [Migrating to MongoDB Atlas on AWS] on the AWS Prescriptive Guidance website.

Tools

- AWS Glue is a fully managed ETL service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- Amazon Kinesis Data Streams helps you collect and process large streams of data records in real time.
- MongoDB Atlas is a fully managed database as a service (DbaaS) for deploying and managing MongoDB databases in the cloud.

Best practices

For guidelines, see Best Practices Guide for MongoDB in the MongoDB GitHub repository.

Epics

Discovery and assessment

Task	Description	Skills required
Determine the cluster size.	Estimate the working set size by using the information from `db.stats()` for the total index space. Assume that a percentage of your data space will be accessed frequentl y. Or, you can estimate your memory requirements based on your assumptions. This task should take approxima tely one week. For more information and examples for this and the other stories in this epic, see the links in the [Related resources](#) section.	MongoDB DBA, Application architect
Estimate network bandwidth requirements.	To estimate your network bandwidth requireme nts, multiply the average document size by the number of documents served per second. Consider the maximum traffic that any node on your cluster will bear as the basis. To calculate downstream data transfer rates from your cluster to client applications, use the sum of the total documents returned over a period of time. If your applications	MongoDB DBA

Task	Description	Skills required
	read from secondary nodes, divide this number of total documents by the number of nodes that can serve read operations. To find the average document size for a database, use the `db.stats().avgObjSize` command. This task will typically take one day.	
Select the Atlas tier.	Follow the instructions in the MongoDB documentation to select the correct Atlas cluster tier.	MongoDB DBA
Plan for cutover.	Plan for application cutover.	MongoDB DBA, Application architect

Set up a new MongoDB Atlas environment on AWS

Task	Description	Skills required
Create a new MongoDB Atlas cluster on AWS.	In MongoDB Atlas, choose **Build a Cluster**, and select AWS as the cloud provider.	MongoDB DBA
Select AWS Regions and global cluster configuration.	Select from the list of available AWS Regions for your Atlas cluster. Configure global clusters if required.	MongoDB DBA
Select the cluster tier.	Select your preferred cluster tier. Your tier selection determines factors such as	MongoDB DBA

Task	Description	Skills required
	memory, storage, and IOPS specification.	
Configure additional cluster settings.	Configure additional cluster settings such as MongoDB version, backup, and encryption options. For more information about these options, see the Related resources section.	MongoDB DBA

Configure security and compliance

Task	Description	Skills required
Configure the access list.	To connect to the Atlas cluster, you must add an entry to the project's access list. Atlas uses Transport Layer Security (TLS) / Secure Sockets Layer (SSL) to encrypt the connections to the virtual private cloud (VPC) for your database. To set up the access list for the project and for more information about the stories in this epic, see the links in the Related resources section.	MongoDB DBA
Authenticate and authorize users.	You must create and authenticate the database users who will access the MongoDB Atlas clusters. To access the clusters in a	MongoDB DBA

Task	Description	Skills required
	project, users must belong to that project, and they can belong to multiple projects. You can also enable authorization with AWS Identity and Access Management (IAM). For more information, see Set Up Authentication with IAM in the MongoDB documentation.	
Create custom roles.	(Optional) Atlas supports creating custom roles if the built-in Atlas database user privileges don't cover your desired set of privileges.	MongoDB DBA
Set up VPC peering.	(Optional) Atlas supports VPC peering with other AWS VPCs.	MongoDB DBA
Set up an AWS PrivateLink endpoint.	(Optional) You can set up private endpoints on AWS by using AWS PrivateLink.	MongoDB DBA
Enable two-factor authentic ation.	(Optional) Atlas supports two-factor authentication (2FA) to help users control access to their Atlas accounts.	MongoDB DBA
Set up user authentication and authorization with LDAP.	(Optional) Atlas supports performing user authentic ation and authorization with Lightweight Directory Access Protocol (LDAP).	MongoDB DBA

Task	Description	Skills required
Set up unified AWS access.	(Optional) Some Atlas features, including Atlas Data Lake and encryption at rest using customer key management, use IAM roles for authentication.	MongoDB DBA
Set up encryption at rest by using AWS KMS.	(Optional) Atlas supports using AWS Key Management Service (AWS KMS) to encrypt storage engines and cloud provider backups.	MongoDB DBA
Set up CSFLE.	(Optional) Atlas supports client-side field-level encryption (CSFLE), including automatic encryption of fields.	MongoDB DBA

Migrate data

Task	Description	Skills required
Launch your target replica set in MongoDB Atlas.	Launch your target replica set in MongoDB Atlas. In Atlas Live Migration Service, choose **I'm ready to migrate**.	MongoDB DBA
Establish the connection of AWS Glue with MongoDB Atlas.	Use an AWS Glue crawler to connect AWS Glue with MongoDB Atlas (target database). This step helps prepare the target environme nt for migration. For more	MongoDB DBA

Task	Description	Skills required
	information, see the [AWS Glue documentation]().	
Establish the connection of AWS Glue with the source database or source stream.	This helps prepare the target environment for migration.	MongoDB DBA
Set up the data transform ation.	Configure the transformation logic to migrate the data from the legacy structured schema to the flexible schema of MongoDB.	MongoDB DBA
Migrate the data.	Schedule the migration in AWS Glue Studio.	MongoDB DBA

Configure operational integration

Task	Description	Skills required
Connect to the cluster.	Connect to the MongoDB Atlas cluster.	App developer
Interact with data.	Interact with cluster data.	App developer
Monitor the clusters.	Monitor your MongoDB Atlas clusters.	MongoDB DBA
Back up and restore data.	Back up and restore cluster data.	MongoDB DBA

Troubleshooting

Issue	Solution
If you encounter issues	See Troubleshooting in the MongoDB Atlas CloudFormation Resources repository.

Related resources

All of the following links, unless noted otherwise, go to webpages in the MongoDB documentation.

Migration guide

- Migrating to MongoDB Atlas on AWS (AWS Prescriptive Guidance)

Discovery and assessment

- Memory
- Sizing example with Atlas sample data sets
- Sizing example for mobile applications
- Network Traffic
- Cluster Auto-Scaling
- Atlas sizing template

Configuring security and compliance

- Configure IP Access List Entries
- Configure Database Users
- Configure Access to the Atlas UI
- Configure Custom Database Roles
- Configure Database Users
- Set up a Network Peering Connection
- Learn About Private Endpoints in Atlas
- Manage Your Multi-Factor Authentication Options

- Set up User Authentication and Authorization with LDAP
- Atlas Data Lake
- Encryption at Rest using Customer Key Management
- Methods to assume a role (IAM documentation)
- Client-Side Field Level Encryption
- Automatic Encryption
- MongoDB Atlas Security Controls
- MongoDB Trust Center
- Configure Security Features for Clusters

Setting up a new MongoDB Atlas environment on AWS

- Cloud Providers and Regions
- Manage Global Clusters
- Select Cluster Tier
- Configure Additional Settings
- Get Started with Atlas
- Configure Access to the Atlas UI
- Manage Clusters

Migrating data

- Migrate or Import Data

Monitoring clusters

- Monitor Your Clusters

Integrating operations

- Connect to a Cluster
- Interact with Your Data
- Monitor Your Clusters

- Backu Up, Restore, and Archive Data

GitHub repository

- Stream data to MongoDB Atlas using AWS Glue

Migration patterns by workload

Topics

- IBM
- Microsoft
- N/A
- Open source
- Oracle
- SAP

IBM

- Migrate a Db2 database from Amazon EC2 to Aurora MySQL-Compatible by using AWS DMS
- Migrate Db2 for LUW to Amazon EC2 by using log shipping to reduce outage time
- Migrate Db2 for LUW to Amazon EC2 with high availability disaster recovery
- Migrate from IBM Db2 on Amazon EC2 to Aurora PostgreSQL-Compatible using AWS DMS and AWS SCT
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2
- Stream data from IBM Db2, SAP, Sybase, and other databases to MongoDB Atlas on AWS

Microsoft

- Accelerate the discovery and migration of Microsoft workloads to AWS
- Change Python and Perl applications to support database migration from Microsoft SQL Server to Amazon Aurora PostgreSQL-Compatible Edition
- Create AWS CloudFormation templates for AWS DMS tasks using Microsoft Excel and Python
- Export a Microsoft SQL Server database to Amazon S3 by using AWS DMS
- Implement SHA1 hashing for PII data when migrating from SQL Server to PostgreSQL
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account
- Migrate a messaging queue from Microsoft Azure Service Bus to Amazon SQS
- Migrate a Microsoft SQL Server database from Amazon EC2 to Amazon DocumentDB by using AWS DMS
- Migrate a Microsoft SQL Server database to Aurora MySQL by using AWS DMS and AWS SCT
- Migrate a .NET application from Microsoft Azure App Service to AWS Elastic Beanstalk
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using linked servers
- Migrate an on-premises Microsoft SQL Server database to Amazon RDS for SQL Server using native backup and restore methods
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS DMS
- Migrate an on-premises Microsoft SQL Server database to Amazon Redshift using AWS SCT data extraction agents
- Migrate an on-premises Microsoft SQL Server database to Microsoft SQL Server on Amazon EC2 running Linux
- Migrate data from Microsoft Azure Blob to Amazon S3 by using Rclone
- Migrate an on-premises Microsoft SQL Server database to Amazon EC2 using Application Migration Service
- Migrate a relational database to MongoDB Atlas on AWS
- Migrate Windows SSL certificates to an Application Load Balancer using ACM
- Rehost on-premises workloads in the AWS Cloud: migration checklist
- Resolve connection errors after migrating Microsoft SQL Server to the AWS Cloud

- Set up Multi-AZ infrastructure for a SQL Server Always On FCI by using Amazon FSx

N/A

- Create an approval process for firewall requests during a rehost migration to AWS

Open source

- Create application users and roles in Aurora PostgreSQL-Compatible
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL with AWS SCT and AWS DMS using AWS CLI and AWS CloudFormation
- Migrate an on-premises MariaDB database to Amazon RDS for MariaDB using native tools
- Migrate an on-premises MySQL database to Amazon EC2
- Migrate an on-premises MySQL database to Amazon RDS for MySQL
- Migrate an on-premises MySQL database to Aurora MySQL
- Migrate an on-premises PostgreSQL database to Aurora PostgreSQL
- Migrate a Couchbase Server database to Amazon EC2
- Migrate from IBM WebSphere Application Server to Apache Tomcat on Amazon EC2 with Auto Scaling
- Migrate from Oracle GlassFish to AWS Elastic Beanstalk
- Migrate from PostgreSQL on Amazon EC2 to Amazon RDS for PostgreSQL using pglogical
- Migrate on-premises Java applications to AWS using AWS App2Container
- Migrate on-premises MySQL databases to Aurora MySQL using Percona XtraBackup, Amazon EFS, and Amazon S3
- Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible
- Migrate Redis workloads to Redis Enterprise Cloud on AWS
- Restart the AWS Replication Agent automatically without disabling SELinux after rebooting a RHEL source server
- Transport PostgreSQL databases between two Amazon RDS DB instances using pg_transport

Oracle

- Convert VARCHAR2(1) data type for Oracle to Boolean data type for Amazon Aurora PostgreSQL
- Emulate Oracle DR by using a PostgreSQL-compatible Aurora global database
- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT
- Load BLOB files into TEXT by using file encoding in Aurora PostgreSQL-Compatible
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL in SSL mode by using AWS DMS
- Migrate an Amazon RDS for Oracle database to another AWS account and AWS Region using AWS DMS for ongoing replication
- Migrate an Amazon RDS for Oracle DB instance to another VPC
- Migrate an on-premises Oracle database to Amazon EC2 by using Oracle Data Pump
- Migrate an on-premises Oracle database to Amazon OpenSearch Service using Logstash
- Migrate an on-premises Oracle database to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate an on-premises Oracle database to Amazon RDS for Oracle
- Migrate an on-premises Oracle database to Amazon RDS for Oracle by using direct Oracle Data Pump Import over a database link
- Migrate an on-premises Oracle database to Amazon RDS for Oracle using Oracle Data Pump
- Migrate an on-premises Oracle database to Amazon RDS for PostgreSQL by using an Oracle bystander and AWS DMS
- Migrate an on-premises Oracle database to Oracle on Amazon EC2
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for MariaDB using AWS DMS and AWS SCT
- Migrate an Oracle database from Amazon EC2 to Amazon RDS for Oracle using AWS DMS
- Migrate an Oracle database to Amazon DynamoDB using AWS DMS
- Migrate an Oracle database to Amazon RDS for Oracle by using Oracle GoldenGate flat file adapters
- Migrate an Oracle Database to Amazon Redshift using AWS DMS and AWS SCT
- Migrate an Oracle database to Aurora PostgreSQL using AWS DMS and AWS SCT
- Migrate an Oracle JD Edwards EnterpriseOne database to AWS by using Oracle Data Pump and AWS DMS

- Migrate an Oracle partitioned table to PostgreSQL by using AWS DMS
- Migrate an Oracle PeopleSoft database to AWS by using AWS DMS
- Migrate data from an on-premises Oracle database to Aurora PostgreSQL
- Migrate from Amazon RDS for Oracle to Amazon RDS for MySQL
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using materialized views and AWS DMS
- Migrate from Oracle 8i or 9i to Amazon RDS for PostgreSQL using SharePlex and AWS DMS
- Migrate from Oracle Database to Amazon RDS for PostgreSQL by using Oracle GoldenGate
- Migrate from Oracle on Amazon EC2 to Amazon RDS for MySQL using AWS DMS and AWS SCT
- Migrate from Oracle to Amazon DocumentDB using AWS DMS
- Migrate from Oracle WebLogic to Apache Tomcat (TomEE) on Amazon ECS
- Migrate function-based indexes from Oracle to PostgreSQL
- Migrate legacy applications from Oracle Pro*C to ECPG
- Migrate Oracle CLOB values to individual rows in PostgreSQL on AWS
- Migrate Oracle Database error codes to an Amazon Aurora PostgreSQL-Compatible database
- Migrate Oracle native functions to PostgreSQL using extensions
- Migrate Oracle PeopleSoft to Amazon RDS Custom
- Migrate Oracle ROWID functionality to PostgreSQL on AWS
- Migrate Oracle SERIALLY_REUSABLE pragma packages into PostgreSQL
- Migrate virtual generated columns from Oracle to PostgreSQL
- Set up Oracle UTL_FILE functionality on Aurora PostgreSQL-Compatible
- Validate database objects after migrating from Oracle to Amazon Aurora PostgreSQL

SAP

- Migrate an on-premises SAP ASE database to Amazon EC2
- Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS
- Migrate SAP ASE on Amazon EC2 to Amazon Aurora PostgreSQL-Compatible using AWS SCT and AWS DMS
- Reduce homogeneous SAP migration cutover time by using Application Migration Service

More patterns

- Access AWS services from IBM z/OS by installing the AWS CLI
- Assess application readiness for migration to the AWS Cloud by using CAST Highlight
- Assess query performance for migrating SQL Server databases to MongoDB Atlas on AWS
- Automate cross-Region failover and failback by using DR Orchestrator Framework
- Automate database tasks in SQL Server Express edition running on Amazon EC2 by using AWS Lambda and Task Scheduler
- Build an advanced mainframe file viewer in the AWS Cloud
- Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode
- Connect to Application Migration Service data and control planes over a private network
- Containerize mainframe workloads that have been modernized by Blu Age
- Convert JSON Oracle queries into PostgreSQL database SQL
- Convert the Teradata NORMALIZE temporal feature to Amazon Redshift SQL
- Convert the Teradata RESET WHEN feature to Amazon Redshift SQL
- Copy Amazon DynamoDB tables across accounts using AWS Backup
- Deploy a Cassandra cluster on Amazon EC2 with private static IPs to avoid rebalancing
- Deploy multiple-stack applications using AWS CDK with TypeScript
- Deploy SQL Server failover cluster instances on Amazon EC2 and Amazon FSx by using Terraform
- Emulate Oracle PL/SQL associative arrays in Amazon Aurora PostgreSQL and Amazon RDS for PostgreSQL
- Estimate the Amazon RDS engine size for an Oracle database by using AWR reports
- Generate data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Handle anonymous blocks in Dynamic SQL statements in Aurora PostgreSQL
- Handle overloaded Oracle functions in Aurora PostgreSQL-Compatible
- Identify duplicate container images automatically when migrating to an Amazon ECR repository
- Integrate VMware vRealize Network Insight with VMware Cloud on AWS
- Migrate Apache Cassandra workloads to Amazon Keyspaces by using AWS Glue
- Migrate your container workloads from Azure Red Hat OpenShift (ARO) to Red Hat OpenShift Service on AWS (ROSA)
- Migrate from Oracle 8i or 9i to Amazon RDS for Oracle using SharePlex and AWS DMS

- Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator
- Migrate Microsoft SQL Server Always On availability group using AWS Application Migration Service
- Migrate Oracle functions and procedures that have more than 100 arguments to PostgreSQL
- Migrate SAP HANA to AWS using SAP HSR with the same hostname
- Migrate SQL Server to AWS using distributed availability groups
- Migrate VMs to VMware Cloud on AWS by using HCX OS Assisted Migration
- Modernize mainframe online printing workloads on AWS by using Micro Focus Enterprise Server and LRS VPSX/MFI
- Modernize mainframe output management on AWS by using Rocket Enterprise Server and LRS PageCenterX
- Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS
- Manage Multi-AZ failover for EMR clusters by using Application Recovery Controller
- Send logs from VMware Cloud on AWS to Splunk by using VMware Aria Operations for Logs
- Set up a CI/CD pipeline for database migration by using Terraform
- Set up disaster recovery for Oracle JD Edwards EnterpriseOne with AWS Elastic Disaster Recovery
- Simplify private certificate management by using AWS Private CA and AWS RAM
- Transfer large-scale Db2 z/OS data to Amazon S3 in CSV files

Modernization

Topics

- Analyze and visualize software architecture in CAST Imaging
- Assess application readiness for migration to the AWS Cloud by using CAST Highlight
- Automatically archive items to Amazon S3 using DynamoDB TTL
- Build a multi-tenant serverless architecture in Amazon OpenSearch Service
- Deploy multiple-stack applications using AWS CDK with TypeScript
- Automate deployment of nested applications using AWS SAM
- Implement the serverless saga pattern by using AWS Step Functions
- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK
- Modernize ASP.NET Web Forms applications on AWS
- Run event-driven and scheduled workloads at scale with AWS Fargate
- Tenant onboarding in SaaS architecture for the silo model using C# and AWS CDK
- Decompose monoliths into microservices by using CQRS and event sourcing
- More patterns

Analyze and visualize software architecture in CAST Imaging

Arpita Sinha, Amazon Web Services

James Hurrell, Cast Software

Summary

This pattern shows how you can use CAST Imaging to navigate a complex software system visually and perform a precise analysis of the software structure. By using CAST Imaging in this way, you can make more informed decisions about your application's architecture, especially for modernization purposes.

To view your application's architecture in CAST Imaging, you must first onboard your application's source code through the CAST Console. The console then publishes your application's data to CAST Imaging, where you can visualize and navigate your application architecture layer by layer.

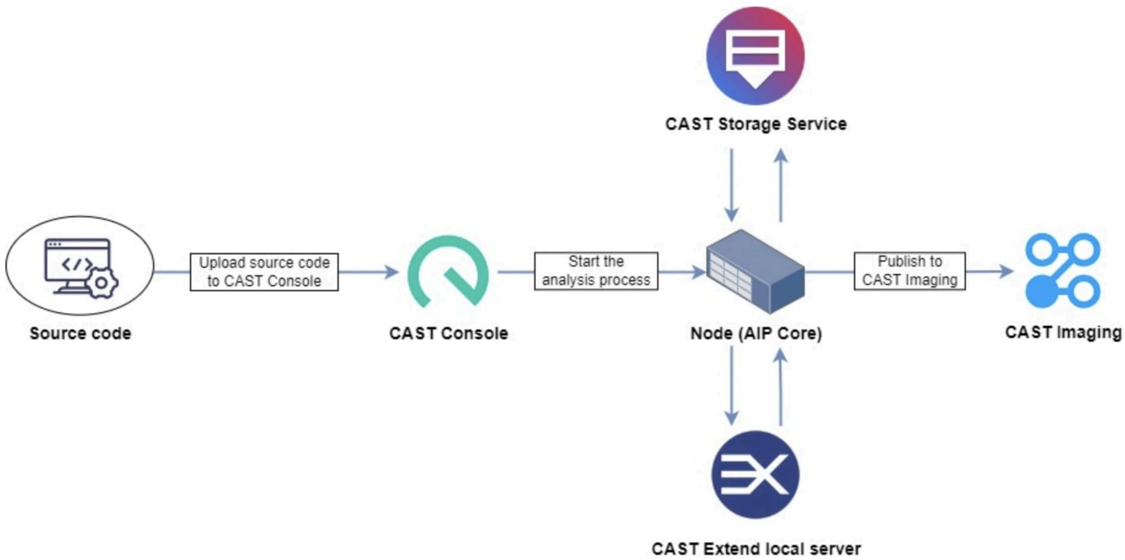
Prerequisites and limitations

Prerequisites

- An active AWS account
- The Amazon Machine Image (AMI) for CAST Imaging
- An Amazon Elastic Compute Cloud (Amazon EC2) instance that includes the following (a memory optimized **r5.xlarge** Amazon EC2 instance is recommended):
  - 4 vCPU
  - 32 GB RAM
  - 500 GB minimum General Purpose solid state drive (SSD) (gp3) volume
- CAST Console and CAST Imaging license keys (to get the required license keys, contact CAST at aws.contact-me@castsoftware.com)
- The complete source code of the application that you want to analyze in compressed (.zip) format
- Microsoft Edge, Mozilla Firefox, or Google Chrome

Architecture

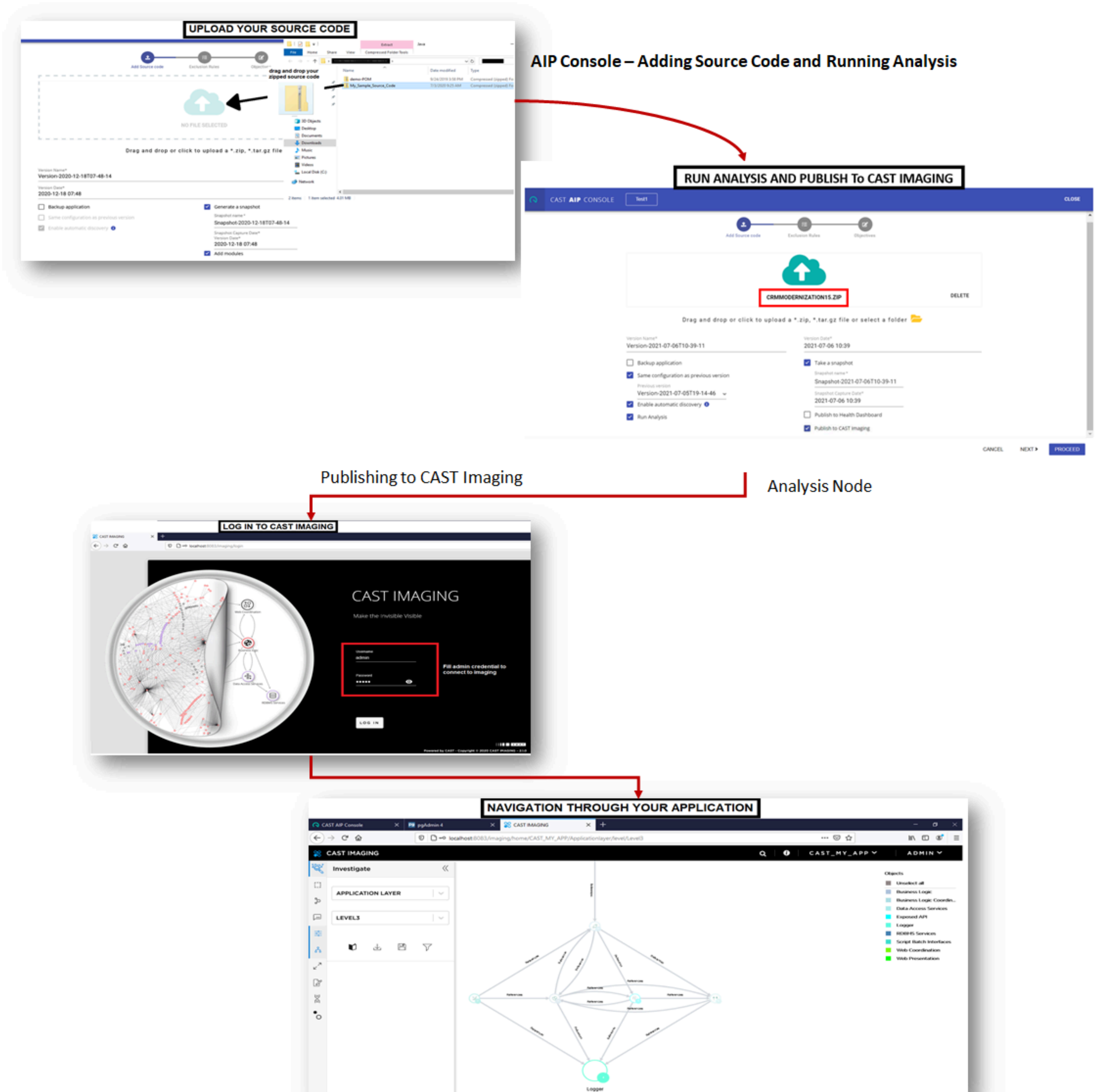
The following diagram shows an example workflow for onboarding an application's source code through the CAST Console and then viewing it in CAST Imaging:



The diagram shows the following workflow:

1. CAST generates application source code metadata by reverse-engineering front-end, middleware, and back-end code.
2. The application data that is generated by CAST is automatically imported into CAST Imaging, where it can be visualized and analyzed.

Here's a snapshot of how this process works:



CAST Imaging – Analyzing and Publishing Data, and Application visualization

Tools

- [CAST Imaging](#) is a browser-based application that helps you view and navigate your software system visually, so you can make informed decisions about its architecture.

- CAST Console is a browser-based application that helps you configure, run, and manage CAST AIP analyses.

> **Note**
>
> CAST Imaging and CAST Console are included in the AMI for CAST Imaging.

Epics

Set up the CAST Imaging environment

Task	Description	Skills required
Run the initial CAST Console configuration.	1. Open your web browser and connect to the CAST Console by entering the following URL: **http://localhost:8081** 2. When prompted, enter your **CAST Console license key**. Then, choose **Next**. 3. Review the configuration settings. If no changes are needed, choose **Save and Finish**.	Software architects, Developers, Technical leaders
Run the initial CAST Imaging configuration.	1. Open your web browser and connect to CAST Imaging by entering the following URL: **http://localhost:8083** 2. When prompted, log in by entering **admin** for both the username and password.	Software architects, Developers, Technical leaders

Task	Description	Skills required
	3. When prompted, enter your **CAST Imaging license key**. Then, choose **Update** to save the key.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Configure CAST Extend local server.</td><td>(Optional) By default, CAST Extend local server is configured to function in offline mode. If this is acceptable, no additional configuration is necessary. However, if you prefer to configure CAST Extend local server in online/proxy mode with a direct connection to CAST Extend, follow these steps.

Note
For CAST Extend credentials, see the CAST Extend registration page.

1. Use the CAST Extend Admin Center shortcut on the desktop to load your web browser and connect to CAST Extend local server.
2. Choose the Online option.
3. Enter your CAST Extend credentials (email and password), and choose Save to complete the process.</td><td>Software architects, Developers, Technical leaders</td></tr>
</table>

Onboard your application to CAST Imaging

Task	Description	Skills required
Prepare the source code for your application.	Save your application’s source code in a single, compresse d .zip file.	Software architects, Developers, Technical leaders
Add your application to the CAST Console.	1. Open your web browser and connect to the CAST Console by entering the following URL: **http://lo calhost:8081** 2. When prompted, log in by entering **admin** for both the username and password. 3. Choose **Add application**. Then, enter the application name and choose **Add**.	Software architects, Developers, Technical leaders
Open the source code delivery wizard.	Find the application that you created in the CAST Console. Then, choose **Add version**.	Software architects, Developers, Technical leaders
Upload the source code for your application.	Do one of the following: • Drag and drop the .zip file that contains your applicati on’s source code into the source code delivery wizard. −or− • Choose the **upload cloud icon**. Then, open the .zip file that contains your application’s source code.	Software architects, Developers, Technical leaders

Task	Description	Skills required
Start the analysis process.	1. In the delivery wizard, provide the version details and specify the configura tion options. For more information, see Standard onboarding for CAST Imaging in the CAST Imaging documentation. 2. Make sure that the **Publish to CAST Imaging** option is selected. Then, choose **Proceed**. **Note** Choosing **Proceed** starts the analysis process for the source code. The progress window in the CAST Console shows each step of the analysis process, and displays a notification when the analysis is complete.	Software architects, Developers, Technical leaders

Verify the analysis results and data published to CAST Imaging

Task	Description	Skills required
Check status and logs.	When all analysis actions are complete, verify that there's	Software architects, Developers, Technical leaders

Task	Description	Skills required
	a **success** message in the progress window. **Note** You can check the individual logs for each analysis action immediately after it's completed. To review the logs for a specific action, choose **View log** in the **Progress** window.	
Check the application details.	In the [Application details panel](), review the details about the analysis results. Make sure to look at the technologies that were discovered and the source code organization.	Software architects, Developers, Technical leaders

Task	Description	Skills required
Verify and access CAST Imaging.	1. In the **Application Management** pane in the CAST Console, verify that the version status of your application is [Imaging processed](). A CAST Imaging icon appears. 2. Choose the CAST Imaging icon to navigate directly to your application data in CAST Imaging. **Note** The *Imaging processed* status means that the source code has been analyzed and uploaded to your CAST Imaging instance.	Software architects, Developers, Technical leaders

Start analyzing your application with CAST Imaging

Task	Description	Skills required
Log in to CAST Imaging.	Open Cast Imaging and enter the default admin credentials (**admin/admin**). Your application's data appears.	Software architects, Developers, Technical leaders

Task	Description	Skills required
Explore your application's data in CAST Imaging.	Start viewing your software architecture by using CAST Imaging features. For a quick tutorial on how to use CAST Imaging's features, choose the **Help** icon to display the CAST Imaging Helper. For more information, see the CAST Imaging User Guide.	Software architects, Developers, Technical leaders

Related resources

CAST Console documentation

- Login
- Configuring options via CAST Console

CAST Imaging documentation

- Application onboarding for CAST Imaging - prerequisites
- Add a new application for CAST Imaging
- Standard onboarding for CAST Imaging – check results
- Login
- Configuration options – Admin Center GUI

More resources on CAST Imaging on AWS

- Application Modernization to AWS Accelerated by CAST – Technical (AWS PartnerCast webinar, requires free account)
- Using CAST and AWS Migration Hub Refactor Spaces to Modernize Legacy Applications (AWS blog post)

- Modernize Applications to AWS Architectures with CAST Imaging (AWS workshop)
- AWS Marketplace: CAST Imaging
- All CAST on AWS resources

Assess application readiness for migration to the AWS Cloud by using CAST Highlight

Greg Rivera, Amazon Web Services

Summary

CAST Highlight is a software as a service (SaaS) solution for performing rapid application portfolio analysis. This pattern describes how to configure and use CAST Highlight to assess the cloud readiness of custom software applications across an organization's IT portfolio, and to plan for modernization or migration to the Amazon Web Services (AWS) Cloud.

CAST Highlight generates insights into an application's cloud readiness, identifies code blockers that need to be removed before a migration, estimates the effort to remove these blockers, and recommends AWS services that individual applications could use after the migration.

This pattern describes the procedure for setting up and using CAST Highlight, which consists of five steps: new user setup, application management, campaign management, source code analysis, and results analysis. You must complete all the steps in the *Epics* section of this pattern to ensure a successful application scan and analysis.

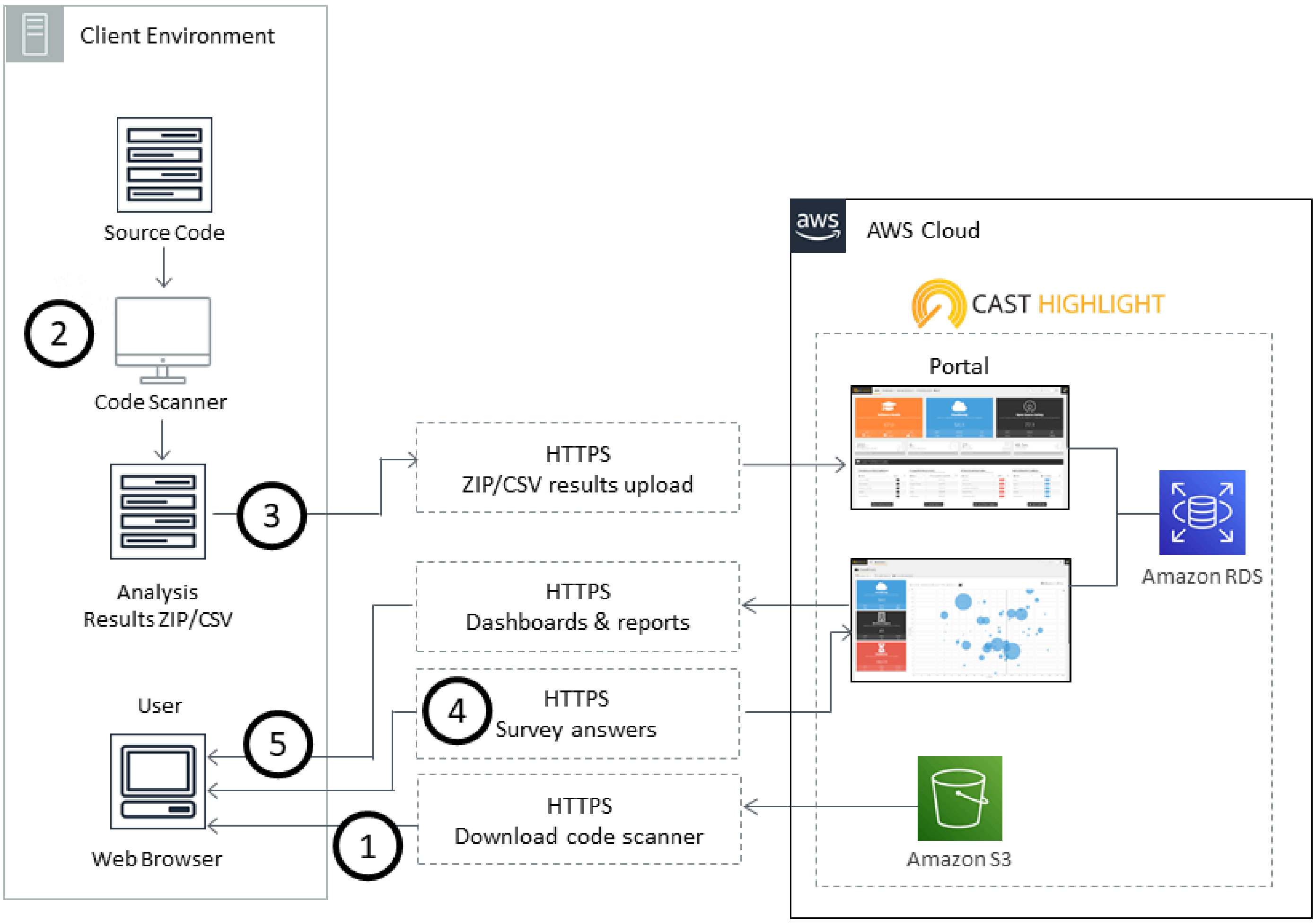
Prerequisites and limitations

Prerequisites

- An active CAST Highlight account with Portfolio Manager permissions.
- At least 300 MB free disk space and 4 GB memory on your local computer to install the CAST Highlight Local Agent.
- Microsoft Windows 8 or later.
- Your application source code must be stored in text files that are accessible from the machine where the Local Agent is installed. No source code leaves the premises and all code is locally scanned.

Architecture

The following diagram illustrates the workflow for using CAST Highlight.



The workflow consists of the following steps:

1. Log in to the CAST Highlight portal, download the Local Agent, and install it on your local computer. Amazon Simple Storage Service (Amazon S3) stores the Local Agent installation package.
2. Scan your source code files and produce a results file.
3. **Important**

   Upload the results file to the CAST Highlight portal. : No source code is included in the results file.
4. Answer survey questions for each application that you scanned.

5. View the dashboards and reports available in the CAST Highlight portal. Amazon Relational Database Service (Amazon RDS) stores the code scan, analysis results, and the CAST Highlight software data.

Technology stack

CAST Highlight supports the following technologies to analyze application cloud readiness:

- Java
- COBOL
- C#
- C++
- Clojure
- PHP
- JavaScript
- TypeScript
- Python
- Microsoft Transact-SQL
- VB.Net
- Kotlin
- Scala
- Swift

Automation and scale

- A CLI analyzer can be used to automate the CAST Highlight analysis process.

Tools

No tools are required for this pattern if all the prerequisites are met. However, you can choose to use optional tools, such as source code management (SCM) utilities, code extractors, or other tools to manage your source code files.

Epics

New user setup

Task	Description	Skills required
Activate your CAST Highlight account and choose your password.	All first-time CAST Highlight users receive an account activation email. Follow the activation link to activate your CAST Highlight account and enter a password to complete the activation process.	N/A
Log in to the CAST Highlight portal.	The CAST Highlight homepage appears after you enter your new password. Log in to the CAST Highlight portal with your user credentials.	N/A

Application management

Task	Description	Skills required
Create an application record.	In the CAST Highlight portal, navigate to the **Manage Application** tab in the **Manage Portfolio** section. In the **Applications** tile at the top of the screen, choose **Add**.	N/A
Choose an application name.	Enter a name for your application, and then choose **Save**. This name is used for	N/A

Task	Description	Skills required
	your application record in CAST Highlight.	
Repeat the steps for all applications.	Repeat these steps for each application that you want to scan.	N/A

Campaign management

Task	Description	Skills required
Create a campaign.	CAST Highlight uses "campaign" to describe a set of applications that will be analyzed at a specific time. In the CAST Highlight portal, navigate to the **Manage Campaigns** tab in the **Manage Portfolio** section. Choose **Create Campaign** to launch the campaign creation screen.	N/A
Enter a name and choose a closing date for the campaign.	Enter a name for your campaign and choose a closing date for it. **Important** Contributors cannot submit applicati on analysis results after the campaign's closing date.	N/A

Task	Description	Skills required
Decide to include source code scan, survey answers, and domain and application scope.	Choose one or more of the standard surveys that are used to enhance the source code analysis data with qualitative informati on. The survey categorie s are **Business impact**, **Software maintenan ce effort**, **CloudReady**, **Application properties**, and **Green Impact**. Choose the domain and applications that are analyzed during the campaign. **Important** Make sure that you add all applicati ons you want to scan in the **Manage Applications** section before you begin the campaign.	N/A
Customize the launch message.	Customize the launch message that will be sent by email to all contributors associated with the applicati ons in the campaign.	N/A
Launch the campaign.	Choose **Complete** to launch the campaign.	N/A

Source code analysis

Task	Description	Skills required
Download the CAST Highlight Local Agent.	In the CAST Highlight portal, choose **Application Scans** and download the Local Agent to your local computer.	N/A
Install the Local Agent.	Launch the CASTHighl ightSetup.exe installation program and follow the setup instructions that appear. After the Local Agent is installed, you are ready to analyze your applications.	N/A
Define the scope of the Local Agent code scan.	The code analysis is performed at the file level and doesn't consider the logical links or dependenc ies among files. All files are considered equal and part of the application. To provide accurate and consistent results, prepare your code scan scope by using the file or folder exclusion features available in the Local Agent.	N/A
Include open-source or COTS packages.	(Optional) If you want to include open-source or commercial off-the-shelf (COTS) packages, make sure they're included in the folders that you're planning	N/A

Task	Description	Skills required
	to scan. Typically, external libraries are grouped in a subfolder called "third-party" or something similar, and the main code is often located in the "src/main" file folder.	
Exclude test classes.	Test classes are typically excluded from the source code analysis because they are generally not part of the compiled application. However, you can choose to include them in the scan if required.	N/A
Exclude SCM, build, and deployment folders.	For more consistent results, you should avoid including SCM, build, or deploymen t folders (for example, .git or .svn files) in your scan.	N/A
Include dependency files.	If you want insights into frameworks and dependenc ies whose physical files are not part of the folder you're scanning, then make sure you include the dependency files (such as pom.xml, build.gra dle, package.json, or .vcsproj files).	N/A
Invoke the Local Agent.	Run the Local Agent on your local Windows machine.	N/A

Task	Description	Skills required
Choose the folder that contains your source code.	Choose the folder that contains your source code. You can add multiple folders to be discovered by the Local Agent. Although the Local Agent does support source discovery through network paths, you should make sure source folders are located on your local machine. **Important** We recommend running multiple scans if there are more than 10,000 files in your source folders.	N/A

Task	Description	Skills required
Start file discovery.	On the Local Agent dashboard, choose **Discover Files**. The Local Agent discovers files in your folders and subfolders, and detects their technologies. You can choose the **Cancel** button to cancel the discovery at any time. After the file discovery finishes, the Local Agent lists the folders and files that were found. The **Technologies** column shows associated technologies and file count. The **Path** column shows the location of the folders and files.	N/A

Task	Description	Skills required
Refine the source code scan configuration.	(Optional) To refine the Local Agent scan, you can deactivate one or more technologies for a specific folder or file. If all technologies are deactivated, your folder or file will be excluded from the scan's scope. To deactivate technologies, choose the yellow label of the technology you want to deactivate. You can also choose the filter icon when hovering over a file or a folder to associate a technology with a specific file or folder. These settings are saved and speed up the discovery process for the folder or file.	N/A
Start source code scan.	After you configure your scan, choose "Scan Files" to begin the scanning process.	N/A

Task	Description	Skills required
Check for green or grey labels.	After the source code scan is complete, a status label is shown at folder and file levels. A green label means that files were correctly scanned with the associated technology. A grey label means that files were not scanned and are excluded. The reason for their exclusion is shown when you hover over the label of each file. Possible reasons for file exclusion include binary files, unreadable files, missing files, external library, encoded files, generated files, syntax errors, content that isn't in the expected language, code that isn't compliant with enough analysis criteria, files that exceed the size limit (10 MB), timeout issues, or unavailability of the analyzer.	N/A
Modify the scan configuration and scan code again.	(Optional) You can modify your scan configuration settings and choose **Scan Files** to scan the files again.	N/A
Confirm scan results.	Choose **Confirm Results** if the scan results meet your requirements.	N/A

Task	Description	Skills required
View frameworks and software libraries found by the Local Agent.	View the frameworks and software libraries used or referenced by your applicati ons, and discovered by the Local Agent during the code scan. You can keep or ignore elements from these lists by choosing their individual switch button. Choose **Confirm dependenc ies** to proceed. **Important** If a framework is turned off, it is not listed in the CAST Highlight portal or attached to your application.	N/A

Task	Description	Skills required
Save code scan results.	The Local Agent displays a summary of your code scan results grouped by technology. Choose **Save** and specify the folder you want the results to be saved to. The Local Agent generates one .zip file per scan, which contains all the analysis results. Depending on the number of distinct technologies and root source folders, the Local Agent automatically generates one or several .csv files with the **FolderNam e.Technology.date.csv** naming structure.	N/A
Upload the code scan results to the CAST Highlight portal.	In the CAST Highlight portal, choose the applications you analyzed in the **Application Scans** section. Choose **Upload Results** and choose the .csv files. You can also upload the .csv files individually. After each file is uploaded, a record of the upload appears on your screen.	N/A

Task	Description	Skills required
Delete analysis result files, if required.	(Optional) An analysis results file can be deleted at any time during the upload process by choosing the trash can icon. **Important** Only users with Portfolio Manager privileges or the contributor who uploaded the results can delete the results.	N/A
Answer application survey.	A **Survey** button appears on applications that require a survey. Choose **Survey**, answer the questions for each section of the survey, and choose **Submit** after you finish. The progress of your survey is displayed at the top of your screen. You are able to submit your results after all mandatory information is submitted. However, you can enrich the data in your organization's CAST Highlight instance by answering all the questions.	N/A

Task	Description	Skills required
Submit code scan results.	After you upload all the .csv result files for the applicati on and complete the survey questions, choose **Submit** in the **Application Scans** section. This step is required to complete the process and ensure that the results are available in the CAST Highlight portal.	N/A

Results analysis

Task	Description	Skills required
View CAST Highlight portal homepage.	The CAST Highlight portal homepage includes tiles that have high-level informati on about your application portfolio, such as software health, CloudReady, and open-source safety scores for your entire portfolio. The homepage also includes the number of onboarded applications. For more information about the CAST Highlight metrics definitions and measurement methodolo gy, see [CAST Highlight – Metrics and methodolo gy (Microsoft PowerPoint presentation).]()	N/A

Task	Description	Skills required
View the CloudReady dashboard.	Choose the **CloudReady** tile to open the CloudReady dashboard. This is the primary portfolio-level dashboard for assessing the cloud readiness of your applications. It helps you plan and develop a portfolio roadmap for your cloud migration	N/A

Task	Description	Skills required
View the Portfolio Advisor for Cloud dashboard.	The Portfolio Advisor for Cloud dashboard automatically segments applications into recommended migration categories. Segmentation is based on the technical characteristics of each application. Factors include the source code analysis (cloud readiness, software resiliency, and more) and the business impact, which comes from the survey. In the upper right, choose **Compute** to generate the initial segmentation recommendations. The bubbles in the charts at the top of the dashboard represent each application in the portfolio, organized by the recommended segmentation. Each application is also listed in a data table below the charts, including relevant metrics for each application. The possible segments that are recommended include: • **Rehost** – A recommendation to change the infrastructure configuration of the application in order	N/A

Task	Description	Skills required
	to *lift and shift* it to the cloud by using an infrastructure as a service (IaaS) solution. • **Refactor** – A recommendation to perform modest modifications of the application code without changing the architecture or functionality so that it can be migrated by using a container as a service (CaaS) or platform as a service (PaaS) solution. • **Rearchitect** – A recommendation to dramatically modify the application code to improve the health of the application and to prepare it for migration by using a PaaS solution or deploy it as a serverless application by using a function as a service (FaaS) solution. • **Rebuild** – A recommendation to discard the code of the application and develop it again in the cloud by using a PaaS solution or develop it again as a serverless application by using a FaaS solution.	

Task	Description	Skills required
	• **Retire –** A recommendation to discard the application altogether or potentially replace it with a commercia l software as a service (SaaS) alternative.	
Modify segmentation recommendations.	In some cases, you might choose to change the segment recommended by CAST Highlight. You can do this by browsing to the application in the data table and selecting a different segment from the drop-down list next to the application name. Then choose **Save** in the upper right to save your changes. You can also export this data at any time by choosing **Export** in the upper right.	N/A

Task	Description	Skills required
Choose an application to analyze.	On the Portfolio Advisor for Cloud dashboard, choose an application bubble to analyze that application. Choose the name of the application in the table after the bubble chart to begin the deeper analysis. Different dashboards are available to analyze individua l applications, such as **Code Insights** (software health patterns), **Trends**, and **Software Composition** (open-source risks).	N/A

Task	Description	Skills required
Analyze the CloudRead y results of an individual application.	Choose the **CloudReady** tab, which shows the application's overall CloudReady score. This score is a weighted average based on a combination of the CloudReady survey answers and the CloudRead y code scan. The answers to the survey questions appear in the table below the tiles. Choose **CloudReady Code Scan** to view the code scan results. There is a list of CloudReady patterns that the application code was scanned for. This list includes the following columns: • **Cloud Requirement** is the specific code pattern. • **Technology** is the pattern's programming language. "Impact" is the pattern's impact on the application (C = code, F = framework, A = architecture). • **Criticality** is the level of importance of addressin g this pattern before migrating. • **Contribution** is how this pattern contributes to the overall CloudReady score.	N/A

Task	Description	Skills required
	If the pattern is green, it is a booster and increases the CloudReady score. If the pattern is red, it is a blocker and decreases the CloudReady score. If the pattern has no color, it is a blocker that was not detected and increases the CloudReady score. • **Roadblocks** are the number of individual occurrenc es of a blocker pattern. Choose the roadblock number to show a list of the source code files where the pattern was detected. • **Est. Effort** is an estimate of the number of days it will take to remediate the roadblocks in each row.	
Export data to Microsoft Excel.	(Optional) Choose **Export to Excel** to export the data for further analysis. The applicati on analysis results data can be used to further analyze the cloud readiness of an application and determine what code must to be updated before a migration.	N/A

Task	Description	Skills required
View recommendations.	Choose **Recommend ations** next to **CloudRead y Code Scan** to view the **Cloud Service Recommend ations** screen. This identifie s AWS services that the application could adopt based on its characteristics. Repeat this step to view recommendations for all the applications that you analyzed.	N/A

Related resources

Campaign management

- CAST Highlight Foundation Certification Training Section 3: Portfolio Configuration (video)

Source code analysis

- CAST Highlight Foundation Certification Training Section 4: Application Analysis (video)

Other resources

- CAST Highlight in AWS Marketplace
- AWS and CAST: Accelerate application modernization
- CAST Highlight – Documentation, product tutorials, and third-party tools
- CAST Highlight – Cloud Readiness Product Demo (video)
- Application Portfolio Modernization with CAST Highlight (AWS workshop)

Automatically archive items to Amazon S3 using DynamoDB TTL

Tabby Ward, Amazon Web Services

Summary

This pattern provides steps to remove older data from an Amazon DynamoDB table and archive it to an Amazon Simple Storage Service (Amazon S3) bucket on Amazon Web Services (AWS) without having to manage a fleet of servers.

This pattern uses Amazon DynamoDB Time to Live (TTL) to automatically delete old items and Amazon DynamoDB Streams to capture the TTL-expired items. It then connects DynamoDB Streams to AWS Lambda, which runs the code without provisioning or managing any servers.

When new items are added to the DynamoDB stream, the Lambda function is initiated and writes the data to an Amazon Data Firehose delivery stream. Firehose provides a simple, fully managed solution to load the data as an archive into Amazon S3.

DynamoDB is often used to store time series data, such as webpage click-stream data or Internet of Things (IoT) data from sensors and connected devices. Rather than deleting less frequently accessed items, many customers want to archive them for auditing purposes. TTL simplifies this archiving by automatically deleting items based on the timestamp attribute.

Items deleted by TTL can be identified in DynamoDB Streams, which captures a time-ordered sequence of item-level modifications and stores the sequence in a log for up to 24 hours. This data can be consumed by a Lambda function and archived in an Amazon S3 bucket to reduce the storage cost. To further reduce the costs, Amazon S3 lifecycle rules can be created to automatically transition the data (as soon as it gets created) to lowest-cost storage classes, such as S3 Glacier Instant Retrieval or S3 Glacier Flexible Retrieval, or Amazon S3 Glacier Deep Archive for long-term storage.

Prerequisites and limitations

Prerequisites

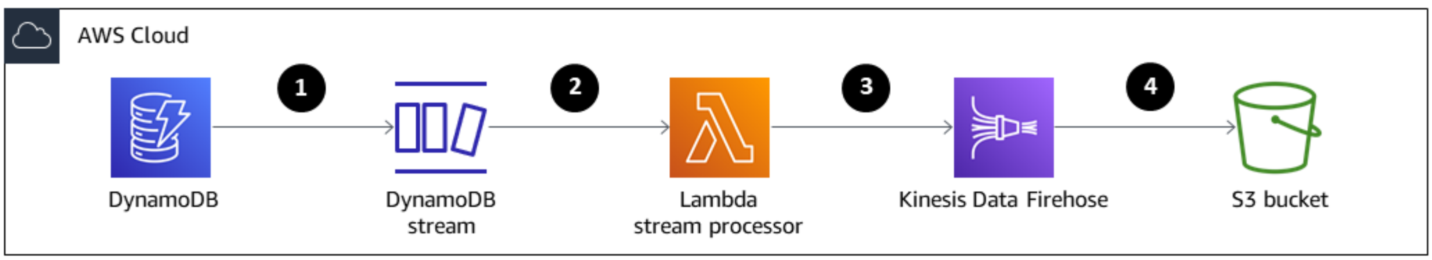
- An active AWS account.
- AWS Command Line Interface (AWS CLI) 1.7 or later, installed and configured on macOS, Linux, or Windows.
- Python 3.7 or later.

- [Boto3](), installed and configured. If Boto3 is not already installed, run the `python -m pip install boto3` command to install it.

Architecture

Technology stack

- Amazon DynamoDB
- Amazon DynamoDB Streams
- Amazon Data Firehose
- AWS Lambda
- Amazon S3



1. Items are deleted by TTL.
2. The DynamoDB stream trigger invokes the Lambda stream processor function.
3. The Lambda function puts records in the Firehose delivery stream in batch format.
4. Data records are archived in the S3 bucket.

Tools

- AWS CLI – The AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services.
- Amazon DynamoDB – Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale.
- Amazon DynamoDB Time to Live (TTL) – Amazon DynamoDB TTL helps you define a per-item timestamp to determine when an item is no longer required.

- Amazon DynamoDB Streams – Amazon DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table and stores this information in a log for up to 24 hours.
- Amazon Data Firehose – Amazon Data Firehose is the easiest way to reliably load streaming data into data lakes, data stores, and analytics services.
- AWS Lambda – AWS Lambda runs code without the need to provision or manage servers. You pay only for the compute time you consume.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Code

The code for this pattern is available in the GitHub Archive items to S3 using DynamoDB TTL repository.

Epics

Set up a DynamoDB table, TTL , and a DynamoDB stream

Task	Description	Skills required

Create a DynamoDB table.	Use the AWS CLI to create a table in DynamoDB called `Reservation` . Choose random read capacity unit (RCU) and write capacity unit (WCU), and give your table two attributes: `ReservationID` and `ReservationDate` .
	
	`aws dynamodb create-table \`
	`--table-name Reservation \`
	`--attribute-definitions AttributeName=ReservationID`
		Cloud architect, App developer

Task	Description	Skills required
	``` ,AttributeType=S AttributeName=Rese rvationDate,Attrib uteType=N \ --key-schema Attribute Name=ReservationID ,KeyType=HASH AttributeName=Rese rvationDate,KeyTyp e=RANGE \ --provisioned-th roughput ReadCapac ityUnits=100,Write CapacityUnits=100 ``` `ReservationDate` is an epoch timestamp that will be used to turn on TTL.	
Turn on DynamoDB TTL.	Use the AWS CLI to turn on DynamoDB TTL for the `ReservationDate` attribute. ``` aws dynamodb update-ti me-to-live \ --table-name Reservati on\ --time-to-live-spe cification Enabled=t rue,AttributeName= ReservationDate ```	Cloud architect, App developer

Task	Description	Skills required
Turn on a DynamoDB stream.	Use the AWS CLI to turn on a DynamoDB stream for the `Reservation` table by using the `NEW_AND_OLD_IMAGES` stream type. `aws dynamodb update-table \ --table-name Reservation \ --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES` This stream will contain records for new items, updated items, deleted items, and items that are deleted by TTL. The records for items that are deleted by TTL contain an additional metadata attribute to distinguish them from items that were deleted manually. The `userIdentity` field for TTL deletions indicates that the DynamoDB service performed the delete action. In this pattern, only the items deleted by TTL are archived, but you could archive only the records where `eventName` is `REMOVE` and `userIdent`	Cloud architect, App developer

Task	Description	Skills required
	`ity` contains `principal Id` equal to `dynamodb. amazonaws.com` .	

Create and configure an S3 bucket

Task	Description	Skills required
Create an S3 bucket.	Use the AWS CLI to create a destination S3 bucket in your AWS Region, replacing `us-east-1` with your Region and amzn-s3-demo-destination-bucket with the name of your bucket. `aws s3api create-bucket \ --bucket amzn-s3-demo-destination-bucket \ --region us-east-1` Make sure that your S3 bucket's name is globally unique, because the namespace is shared by all AWS accounts.	Cloud architect, App developer
Create a 30-day lifecycle policy for the S3 bucket.	1. Sign in to the AWS Management Console and open the Amazon S3 console. 2. Choose the S3 bucket that contains the data from Firehose.	Cloud architect, App developer

Task	Description	Skills required
	3. In the S3 bucket, choose the **Management** tab, and choose **Add lifecycle rule**. 4. Enter a name for your rule in the **Lifecycle rule** dialog box, and configure a 30-day lifecycle rule for your bucket.	

Create a Firehose delivery stream

Task	Description	Skills required
Create and configure a Firehose delivery stream.	Download and edit the `CreateFireHoseToS3.py` code example from the GitHub repository. This code is written in Python and shows you how to create a Firehose delivery stream and an AWS Identity and Access Management (IAM) role. The IAM role will have a policy that can be used by Firehose to write to the destination S3 bucket. To run the script, use the following command and command line arguments. Argument 1= `<Your_S3_bucket_ARN>` , which is the Amazon Resource Name	Cloud architect, App developer

Task	Description	Skills required
	(ARN) for the bucket that you created earlier Argument 2= Your Firehose name (This pilot is using `firehose_to_s3_stream` .) Argument 3= Your IAM role name (This pilot is using `firehose_to_s3` .) `python CreateFireHoseToS3.py <Your_S3_Bucket_ARN> firehose_to_s3_stream firehose_to_s3` If the specified IAM role does not exist, the script will create an assume role with a trusted relationship policy, as well as a policy that grants sufficient Amazon S3 permission. For examples of these policies, see the *Additional information* section.	

Task	Description	Skills required
Verify the Firehose delivery stream.	Describe the Firehose delivery stream by using the AWS CLI to verify that the delivery stream was successfully created. `aws firehose describe-delivery-stream --delivery-stream-name firehose_to_s3_stream`	Cloud architect, App developer

Create a Lambda function to process the Firehose delivery stream

Task	Description	Skills required
Create a trust policy for the Lambda function.	Create a trust policy file with the following information. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" } ] }`	Cloud architect, App developer

Task	Description	Skills required
	This gives your function permission to access AWS resources.	
Create an execution role for the Lambda function.	To create the execution role, run the following code. `aws iam create-role --role-name lambda-ex --assume-role-policy-document file://TrustPolicy.json`	Cloud architect, App developer

Task	Description	Skills required

Add permission to the role.	To add permission to the role, use the `attach-policy-to-role` command.
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaDynamoDBExecutionRole`
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/AmazonKinesisFirehoseFullAccess`
	`aws iam attach-role-policy --role-name lambda-ex --policy-arn arn:aws:iam::aws:policy/IAMFullAccess`
		Cloud architect, App developer

Task	Description	Skills required

Create a Lambda function.	Compress the `LambdaStreamProcessor.py` file from the code repository by running the following command.
	`zip function.zip LambdaStreamProcessor.py`
	When you create the Lambda function, you will need the Lambda execution role ARN. To get the ARN, run the following code.
	`aws iam get-role \ --role-name lambda-ex`
	To create the Lambda function, run the following code.
	`# Review the environment variables and replace them with your values.`
	`aws lambda create-function --function-name LambdaStreamProcessor \ --zip-file fileb://function.zip --handler LambdaStreamProcessor.handler --runtime python3.8 \`
		Cloud architect, App developer

Task	Description	Skills required
	`--role {Your Lamda Execution Role ARN}\ --environment Variables="{firehose_name=firehose_to_s3_stream,bucket_arn = <Your_S3_bucket_ARN>,iam_role_name = firehose_to_s3, batch_size=400}"`	
Configure the Lambda function trigger.	Use the AWS CLI to configure the trigger (DynamoDB Streams), which invokes the Lambda function. The batch size of 400 is to avoid running into Lambda concurrency issues. `aws lambda create-event-source-mapping --function-name LambdaStreamProcessor \ --batch-size 400 --starting-position LATEST \ --event-source-arn <Your Latest Stream ARN From DynamoDB Console>`	Cloud architect, App developer

Test the functionality

Task	Description	Skills required
Add items with expired timestamps to the Reservation table.	To test the functionality, add items with expired epoch timestamps to the `Reservation` table. TTL will automatically delete items based on the timestamp. The Lambda function is initiated upon DynamoDB Stream activities, and it filters the event to identify `REMOVE` activity or deleted items. It then puts records in the Firehose delivery stream in batch format. The Firehose delivery stream transfers items to a destination S3 bucket with the `firehosetos3example/year=current year/month=current month/day=current day/hour=current hour/` prefix. **Important** To optimize data retrieval, configure Amazon S3 with the `Prefix` and `ErrorOutputPrefix` that	Cloud architect

Task	Description	Skills required
	are detailed in the *Additional information* section.	

Clean up the resources

Task	Description	Skills required
Delete all resources.	Delete all the resources to ensure that you aren't charged for any services that you aren't using.	Cloud architect, App developer

Related resources

- Managing your storage lifecycle
- Amazon S3 Storage Classes
- AWS SDK for Python (Boto3) documentation

Additional information

Create and configure a Firehose delivery stream – Policy examples

Firehose trusted relationship policy example document

```
firehose_assume_role = {
        'Version': '2012-10-17',
        'Statement': [
            {
                'Sid': '',
                'Effect': 'Allow',
                'Principal': {
                    'Service': 'firehose.amazonaws.com'
                },
```

```
                'Action': 'sts:AssumeRole'
            }
        ]
    }
```

S3 permissions policy example

```
s3_access = {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "",
                "Effect": "Allow",
                "Action": [
                    "s3:AbortMultipartUpload",
                    "s3:GetBucketLocation",
                    "s3:GetObject",
                    "s3:ListBucket",
                    "s3:ListBucketMultipartUploads",
                    "s3:PutObject"
                ],
                "Resource": [
                    "{your s3_bucket ARN}/*",
                    "{Your s3 bucket ARN}"
                ]
            }
        ]
    }
```

Test the functionality – Amazon S3 configuration

The Amazon S3 configuration with the following `Prefix` and `ErrorOutputPrefix` is chosen to optimize data retrieval.

Prefix

```
firehosetos3example/year=! {timestamp: yyyy}/month=! {timestamp:MM}/day=!
 {timestamp:dd}/hour=!{timestamp:HH}/
```

Firehose first creates a base folder called `firehosetos3example` directly under the S3 bucket. It then evaluates the expressions `!{timestamp:yyyy}`, `!{timestamp:MM}`, `!{timestamp:dd}`, and `!{timestamp:HH}` to year, month, day, and hour using the Java [DateTimeFormatter](https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html) format.

For example, an approximate arrival timestamp of 1604683577 in Unix epoch time evaluates to `year=2020`, `month=11`, `day=06`, and `hour=05`. Therefore, the location in Amazon S3, where data records are delivered, evaluates to `firehosetos3example/year=2020/month=11/day=06/hour=05/`.

ErrorOutputPrefix

```
firehosetos3erroroutputbase/!{firehose:random-string}/!{firehose:error-output-type}/!
{timestamp:yyyy/MM/dd}/
```

The `ErrorOutputPrefix` results in a base folder called `firehosetos3erroroutputbase` directly under the S3 bucket. The expression `!{firehose:random-string}` evaluates to an 11-character random string such as `ztWxkdg3Thg`. The location for an Amazon S3 object where failed records are delivered could evaluate to `firehosetos3erroroutputbase/ztWxkdg3Thg/processing-failed/2020/11/06/`.

Build a multi-tenant serverless architecture in Amazon OpenSearch Service

Tabby Ward and Nisha Gambhir, Amazon Web Services

Summary

Amazon OpenSearch Service is a managed service that makes it easy to deploy, operate, and scale Elasticsearch, which is a popular open-source search and analytics engine. OpenSearch Service provides free-text search as well as near real-time ingestion and dashboarding for streaming data such as logs and metrics.

Software as a service (SaaS) providers frequently use OpenSearch Service to address a broad range of use cases, such as gaining customer insights in a scalable and secure way while reducing complexity and downtime.

Using OpenSearch Service in a multi-tenant environment introduces a series of considerations that affect partitioning, isolation, deployment, and management of your SaaS solution. SaaS providers have to consider how to effectively scale their Elasticsearch clusters with continually shifting workloads. They also need to consider how tiering and noisy neighbor conditions could impact their partitioning model.

This pattern reviews the models that are used to represent and isolate tenant data with Elasticsearch constructs. In addition, the pattern focuses on a simple serverless reference architecture as an example to demonstrate indexing and searching using OpenSearch Service in a multi-tenant environment. It implements the pool data partitioning model, which shares the same index among all tenants while maintaining a tenant's data isolation. This pattern uses the following AWS services: Amazon API Gateway, AWS Lambda, Amazon Simple Storage Service (Amazon S3), and OpenSearch Service.

For more information about the pool model and other data partitioning models, see the [Additional information](#) section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- [AWS Command Line Interface (AWS CLI) version 2.x](#), installed and configured on macOS, Linux, or Windows

- Python version 3.9
- pip3 – The Python source code is provided as a .zip file to be deployed in a Lambda function. If you want to use the code locally or customize it, follow these steps to develop and recompile the source code:
  1. Generate the `requirements.txt` file by running the the following command in the same directory as the Python scripts: `pip3 freeze > requirements.txt`
  2. Install the dependencies: `pip3 install -r requirements.txt`

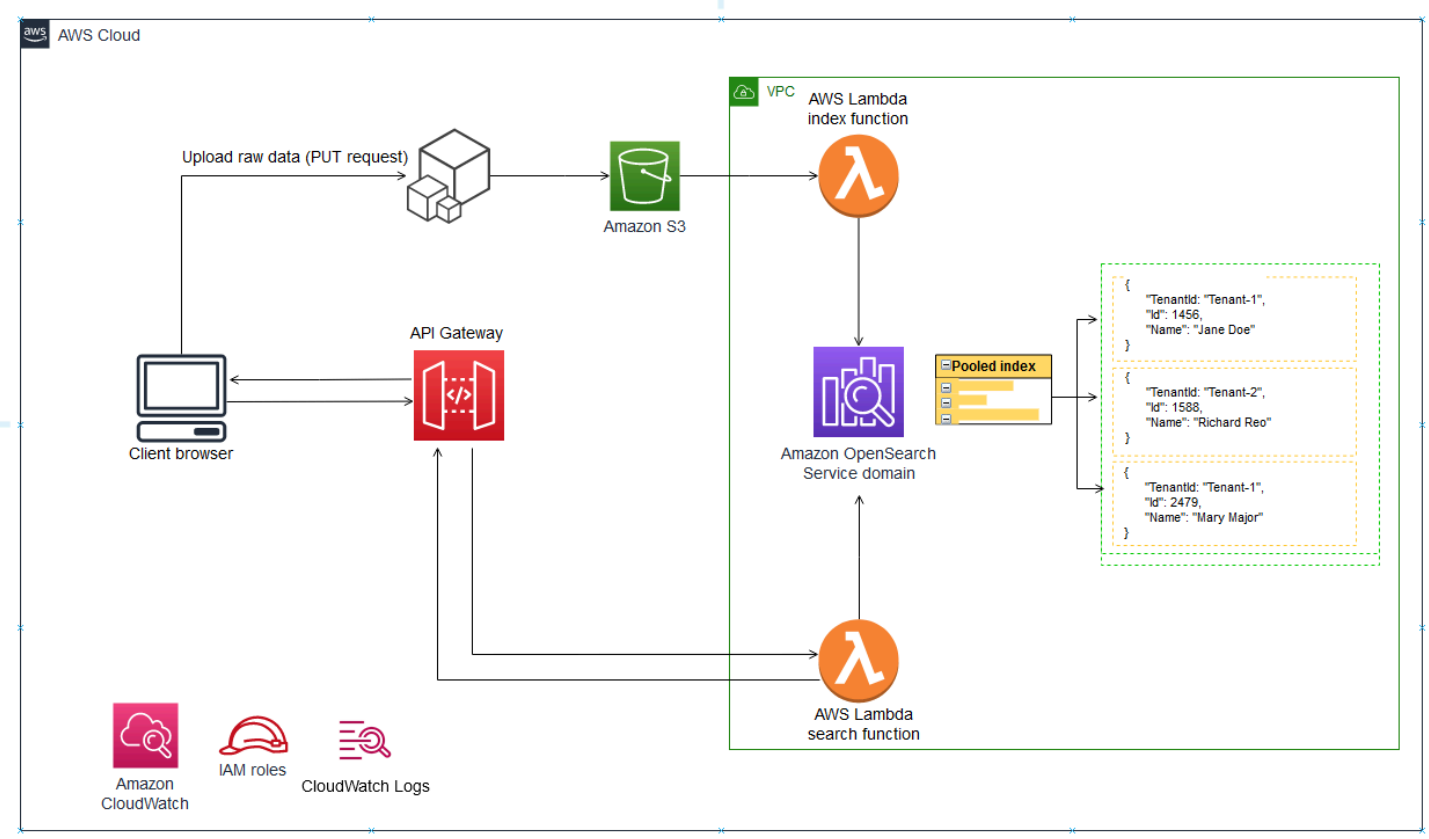
Limitations

- This code runs in Python, and doesn't currently support other programming languages.
- The sample application doesn't include AWS cross-Region or disaster recovery (DR) support.
- This pattern is intended for demonstration purposes only. It is not intended to be used in a production environment.

Architecture

The following diagram illustrates the high-level architecture of this pattern. The architecture includes the following:

- Lambda to index and query the content
- OpenSearch Service to perform search
- API Gateway to provide an API interaction with the user
- Amazon S3 to store raw (non-indexed) data
- Amazon CloudWatch to monitor logs
- AWS Identity and Access Management (IAM) to create tenant roles and policies



Automation and scale

For simplicity, the pattern uses AWS CLI to provision the infrastructure and to deploy the sample code. You can create an AWS CloudFormation template or AWS Cloud Development Kit (AWS CDK) scripts to automate the pattern.

Tools

AWS services

- [AWS CLI](#) is a unified tool for managing AWS services and resources by using commands in your command-line shell.
- [Lambda](#) is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- [API Gateway](#) is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- [Amazon S3](#) is an object storage service that lets you store and retrieve any amount of information at any time, from anywhere on the web.

- [OpenSearch Service](https://docs.aws.amazon.com/opensearch-service/) is a fully managed service that makes it easy for you to deploy, secure, and run Elasticsearch cost-effectively at scale.

Code

The attachment provides sample files for this pattern. These include:

- `index_lambda_package.zip` – The Lambda function for indexing data in OpenSearch Service by using the pool model.
- `search_lambda_package.zip` – The Lambda function for searching for data in OpenSearch Service.
- `Tenant-1-data` – Sample raw (non-indexed) data for Tenant-1.
- `Tenant-2-data` – Sample raw (non-indexed) data for Tenant-2.

> **Important**
>
> The stories in this pattern include AWS CLI command examples that are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) Unix continuation character at the end of each line with a caret (^).

> **Note**
>
> In AWS CLI commands, replace all values within the angle brackets (<>) with correct values.

Epics

Create and configure an S3 bucket

Task	Description	Skills required
Create an S3 bucket.	Create an S3 bucket in your AWS Region. This bucket will hold the non-indexed tenant data for the sample	Cloud architect, Cloud administrator

Task	Description	Skills required
	application. Make sure that the S3 bucket's name is globally unique, because the namespace is shared by all AWS accounts. To create an S3 bucket, you can use the AWS CLI [create-bucket](#) command as follows: `aws s3api create-bucket \ --bucket <tenantrawdata> \ --region <your-AWS-Region>` where `tenantrawdata` is the S3 bucket name. (You can use any unique name that follows [the bucket naming guidelines](#).)	

Create and configure an Elasticsearch cluster

Task	Description	Skills required
Create an OpenSearch Service domain.	Run the AWS CLI [create-elasticsearch-domain](#) command to create an OpenSearch Service domain: `aws es create-elasticsearch-domain \ --domain-name vpc-cli-example \`	Cloud architect, Cloud administrator

Task	Description	Skills required
	`--elasticsearch-ve rsion 7.10 \ --elasticsearch-cl uster-config InstanceT ype=t3.medium.elas ticsearch,Instance Count=1 \ --ebs-options EBSEnabled=true,Vo lumeType=gp2,Volum eSize=10 \ --domain-endpoint- options "{\"Enfor ceHTTPS\": true}" \ --encryption-at-re st-options "{\"Enabl ed\": true}" \ --node-to-node- encryption-options "{\"Enabled\": true}" \ --advanced-securit y-options "{\"Enabl ed\": true, \"Interna lUserDatabaseEnabled \": true, \ \"MasterUserOption s\": {\"MasterUserName \": \"KibanaUser\", \ \"MasterUserPasswo rd\": \"NewKiba naPassword@123\"}}" \ --vpc-options "{\"SubnetIds\": [\"<subnet-id>\"], \"SecurityGroupIds\": [\"<sg-id>\"]}" \ --access-policies "{\"Version\": \"2012-10-17\", \"Statement\":`	

Task	Description	Skills required

	``
	 [{ \"Effect\":
	 \"Allow\", \
	   \"Principal\":
	 {\"AWS\": \"*\" },
	 \"Action\":\"es:*\",
	 \
	   \"Resource\":
	 \"arn:aws:es:<regi
	on>:<account-id>:d
	omain\/vpc-cli-exa
	mple\/*\" }] }"
	``
	
	The instance count is set to 1 because the domain is for testing purposes. You need to enable fine-grained access control by using the `advanced-security-options` parameter, because the details cannot be changed after the domain has been created.
	
	This command creates a master user name (`KibanaUser`) and a password that you can use to log in to the Kibana console.
	
	Because the domain is part of a virtual private cloud (VPC), you have to make sure that you can reach the Elasticsearch instance by specifying the access policy to use.	

Task	Description	Skills required
	For more information, see Launching your Amazon OpenSearch Service domains within a VPC in the AWS documentation.	

Task	Description	Skills required
Set up a bastion host.	Set up a Amazon Elastic Compute Cloud (Amazon EC2) Windows instance as a bastion host to access the Kibana console. The Elasticsearch security group must allow traffic from the Amazon EC2 security group. For instructions, see the blog post [Controlling Network Access to EC2 Instances Using a Bastion Server](). When the bastion host has been set up, and you have the security group that is associated with the instance available, use the AWS CLI [authorize-security-group-ingress]() command to add permission to the Elasticsearch security group to allow port 443 from the Amazon EC2 (bastion host) security group. `aws ec2 authorize-security-group-ingress \ --group-id <SecurityGroupIdfElasticSearch> \ --protocol tcp \ --port 443 \`	Cloud architect, Cloud administrator

Task	Description	Skills required
	``` --source-group <SecurityGroupIdfB ashionHostEC2> ```	

Create and configure the Lambda index function

Task	Description	Skills required

Create the Lambda execution role.	Run the AWS CLI [create-role](https://docs.aws.amazon.com/cli/latest/reference/iam/create-role.html) command to grant the Lambda index function access to AWS services and resources:
	
	``
	aws iam create-role \
	  --role-name index-lambda-role \
	  --assume-role-policy-document file://lambda_assume_role.json
	``
	
	where `lambda_assume_role.json` is a JSON document that grants `AssumeRole` permissions to the Lambda function, as follows:
	
	``
	{
	     "Version": "2012-10-17",
	     "Statement": [
	         {
	             "Effect": "Allow",
	             "Principal": {
	``
		Cloud architect, Cloud administrator

Task	Description	Skills required

	``
	 "Service": "lambda.a
	mazonaws.com"
	                },
	              "Action":
	 "sts:AssumeRole"
	         }
	     ]
	 }
	``	

Task	Description	Skills required

Attach managed policies to the Lambda role.	Run the AWS CLI attach-role-policy command to attach managed policies to the role created in the previous step. These two policies give the role permissions to create an elastic network interface and to write logs to CloudWatch Logs.
	
	`aws iam attach-role-policy \ --role-name index-lambda-role \ --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	
	`aws iam attach-role-policy \ --role-name index-lambda-role \ --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole`
		Cloud architect, Cloud administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr>
<td>Create a policy to give the Lambda index function permission to read the S3 objects.</td>
<td>

Run the AWS CLI [create-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/create-policy.html) command to to give the Lambda index function `s3:GetObject` permission to read the objects in the S3 bucket:

```
aws iam create-policy \
  --policy-name s3-permission-policy \
  --policy-document file://s3-policy.json
```

The file `s3-policy.json` is a JSON document shown below that grants `s3:GetObject` permissions to allow read access to S3 objects. If you used a different name when you created the S3 bucket, provide the correct bucket name in the `Resource` section:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<tenantrawdata>/*"
        }
```

</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</table>

Task	Description	Skills required

	``
	    ]
	}
	``	

Attach the Amazon S3 permission policy to the Lambda execution role.	Run the AWS CLI [attach-role-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html) command to attach the Amazon S3 permission policy you created in the previous step to the Lambda execution role:
	``
	aws iam attach-role-policy \
	  --role-name index-lambda-role \
	  --policy-arn <PolicyARN>
	``
	where `PolicyARN` is the Amazon Resource Name (ARN) of the Amazon S3 permission policy. You can get this value from the output of the previous command.
		Cloud architect, Cloud administrator

Task	Description	Skills required

Create the Lambda index function.	Run the AWS CLI [create-function](https://docs.aws.amazon.com/cli/latest/reference/lambda/create-function.html) command to create the Lambda index function, which will access OpenSearch Service:
	
	`aws lambda create-function \`
	`  --function-name index-lambda-function \`
	`  --zip-file fileb://index_lambda_package.zip \`
	`  --handler lambda_index.lambda_handler \`
	`  --runtime python3.9 \`
	`  --role "arn:aws:iam::account-id:role/index-lambda-role" \`
	`  --timeout 30 \`
	`  --vpc-config "{\"SubnetIds\": [\"<subnet-id1\>", \"<subnet-id2>\"], \`
	`    \"SecurityGroupIds\": [\"<sg-1>\"]}"`
		Cloud architect, Cloud administrator

Task	Description	Skills required
Allow Amazon S3 to call the Lambda index function.	Run the AWS CLI [add-permission](https://docs.aws.amazon.com/cli/latest/reference/lambda/add-permission.html) command to give Amazon S3 the permission to call the Lambda index function: `aws lambda add-permission \ --function-name index-lambda-function \ --statement-id s3-permissions \ --action lambda:InvokeFunction \ --principal s3.amazonaws.com \ --source-arn "arn:aws:s3:::<tenantrawdata>" \ --source-account "<account-id>"`	Cloud architect, Cloud administrator

Task	Description	Skills required
Add a Lambda trigger for the Amazon S3 event.	Run the AWS CLI put-bucket-notification-configuration command to send notifications to the Lambda index function when the Amazon S3 `ObjectCreated` event is detected. The index function runs whenever an object is uploaded to the S3 bucket. `aws s3api put-bucket-notification-configuration \ --bucket <tenantrawdata> \ --notification-configuration file://s3-trigger.json` The file `s3-trigger.json` is a JSON document in the current folder that adds the resource policy to the Lambda function when the Amazon S3 `ObjectCreated` event occurs.	Cloud architect, Cloud administrator

Create and configure the Lambda search function

Task	Description	Skills required
Create the Lambda execution role.	Run the AWS CLI create-role command to grant the Lambda search function	Cloud architect, Cloud administrator

Task	Description	Skills required

	access to AWS services and resources:
	```aws iam create-role \ --role-name search-lambda-role \ --assume-role-policy-document file://lambda_assume_role.json```
	 where `lambda_assume_role.json` is a JSON document in the current folder that grants `AssumeRole` permissions to the Lambda function, as follows:
	```{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Principal": { "Service": "lambda.amazonaws.com" }, "Action": "sts:AssumeRole" }] }```	

Task	Description	Skills required

Attach managed policies to the Lambda role.	Run the AWS CLI [attach-role-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html) command to attach managed policies to the role created in the previous step. These two policies give the role permissions to create an elastic network interface and to write logs to CloudWatch Logs.
	`aws iam attach-role-policy \ --role-name search-lambda-role \ --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole`
	
	`aws iam attach-role-policy \ --role-name search-lambda-role \ --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole`
		Cloud architect, Cloud administrator

Task	Description	Skills required
Create the Lambda search function.	Run the AWS CLI create-function command to create the Lambda search function, which will access OpenSearch Service: `aws lambda create-function \ --function-name search-lambda-function \ --zip-file fileb://search_lambda_package.zip \ --handler lambda_search.lambda_handler \ --runtime python3.9 \ --role "arn:aws:iam::account-id:role/search-lambda-role" \ --timeout 30 \ --vpc-config "{\"SubnetIds\": [\"<subnet-id1\>", \"<subnet-id2>\"], \ \"SecurityGroupIds\": [\"<sg-1>\"]}"`	Cloud architect, Cloud administrator

```
aws lambda create-function \
  --function-name search-lambda-function \
  --zip-file fileb://search_lambda_package.zip \
  --handler lambda_search.lambda_handler \
  --runtime python3.9 \
  --role "arn:aws:iam::account-id:role/search-lambda-role" \
  --timeout 30 \
  --vpc-config "{\"SubnetIds\": [\"<subnet-id1\>", \"<subnet-id2>\"], \
    \"SecurityGroupIds\": [\"<sg-1>\"]}"
```

Create and configure tenant roles

Task	Description	Skills required
Create tenant IAM roles.	Run the AWS CLI create-role command to create two tenant roles that will be used to test the search functionality:	Cloud architect, Cloud administrator

Task	Description	Skills required

	<pre>aws iam create-role \
	  --role-name Tenant-1-
	role \
	  --assume-role-poli
	cy-document file://as
	sume-role-policy.json</pre><pre>aws iam create-role \
	  --role-name Tenant-2-
	role \
	  --assume-role-poli
	cy-document file://as
	sume-role-policy.json</pre>The file `assume-role-policy.json` is a JSON document in the current folder that grants `AssumeRole` permissions to the Lambda execution role:<pre>{
	    "Version":
	 "2012-10-17",
	    "Statement": [
	        {
	            "Effect":
	 "Allow",
	            "Principa
	l": {
	                 "AWS":
	 "<Lambda execution role
	 for index function>",
	                 "AWS":
	 "<Lambda execution role
	 for search function>"
	             },
	            "Action":
	 "sts:AssumeRole"</pre>	

Task	Description	Skills required

	``
	        }
	    ]
	}
	``	

Task	Description	Skills required

Create a tenant IAM policy.	Run the AWS CLI [create-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/create-policy.html) command to create a tenant policy that grants access to Elasticsearch operations:
	
	`aws iam create-policy \ --policy-name tenant-policy \ --policy-document file://policy.json`
	
	The file `policy.json` is a JSON document in the current folder that grants permissions on Elasticsearch:
	
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "es:ESHttpDelete", "es:ESHttpGet", "es:ESHttpHead", "es:ESHttpPost", "es:ESHttpPut", "es:ESHttpPatch" ], "Resource": [`
		Cloud architect, Cloud administrator

Task	Description	Skills required

	``
	                "<ARN of
	 Elasticsearch domain
	 created earlier>"
	            ]
	        }
	    ]
	}
	``	

Attach the tenant IAM policy to the tenant roles.	Run the AWS CLI [attach-role-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html) command to attach the tenant IAM policy to the two tenant roles you created in the earlier step:
	
	``
	aws iam attach-role-policy \
	  --policy-arn
	 arn:aws:iam::account-id:policy/tenant-policy \
	  --role-name Tenant-1-role
	
	aws iam attach-role-policy \
	  --policy-arn
	 arn:aws:iam::account-id:policy/tenant-policy \
	  --role-name Tenant-2-role
	``
	
	The policy ARN is from the output of the previous step.
		Cloud architect, Cloud administrator

Task	Description	Skills required

Create an IAM policy to give Lambda permissions to assume role.	Run the AWS CLI [create-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/create-policy.html) command to create a policy for Lambda to assume the tenant role:
	`aws iam create-policy \ --policy-name assume-tenant-role-policy \ --policy-document file://lambda_policy.json`
	The file `lambda_policy.json` is a JSON document in the current folder that grants permissions to `AssumeRole` :
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "sts:AssumeRole", "Resource": "<ARN of tenant role created earlier>" } ] }`
	For `Resource`, you can use a wildcard character to avoid
		Cloud architect, Cloud administrator

Task	Description	Skills required
	creating a new policy for each tenant.	

Create an IAM policy to give the Lambda index role permission to access Amazon S3.	Run the AWS CLI [create-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/create-policy.html) command to give the Lambda index role permission to access the objects in the S3 bucket:
	`aws iam create-policy \ --policy-name s3-permission-policy \ --policy-document file://s3_lambda_policy.json`
	The file `s3_lambda_policy.json` is the following JSON policy document in the current folder:
	`{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::tenantrawdata/*" } ] }`
		Cloud architect, Cloud administrator

Task	Description	Skills required

Attach the policy to the Lambda execution role.	Run the AWS CLI [attach-role-policy](https://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html) command to attach the policy created in the previous step to the Lambda index and search execution roles you created earlier:
	`aws iam attach-role-policy \`
	`  --policy-arn arn:aws:iam::account-id:policy/assume-tenant-role-policy \`
	`  --role-name index-lambda-role`
	
	`aws iam attach-role-policy \`
	`  --policy-arn arn:aws:iam::account-id:policy/assume-tenant-role-policy \`
	`  --role-name search-lambda-role`
	
	`aws iam attach-role-policy \`
	`  --policy-arn arn:aws:iam::account-id:policy/s3-permission-policy \`
	`  --role-name index-lambda-role`
	
	The policy ARN is from the output of the previous step.
		Cloud architect, Cloud administrator

Create and configure a search API

Task	Description	Skills required
Create a REST API in API Gateway.	Run the AWS CLI [create-rest-api](https://docs.aws.amazon.com/cli/latest/reference/apigateway/create-rest-api.html) command to create a REST API resource: `aws apigateway create-rest-api \ --name Test-Api \ --endpoint-configuration "{ \"types\": [\"REGIONAL\"] }"` For the endpoint configuration type, you can specify `EDGE` instead of `REGIONAL` to use edge locations instead of a particular AWS Region. Note the value of the `id` field from the command output. This is the API ID that you will use in subsequent commands.	Cloud architect, Cloud administrator
Create a resource for the search API.	The search API resource starts the Lambda search function with the resource name `search`. (You don't have to create an API for the Lambda index function, because it runs automatically when objects are uploaded to the S3 bucket.)	Cloud architect, Cloud administrator

Task	Description	Skills required

	1. Run the AWS CLI get-resources command to get the parent ID for the root path:
	`aws apigateway get-resources \ --rest-api-id <API-ID>`
	Note the value of the ID field. You will use this parent ID in the next command.
	`{ "items": [ { "id": "zpsri964ck", "path": "/" } ] }`
	2. Run the AWS CLI create-resource command to create a resource for the search API. For `parent-id` , specify the ID from the previous command.
	`aws apigateway create-resource \ --rest-api-id <API-ID> \ --parent-id <Parent-ID> \`	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code> --path-part search</code></pre></td><td></td></tr>
<tr><td>Create a GET method for the search API.</td><td>Run the AWS CLI put-method command to create a <code>GET</code> method for the search API:<pre><code>aws apigateway put-method \
  --rest-api-id <API-ID> \
  --resource-id <ID from the previous command output> \
  --http-method GET \
  --authorization-type "NONE" \
  --no-api-key-required</code></pre>For <code>resource-id</code>, specify the ID from the output of the <code>create-resource</code> command.</td><td>Cloud architect, Cloud administrator</td></tr>
</table>

Task	Description	Skills required

Create a method response for the search API.	Run the AWS CLI put-method-response command to add a method response for the search API:
	`aws apigateway put-method-response \`
	`  --rest-api-id <API-ID> \`
	`  --resource-id  <ID from the create-resource command output> \`
	`  --http-method GET \`
	`  --status-code 200 \`
	`  --response-models "{\"application/json\": \"Empty\"}"`
	For `resource-id` , specify the ID from the output of the earlier `create-resource` command.
		Cloud architect, Cloud administrator

Task	Description	Skills required
Set up a proxy Lambda integration for the search API.	Run the AWS CLI [put-integration](https://docs.aws.amazon.com/cli/latest/reference/apigateway/put-integration.html) command to set up an integration with the Lambda search function: `aws apigateway put-integration \ --rest-api-id <API-ID> \ --resource-id <ID from the create-resource command output> \ --http-method GET \ --type AWS_PROXY \ --integration-http-method GET \ --uri arn:aws:apigateway:region:lambda:path/2015-03-31/functions/arn:aws:lambda:<region>:<account-id>:function:<function-name>/invocations` For `resource-id` , specify the ID from the earlier `create-resource` command.	Cloud architect, Cloud administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr>
<td>Grant API Gateway permission to call the Lambda search function.</td>
<td>

Run the AWS CLI [add-permission](https://docs.aws.amazon.com/cli/latest/reference/lambda/add-permission.html) command to give API Gateway permission to use the search function:

```
aws lambda add-permission \
  --function-name <function-name> \
  --statement-id apigateway-get \
  --action lambda:InvokeFunction \
  --principal apigateway.amazonaws.com \
  --source-arn "arn:aws:execute-api:<region>:<account-id>:api-id/*/GET/search
```

Change the `source-arn` path if you used a different API resource name instead of `search`.

</td>
<td>Cloud architect, Cloud administrator</td>
</tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy the search API.</td><td>Run the AWS CLI create-deployment command to create a stage resource named dev:<pre><code>aws apigateway create-deployment \
  --rest-api-id <API-ID> \
  --stage-name dev</code></pre>If you update the API, you can use the same AWS CLI command to redeploy it to the same stage.</td><td>Cloud architect, Cloud administrator</td></tr>
</table>

Create and configure Kibana roles

Task	Description	Skills required
Log in to the Kibana console.	1. Find the link to Kibana on your domain dashboard on the OpenSearch Service console. The URL is in the form: `<domain-endpoint>/_plugin/kibana/` . 2. Use the bastion host you configured in the first epic to access the Kibana console. 3. Log in to the Kibana console by using the master user name and password from the earlier	Cloud architect, Cloud administrator

Task	Description	Skills required
	step, when you created the OpenSearch Service domain. 4. When prompted to select a tenant, choose **Private**.	

Task	Description	Skills required
Create and configure Kibana roles.	To provide data isolation and to make sure that one tenant cannot retrieve the data of another tenant, you need to use document security, which allows tenants to access only documents that contain their tenant ID. 1. On the Kibana console, in the navigation pane, choose **Security**, **Role**. 2. Create a new tenant role. 3. Set cluster permissions to `indices_all` , which gives create, read, update, and delete (CRUD) permissions on the OpenSearch Service index. 4. Restrict index permissions to the `tenant-data` index. (The index name should match the name in the Lambda search and index functions.) 5. Set index permissions to `indices_all` , to enable users to perform all index-related operations. (You can restrict operations for more granular access, depending on your requirements.)	Cloud architect, Cloud administrator

Task	Description	Skills required
	6. For document-level security, use the following policy to filter documents by tenant ID, to provide data isolation for tenants in a shared index: `{ "bool": { "must": { "match": { "TenantId": "Tenant-1" } } } }` The index name, properties, and values are case-sensitive.	

Task	Description	Skills required
Map users to roles.	1. Choose the **Mapped users** tab for the role, and then choose **Map users**. 2. In the **Backend roles** section, specify the ARN of the IAM tenant role that you created earlier, and then choose **Map**. This maps the IAM tenant role to the Kibana role so that tenant-specific search returns data for that tenant only. For example, if the IAM role name for Tenant-1 is `Tenant-1-Role` , specify the ARN for `Tenant-1-Role` (from the *Create and configure tenant roles* epic) in the **Backend roles** box for the Tenant-1 Kibana role. 3. Repeat steps 1 and 2 for Tenant-2. We recommend that you automate the creation of the tenant and Kibana roles at the time of tenant onboardin g.	Cloud architect, Cloud administrator

Task	Description	Skills required

Create the tenant-data index.	In the navigation pane, under **Management**, choose **Dev Tools**, and then run the following command. This command creates the `tenant-data` index to define the mapping for the `TenantId` property.
	`PUT /tenant-data`
	`{`
	`  "mappings": {`
	`    "properties": {`
	`      "TenantId":`
	` { "type": "keyword"}`
	`    }`
	`  }`
	`}`
		Cloud architect, Cloud administrator

Create VPC endpoints for Amazon S3 and AWS STS

Task	Description	Skills required

Create a VPC endpoint for Amazon S3.	Run the AWS CLI [create-vpc-endpoint](#) command to create a VPC endpoint for Amazon S3. The endpoint enables the Lambda index function in the VPC to access Amazon S3.
	`aws ec2 create-vpc-endpoint \`
	`  --vpc-id <VPC-ID> \`
	`  --service-name com.amazonaws.us-east-1.s3 \`
		Cloud architect, Cloud administrator

Task	Description	Skills required
	` --route-table-ids <route-table-ID>` For `vpc-id`, specify the VPC that you're using for the Lambda index function. For `service-name` , use the correct URL for the Amazon S3 endpoint. For `route-table-ids` , specify the route table that's associated with the VPC endpoint.	

Task	Description	Skills required
Create a VPC endpoint for AWS STS.	Run the AWS CLI [create-vpc-endpoint](https://docs.aws.amazon.com/cli/latest/reference/ec2/create-vpc-endpoint.html) command to create a VPC endpoint for AWS Security Token Service (AWS STS). The endpoint enables the Lambda index and search functions in the VPC to access AWS STS. The functions use AWS STS when they assume the IAM role. `aws ec2 create-vpc-endpoint \ --vpc-id <VPC-ID> \ --vpc-endpoint-type Interface \ --service-name com.amazonaws.us-east-1.sts \ --subnet-id <subnet-ID> \ --security-group-id <security-group-ID>` For `vpc-id`, specify the VPC that you're using for the Lambda index and search functions. For `subnet-id`, provide the subnet in which this endpoint should be created. For `security-group-id`, specify the security group to associate this endpoint with. (It could be the same as the security group Lambda uses.)	Cloud architect, Cloud administrator

Test multi-tenancy and data isolation

Task	Description	Skills required
Update the Python files for the index and search functions.	1. In the `index_lambda_package.zip` file, edit the `lamba_index.py` file to update the AWS account ID, AWS Region, and Elasticsearch endpoint information. 2. In the `search_lambda_package.zip` file, edit the `lambda_search.py` file to update the AWS account ID, AWS Region, and Elasticsearch endpoint information. You can get the Elasticsearch endpoint from the **Overview** tab of the OpenSearch Service console. It has the format `<AWS-Region>.es.amazonaws.com` .	Cloud architect, App developer
Update the Lambda code.	Use the AWS CLI [update-function-code](https://docs.aws.amazon.com/cli/latest/reference/lambda/update-function-code.html) command to update the Lambda code with the changes you made to the Python files: `aws lambda update-function-code \`	Cloud architect, App developer

Task	Description	Skills required

	`--function-name index-lambda-function \ --zip-file fileb://index_lambda_package.zip`
	
	`aws lambda update-function-code \ --function-name search-lambda-function \ --zip-file fileb://search_lambda_package.zip`	

Upload raw data to the S3 bucket.	Use the AWS CLI [cp](https://docs.aws.amazon.com/cli/latest/reference/s3/cp.html) command to upload data for the Tenant-1 and Tenant-2 objects to the `tenantrawdata` bucket (specify the name of the S3 bucket you created for this purpose):
	
	`aws s3 cp tenant-1-data s3://tenantrawdata`
	`aws s3 cp tenant-2-data s3://tenantrawdata`
	
	The S3 bucket is set up to run the Lambda index function whenever data is uploaded so that the document is indexed in Elasticsearch.
		Cloud architect, Cloud administrator

Task	Description	Skills required
Search data from the Kibana console.	On the Kibana console, run the following query: `GET tenant-data/_search` This query displays all the documents indexed in Elasticsearch. In this case, you should see two separate documents for Tenant-1 and Tenant-2.	Cloud architect, Cloud administrator

Task	Description	Skills required

Test the search API from API Gateway.	1. In the API Gateway console, open the search API, choose the GET method inside the search resource, and then choose **Test**.
	2. In the test window, provide the following query string (case-sensitive) for the tenant ID, and then choose **Test**.
	`TenantId=Tenant-1`
	The Lambda function sends a query to OpenSearch Service that filters the tenant document based on the document-level security. The method returns the document that belongs to Tenant-1.
	3. Change the query string to:
	`TenantId=Tenant-2`
	This query returns the document that belongs to Tenant-2.
	
	For screen illustrations, see the [Additional information](#) section.
		Cloud architect, App developer

Task	Description	Skills required
Clean up resources.	Clean up all the resources you created to prevent additional charges to your account.	AWS DevOps, Cloud architect, Cloud administrator

Related resources

- AWS SDK for Python (Boto)
- AWS Lambda documentation
- API Gateway documentation
- Amazon S3 documentation
- Amazon OpenSearch Service documentation
  - Fine-grained access control in Amazon OpenSearch Service
  - Creating a search application with Amazon OpenSearch Service
  - Launching your Amazon OpenSearch Service domains within a VPC

Additional information

Data partitioning models

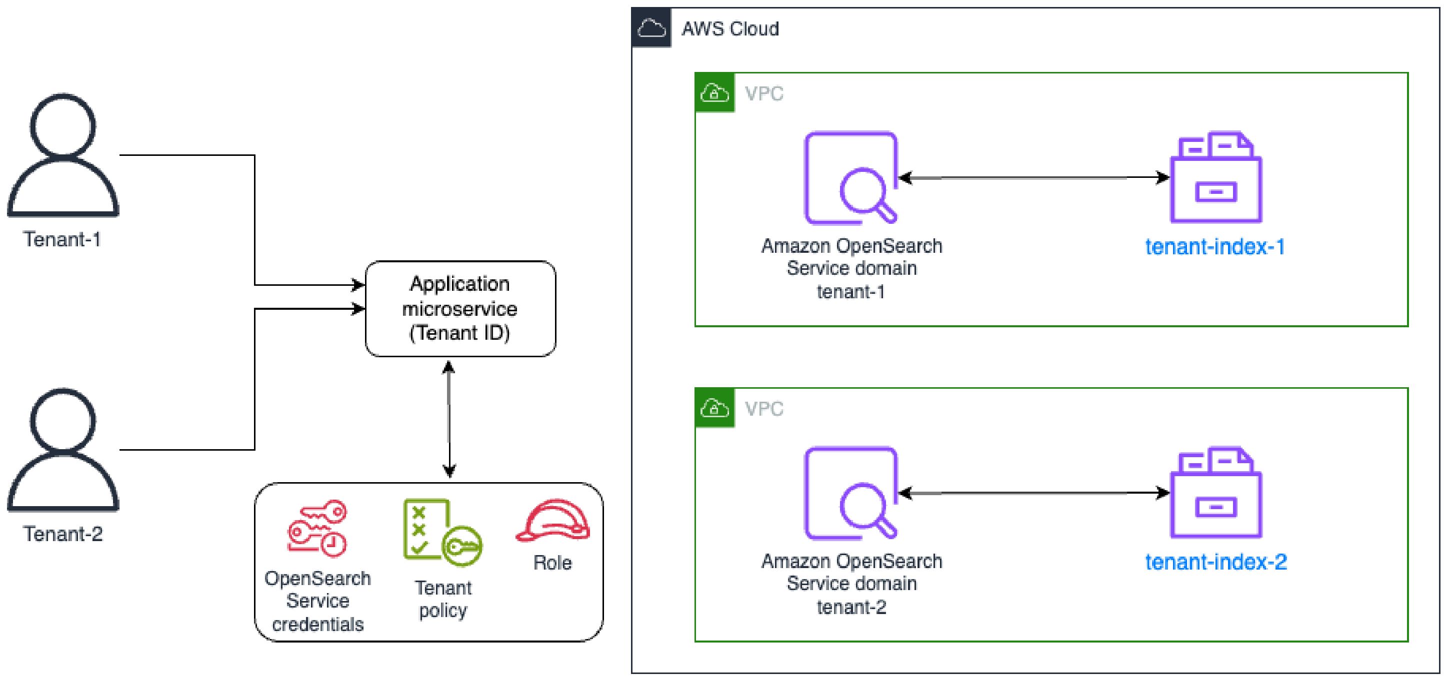
There are three common data partitioning models used in multi-tenant systems: silo, pool, and hybrid. The model you choose depends on the compliance, noisy neighbor, operations, and isolation needs of your environment.

Silo model

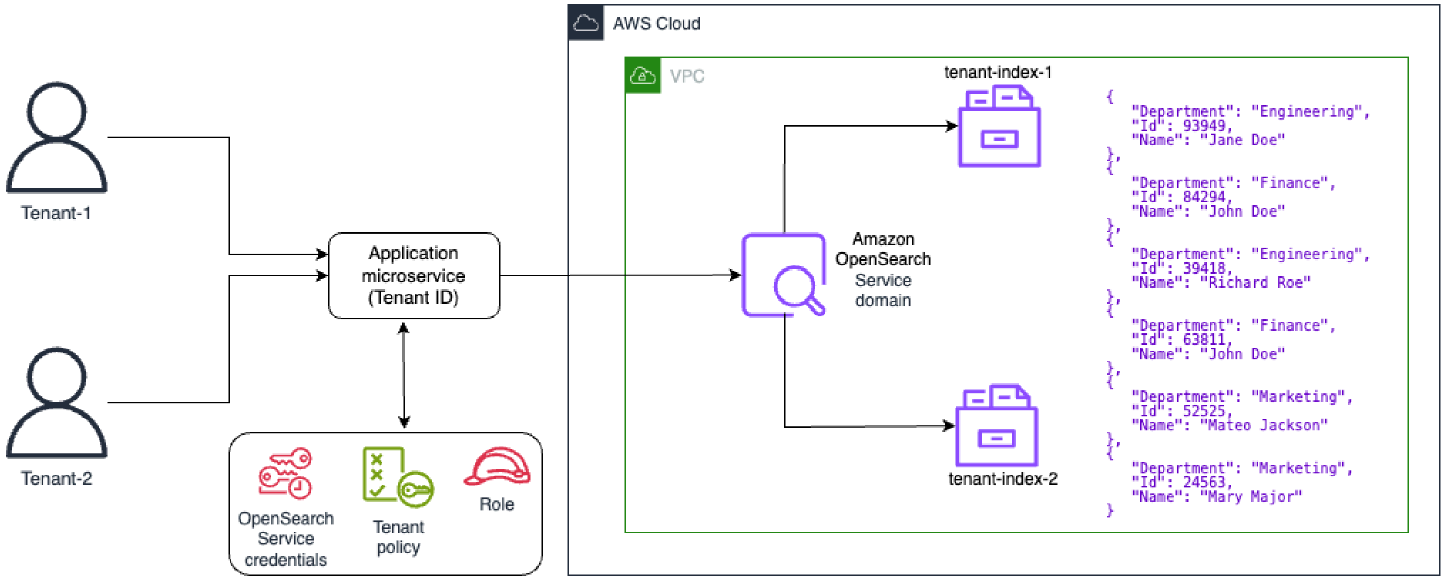
In the silo model, each tenant's data is stored in a distinct storage area where there is no commingling of tenant data. You can use two approaches to implement the silo model with OpenSearch Service: domain per tenant and index per tenant.

- **Domain per tenant** – You can use a separate OpenSearch Service domain (synonymous with an Elasticsearch cluster) per tenant. Placing each tenant in its own domain provides all the benefits associated with having data in a standalone construct. However, this approach introduces management and agility challenges. Its distributed nature makes it harder to aggregate and assess the operational health and activity of tenants. This is a costly option that requires each

OpenSearch Service domain to have three master nodes and two data nodes for production workloads at the minimum.



- **Index per tenant** – You can place tenant data in separate indexes within an OpenSearch Service cluster. With this approach, you use a tenant identifier when you create and name the index, by prepending the tenant identifier to the index name. The index per tenant approach helps you achieve your silo goals without introducing a completely separate cluster for each tenant. However, you might encounter memory pressure if the number of indexes grows, because this approach requires more shards, and the master node has to handle more allocation and rebalancing.



Isolation in the silo model – In the silo model, you use IAM policies to isolate the domains or indexes that hold each tenant's data. These policies prevent one tenant from accessing another tenant's data. To implement your silo isolation model, you can create a resource-based policy that controls access to your tenant resource. This is often a domain access policy that specifies which actions a principal can perform on the domain's sub-resources, including Elasticsearch indexes and APIs. With IAM identity-based polices, you can specify *allowed* or *denied* actions on the domain, indexes, or APIs within OpenSearch Service. The `Action` element of an IAM policy describes the specific action or actions that are allowed or denied by the policy, and the `Principal` element specifies the affected accounts, users, or roles.

The following sample policy grants Tenant-1 full access (as specified by `es:*`) to the sub-resources on the `tenant-1` domain only. The trailing `/*` in the `Resource` element indicates that this policy applies to the domain's sub-resources, not to the domain itself. When this policy is in effect, tenants are not allowed to create a new domain or modify settings on an existing domain.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::<aws-account-id>:user/Tenant-1"
         },
         "Action": "es:*",
         "Resource": "arn:aws:es:<Region>:<account-id>:domain/tenant-1/*"
```

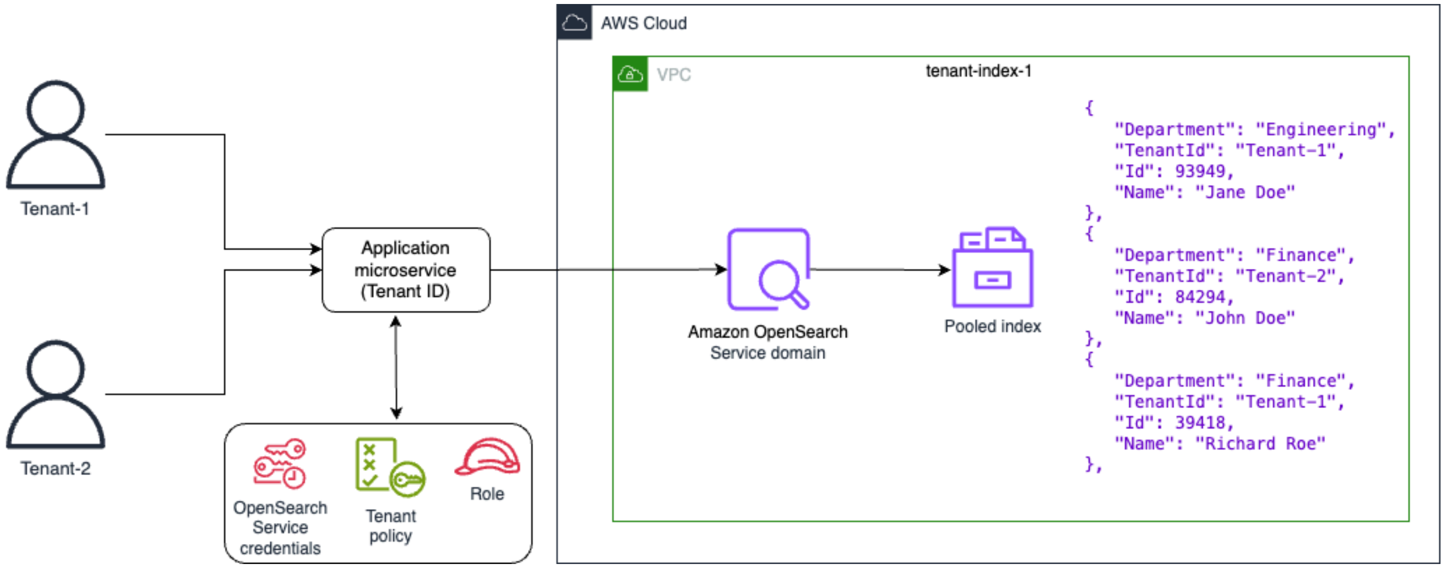
```
        }
    ]
}
```

To implement the tenant per index silo model, you would need to modify this sample policy to further restrict Tenant-1 to the specified index or indexes, by specifying the index name. The following sample policy restricts Tenant-1 to the `tenant-index-1` index.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Tenant-1"
         },
         "Action": "es:*",
         "Resource": "arn:aws:es:<Region>:<account-id>:domain/test-domain/tenant-
index-1/*"
      }
   ]
}
```

Pool model

In the pool model, all tenant data is stored in an index within the same domain. The tenant identifier is included in the data (document) and used as the partition key, so you can determine which data belongs to which tenant. This model reduces the management overhead. Operating and managing the pooled index is easier and more efficient than managing multiple indexes. However, because tenant data is commingled within the same index, you lose the natural tenant isolation that the silo model provides. This approach might also degrade performance because of the noisy neighbor effect.



Tenant isolation in the pool model – In general, tenant isolation is challenging to implement in the pool model. The IAM mechanism used with the silo model doesn't allow you to describe isolation based on the tenant ID stored in your document.

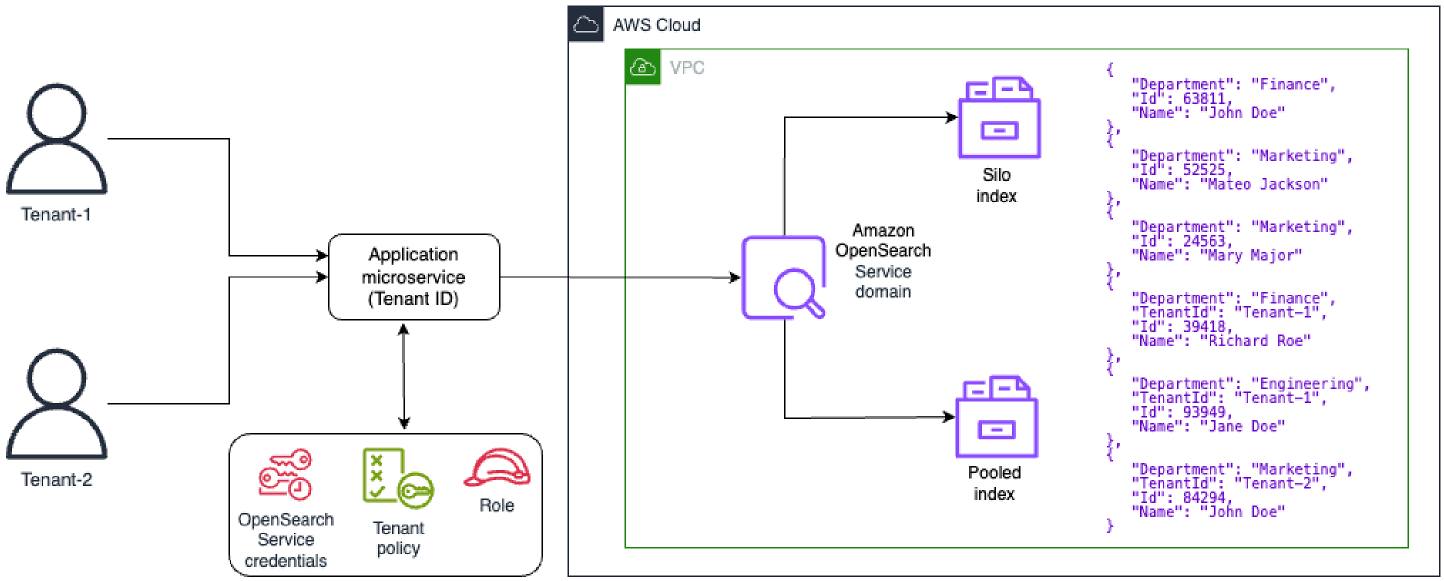
An alternative approach is to use the fine-grained access control (FGAC) support provided by the Open Distro for Elasticsearch. FGAC allows you to control permissions at an index, document, or field level. With each request, FGAC evaluates the user credentials and either authenticates the user or denies access. If FGAC authenticates the user, it fetches all roles mapped to that user and uses the complete set of permissions to determine how to handle the request.

To achieve the required isolation in the pooled model, you can use document-level security, which lets you restrict a role to a subset of documents in an index. The following sample role restricts queries to Tenant-1. By applying this role to Tenant-1, you can achieve the necessary isolation.

```
{
   "bool": {
     "must": {
       "match": {
         "tenantId": "Tenant-1"
       }
     }
   }
 }
```

Hybrid model

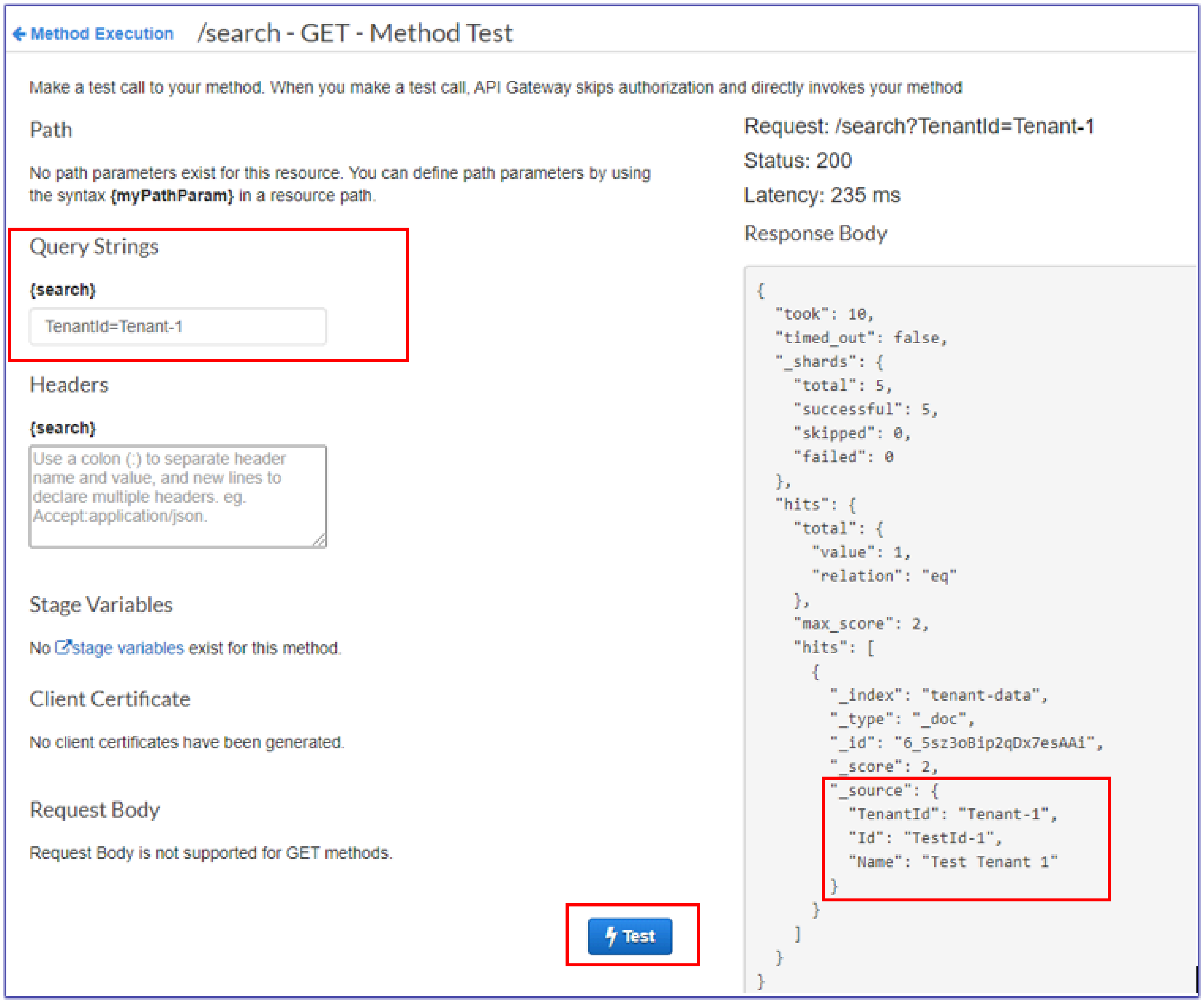
The hybrid model uses a combination of the silo and pool models in the same environment to offer unique experiences to each tenant tier (such as free, standard, and premium tiers). Each tier follows the same security profile that was used in the pool model.



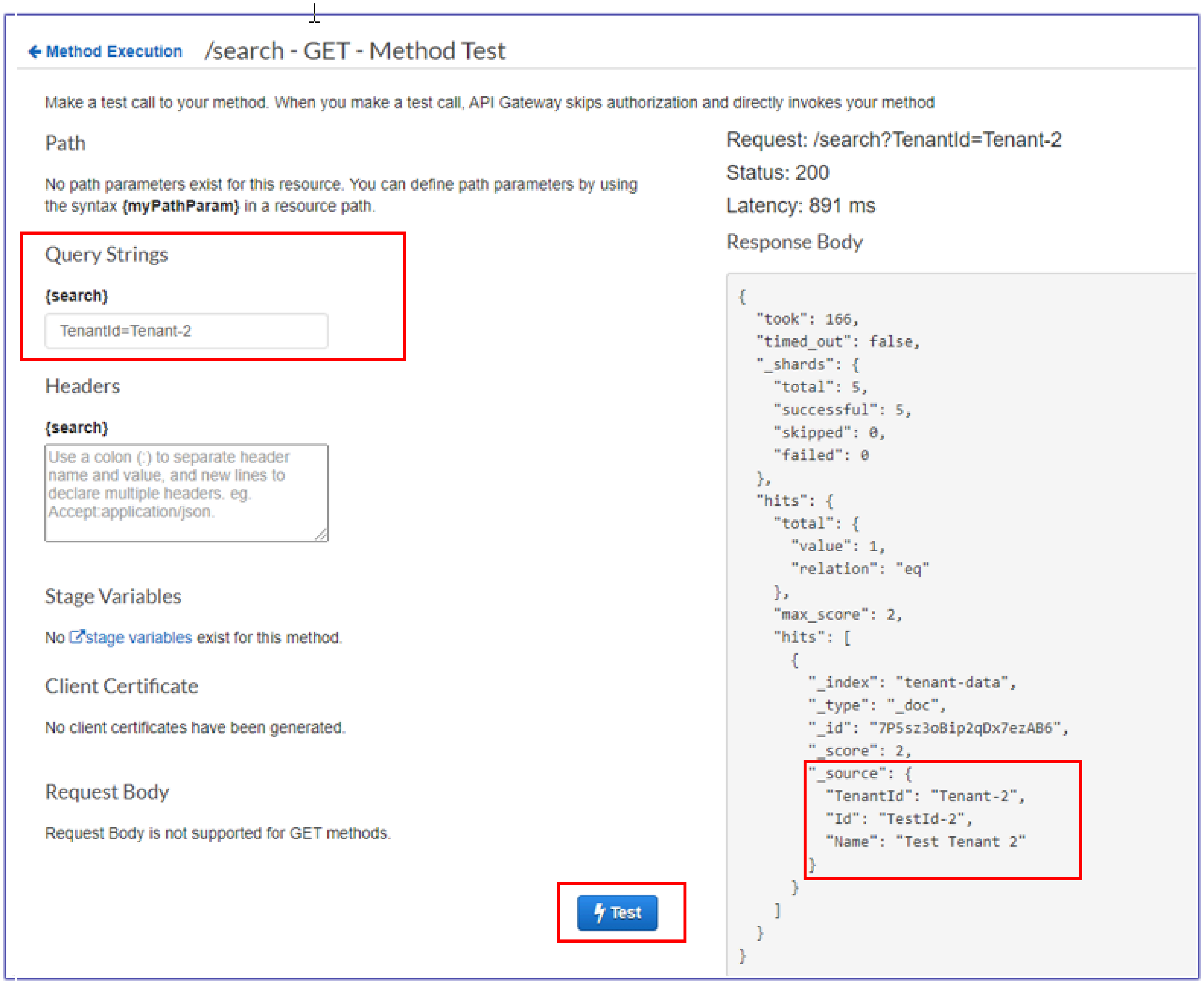
Tenant isolation in the hybrid model – In the hybrid model, you follow the same security profile as in the pool model, where using the FGAC security model at the document level provided tenant isolation. Although this strategy simplifies cluster management and offers agility, it complicates other aspects of the architecture. For example, your code requires additional complexity to determine which model is associated with each tenant. You also have to ensure that single-tenant queries don't saturate the entire domain and degrade the experience for other tenants.

Testing in API Gateway

Test window for Tenant-1 query



Test window for Tenant-2 query



Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Deploy multiple-stack applications using AWS CDK with TypeScript

Dr. Rahul Sharad Gaikwad, Amazon Web Services

Summary

This pattern provides a step-by-step approach for application deployment on Amazon Web Services (AWS) using AWS Cloud Development Kit (AWS CDK) with TypeScript. As an example, the pattern deploys a serverless real-time analytics application.

The pattern builds and deploys nested stack applications. The parent AWS CloudFormation stack calls the child, or nested, stacks. Each child stack builds and deploys the AWS resources that are defined in the CloudFormation stack. AWS CDK Toolkit, the command line interface (CLI) command `cdk`, is the primary interface for the CloudFormation stacks.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Existing virtual private cloud (VPC) and subnets
- AWS CDK Toolkit installed and configured
- A user with administrator permissions and a set of access keys.
- Node.js
- AWS Command Line Interface (AWS CLI)

Limitations

- Because AWS CDK uses AWS CloudFormation, AWS CDK applications are subject to CloudFormation service quotas. For more information, see [AWS CloudFormation quotas].

Product versions

This pattern has been built and tested using the following tools and versions.

- AWS CDK Toolkit 1.83.0
- Node.js 14.13.0

- npm 7.0.14

The pattern should work with any version of AWS CDK or npm. Note that Node.js versions 13.0.0 through 13.6.0 are not compatible with the AWS CDK.

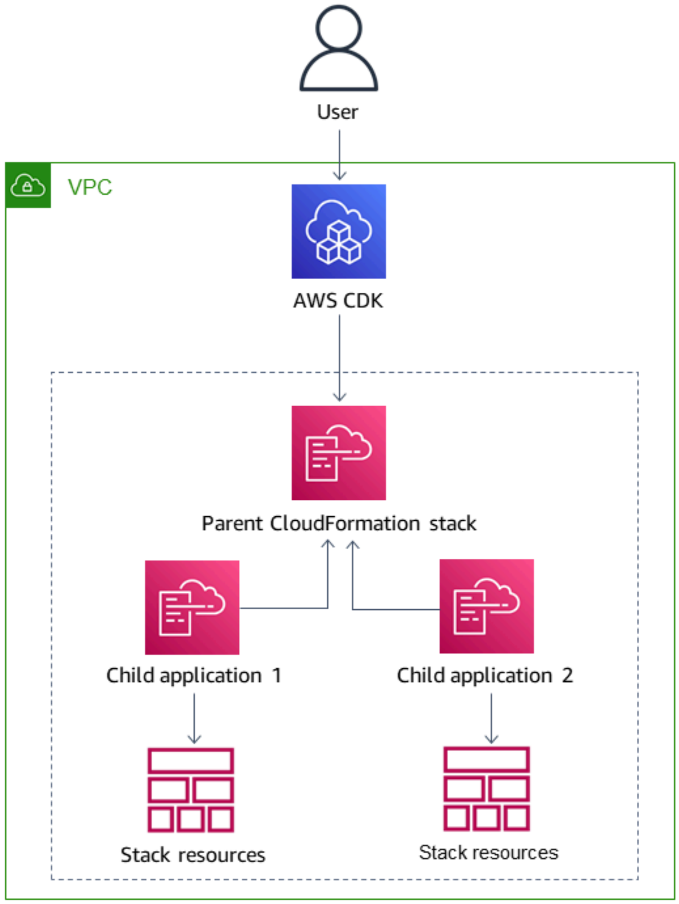
Architecture

Target technology stack

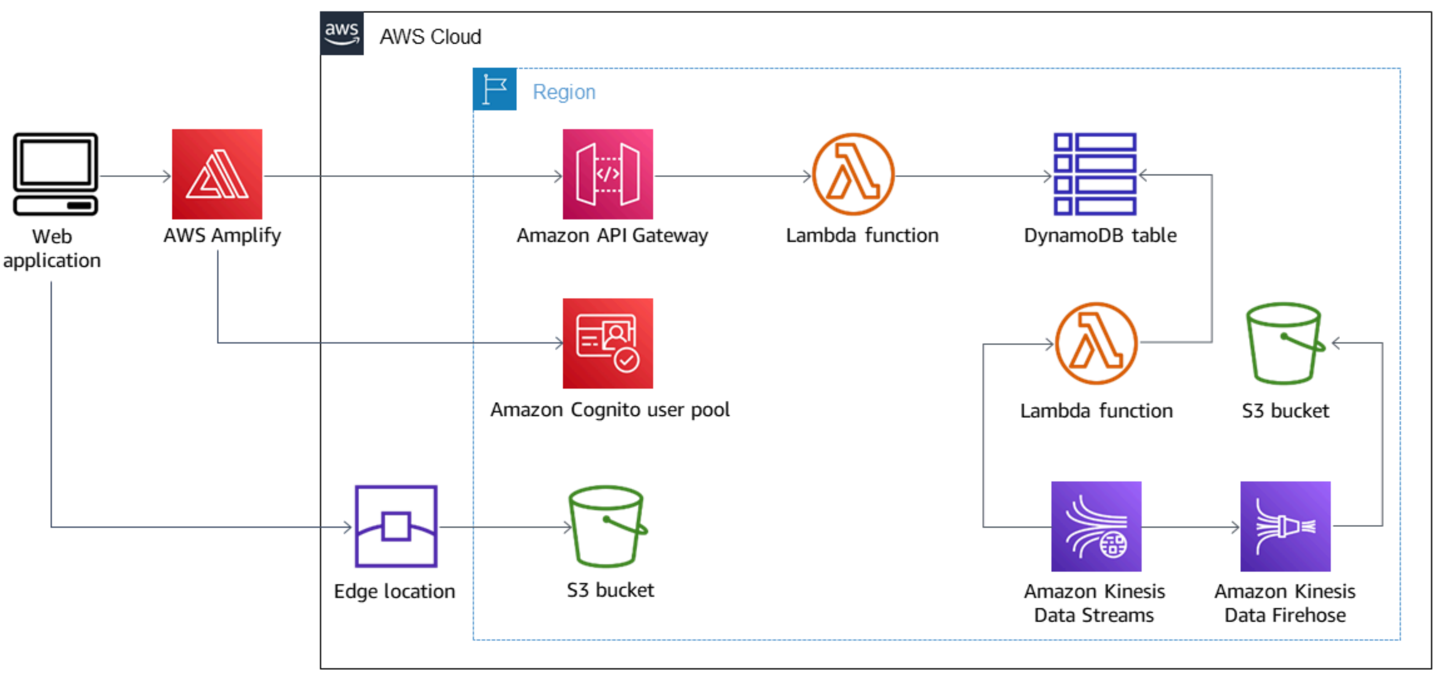
- AWS Amplify Console
- Amazon API Gateway
- AWS CDK
- Amazon CloudFront
- Amazon Cognito
- Amazon DynamoDB
- Amazon Data Firehose
- Amazon Kinesis Data Streams
- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)

Target architecture

The following diagram shows multiple-stack application deployment using AWS CDK with TypeScript.



The following diagram shows the architecture of the example serverless real-time application.



Tools

Tools

- AWS Amplify Console is the control center for fullstack web and mobile application deployments in AWS. Amplify Console hosting provides a git-based workflow for hosting fullstack serverless web apps with continuous deployment. The Admin UI is a visual interface for frontend web and mobile developers to create and manage app backends outside the AWS console.
- Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CDK Toolkit is a command line cloud development kit that helps you interact with your AWS CDK app. The `cdk` CLI command is the primary tool for interacting with your AWS CDK app. It runs your app, interrogates the application model you defined, and produces and deploys the AWS CloudFormation templates generated by the AWS CDK.
- Amazon CloudFront is a web service that speeds up distribution of static and dynamic web content, such as .html, .css, .js, and image files. CloudFront delivers your content through a worldwide network of data centers called edge locations for lower latency and improved performance.
- Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly or through a third party.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- Amazon Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers.
- Amazon Kinesis Data Streams is a service for collecting and processing large streams of data records in real time.
- AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code

The code for this pattern is attached.

Epics

Install AWS CDK Toolkit

Task	Description	Skills required
Install AWS CDK Toolkit.	To install AWS CDK Toolkit globally, run the following command. `npm install -g aws-cdk`	DevOps
Verify the version.	To verify the AWS CDK Toolkit version, run the following command. `cdk --version`	DevOps

Set up AWS credentials

Task	Description	Skills required

Set up credentials.	To set up credentials, run the `aws configure` command and follow the prompts.
	
	`$aws configure`
	`AWS Access Key ID [None]:`
	`AWS Secret Access Key [None]: your_secret_access_key`
	`Default region name [None]:`
		DevOps

Task	Description	Skills required
	`Default output format [None]:`	

Download the project code

Task	Description	Skills required
Download the attached project code.	For more information about the directory and file structure, see the *Additional information* section.	DevOps

Bootstrap the AWS CDK environment

Task	Description	Skills required
Bootstrap the environment.	To deploy the AWS CloudFormation template to the account and AWS Region that you want to use, run the following command. `cdk bootstrap <account>/<Region>` For more information, see the AWS documentation.	DevOps

Build and deploy the project

Task	Description	Skills required
Build the project.	To build the project code, run the `npm run build` command.	DevOps
Deploy the project.	To deploy the project code, run the `cdk deploy` command.	

Verify outputs

Task	Description	Skills required
Verify stack creation.	On the AWS Management Console, choose **CloudFormation**. In the stacks for the project, verify that a parent stack and two child stacks have been created.	DevOps

Test the application

Task	Description	Skills required
Send data to Kinesis Data Streams.	Configure your AWS Account to send data to Kinesis Data Streams using Amazon Kinesis Data Generator (KDG). For more information, see [Amazon Kinesis Data Generator]().	DevOps

Task	Description	Skills required
Create an Amazon Cognito user.	To create an Amazon Cognito user, download the cognito-setup.json CloudFormation template from the *Create an Amazon Cognito User* section on the [Kinesis Data Generator help page](). Initiate the template, and then enter your Amazon Cognito **Username** and **Password**. The **Outputs** tab lists the Kinesis Data Generator URL.	DevOps
Log in to Kinesis Data Generator	To log in to KDG, use the Amazon Cognito credentia ls that you provided and the Kinesis Data Generator URL.	DevOps
Test the application.	In KDG, in **Record template**, **Template 1**, paste the test code from the *Additiona l information* section, and choose **Send data**.	DevOps
Test API Gateway.	After the data has been ingested, test API Gateway by using the GET method to retrieve data.	DevOps

Related resources

References

- [AWS Cloud Development Kit]()

- AWS CDK on GitHub
- Working with nested stacks
- AWS sample example - Serverless real-time analytics

Additional information

Directory and file details

This pattern sets up the following three stacks.

- `parent-cdk-stack.ts` – This stack acts as the parent stack and calls the two child applications as nested stacks.
- `real-time-analytics-poc-stack.ts` – This nested stack contains the infrastructure and application code.
- `real-time-analytics-web-stack.ts` – This nested stack contains only the static web application code.

Important files and their functionality

- `bin/real-time-analytics-poc.ts` – Entry point of the AWS CDK application. It loads all stacks defined under `lib/`.
- `lib/real-time-analytics-poc-stack.ts` – Definition of the AWS CDK application's stack (`real-time-analytics-poc`).
- `lib/real-time-analytics-web-stack.ts` – Definition of the AWS CDK application's stack (`real-time-analytics-web-stack`).
- `lib/parent-cdk-stack.ts` – Definition of the AWS CDK application's stack (`parent-cdk`).
- `package.json` – npm module manifest, which includes the application name, version, and dependencies.
- `package-lock.json` – Maintained by npm.
- `cdk.json` – Toolkit for running the application.
- `tsconfig.json` – The project's TypeScript configuration.
- `.gitignore` – List of files that Git should exclude from source control.
- `node_modules` – Maintained by npm; includes the project's dependencies.

The following section of code in the parent stack calls child applications as a nested AWS CDK stacks.

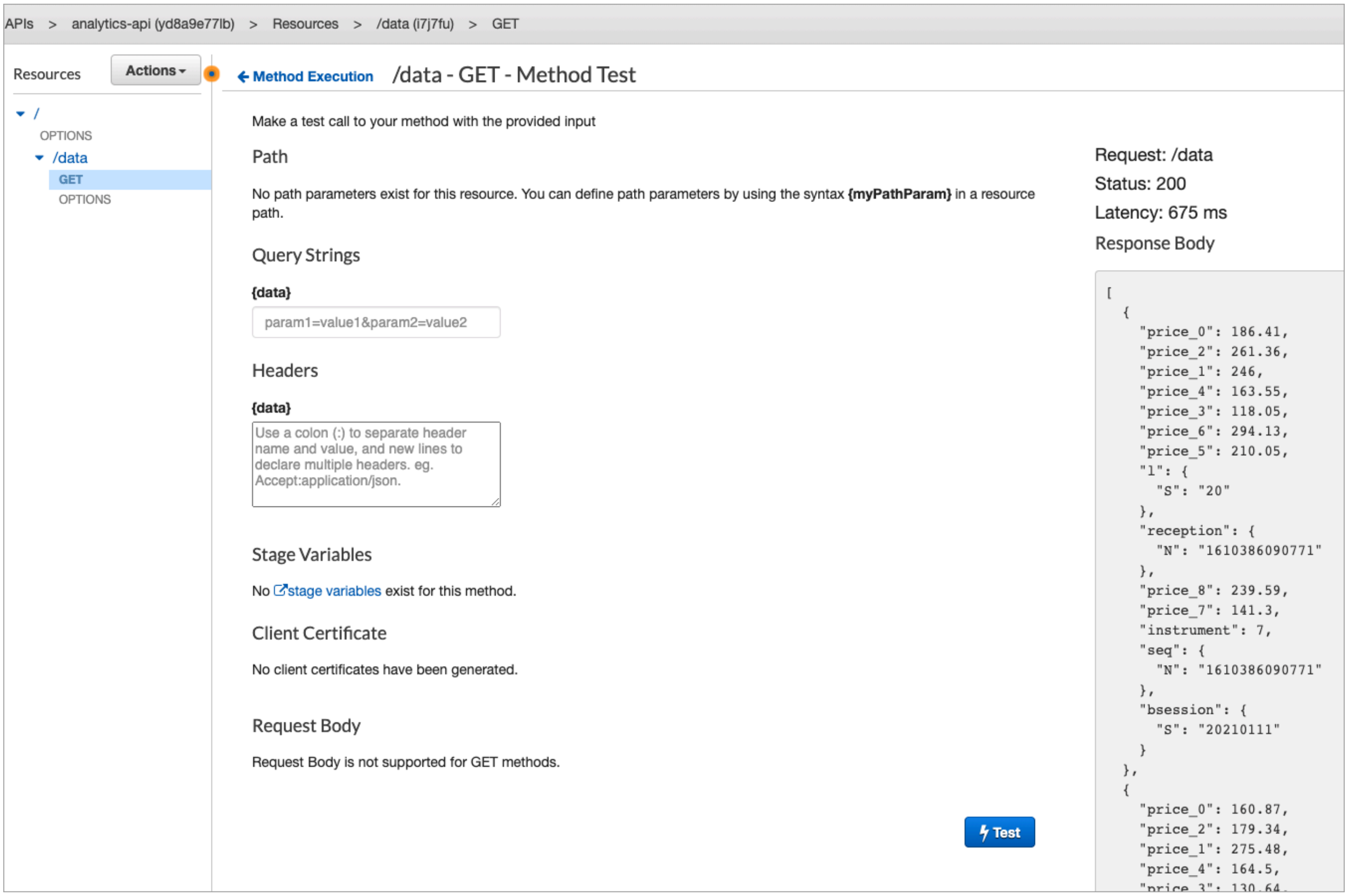
```
import * as cdk from '@aws-cdk/core';
import { Construct, Stack, StackProps } from '@aws-cdk/core';
import { RealTimeAnalyticsPocStack } from './real-time-analytics-poc-stack';
import { RealTimeAnalyticsWebStack } from './real-time-analytics-web-stack';


export class CdkParentStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);


    new RealTimeAnalyticsPocStack(this, 'RealTimeAnalyticsPocStack');
    new RealTimeAnalyticsWebStack(this, 'RealTimeAnalyticsWebStack');
  }
}
```

Code for testing

```
session={{date.now('YYYYMMDD')}}|sequence={{date.now('x')}}|
reception={{date.now('x')}}|instrument={{random.number(9)}}|
l={{random.number(20)}}|price_0={{random.number({"min":10000,
 "max":30000})}}|price_1={{random.number({"min":10000, "max":30000})}}|
price_2={{random.number({"min":10000, "max":30000})}}|
price_3={{random.number({"min":10000, "max":30000})}}|
price_4={{random.number({"min":10000, "max":30000})}}|
price_5={{random.number({"min":10000, "max":30000})}}|
price_6={{random.number({"min":10000, "max":30000})}}|
price_7={{random.number({"min":10000, "max":30000})}}|
price_8={{random.number({"min":10000, "max":30000})}}|
```

Testing API Gateway

On the API Gateway console, test API Gateway by using the GET method.



Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automate deployment of nested applications using AWS SAM

Dr. Rahul Sharad Gaikwad, Ishwar Chauthaiwale, Dmitry Gulin, and Tabby Ward, Amazon Web Services

Summary

On Amazon Web Services (AWS), AWS Serverless Application Model (AWS SAM) is an open-source framework that provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines for each resource, you can define the application you want and model it by using YAML. During deployment, SAM transforms and expands the SAM syntax into AWS CloudFormation syntax that you can use to build serverless applications faster.

AWS SAM simplifies the development, deployment, and management of serverless applications on the AWS platform. It provides a standardized framework, faster deployment, local testing capabilities, resource management, seamless Integration with Development Tools, and a supportive community. These features make it a valuable tool for building serverless applications efficiently and effectively.

This pattern uses AWS SAM templates to automate the deployment of nested applications. A nested application is an application within another application. Parent applications call their child applications. These are loosely coupled components of a serverless architecture.

Using nested applications, you can rapidly build highly sophisticated serverless architectures by reusing services or components that are independently authored and maintained but are composed using AWS SAM and the Serverless Application Repository. Nested applications help you to build applications that are more powerful, avoid duplicated work, and ensure consistency and best practices across your teams and organizations. To demonstrate nested applications, the pattern deploys an example AWS serverless shopping cart application.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An existing virtual private cloud (VPC) and subnets
- An integrated development environment such as Visual Studio Code (for more information, see Tools to Build on AWS)

- Python wheel library installed using pip install wheel, if it’s not already installed

Limitations

- The maximum number of applications that can be nested in a serverless application is 200.
- The maximum number of parameters for a nested application can have 60.

Product versions

- This solution is built on AWS SAM command line interface (AWS SAM CLI) version 1.21.1, but this architecture should work with later AWS SAM CLI versions.

Architecture

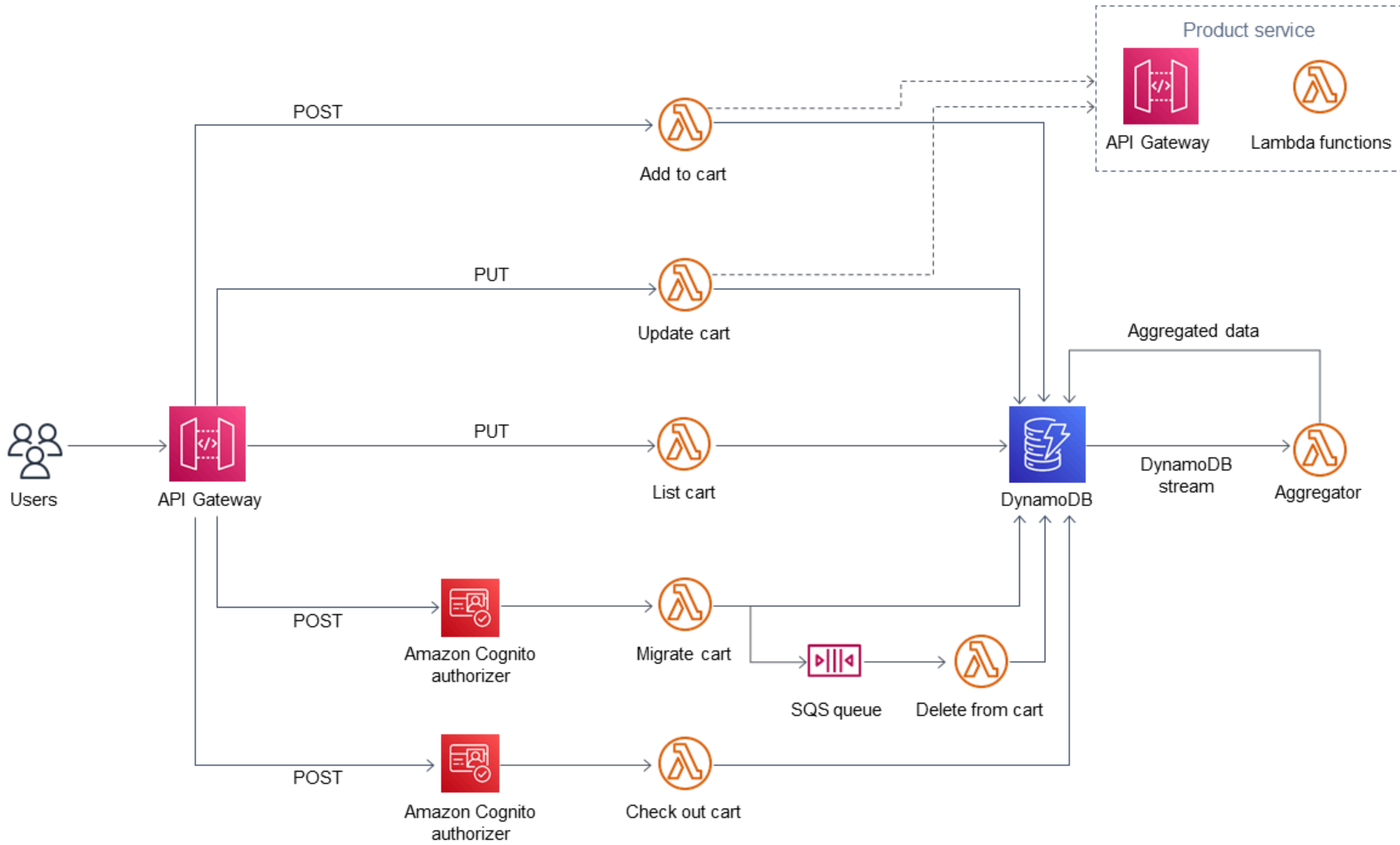
Target technology stack

- Amazon API Gateway
- AWS SAM
- Amazon Cognito
- Amazon DynamoDB
- AWS Lambda
- Amazon Simple Queue Service (Amazon SQS) queue

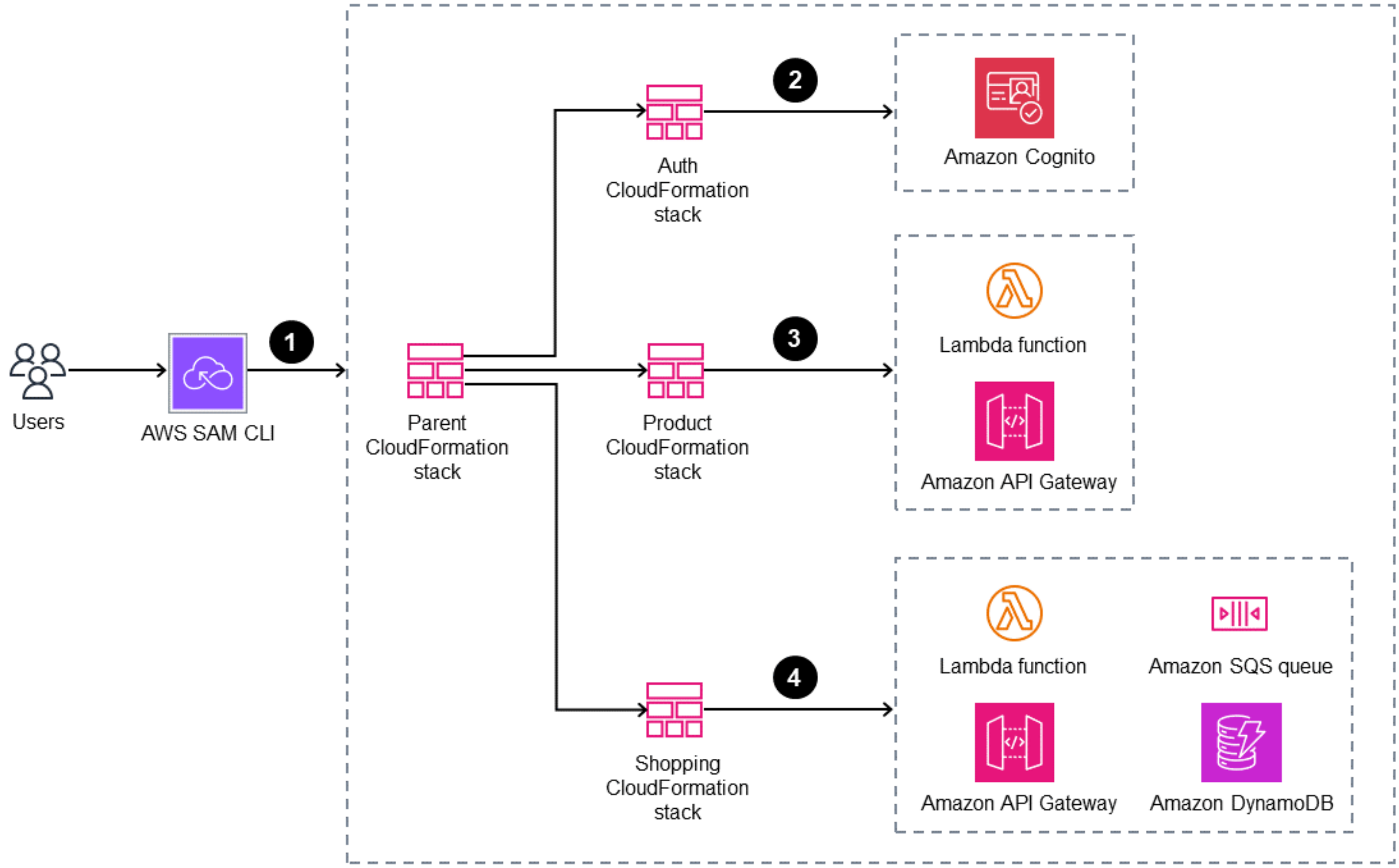
Target architecture

The following diagram shows how user requests are made to the shopping services by calling APIs. The user's request, including all necessary information, is sent to Amazon API Gateway and the Amazon Cognito authorizer, which performs authentication and authorization mechanisms for the APIs.

When an item is added, deleted, or updated in DynamoDB, an event is put onto DynamoDB Streams, which in turn initiates a Lambda function. To avoid immediate deletion of old items as part of a synchronous workflow, messages are put onto an SQS queue, which initiates a worker function to delete the messages.



In this solution setup, AWS SAM CLI serves as the interface for AWS CloudFormation stacks. AWS SAM templates automatically deploy nested applications. The parent SAM template calls the child templates, and the parent CloudFormation stack deploys the child stacks. Each child stack builds the AWS resources that are defined in the AWS SAM CloudFormation templates.



1. Build and deploy the stacks.
2. The Auth CloudFormation stack contains Amazon Cognito.
3. The Product CloudFormation stack contains an Lambda function and Amazon API Gateway
4. The Shopping CloudFormation stack contains a Lambda function, Amazon API Gateway, the SQS queue, and the Amazon DynamoDB database.

Tools

Tools

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- Amazon Cognito provides authentication, authorization, and user management for web and mobile apps.

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Serverless Application Model (AWS SAM) is an open-source framework that helps you build serverless applications in the AWS Cloud.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.

Code

The code for this pattern is available in the GitHub AWS SAM Nested Stack Sample repository.

Epics

Install AWS SAM CLI

Task	Description	Skills required
Install AWS SAM CLI.	To install AWS SAM CLI, see the instructions in the AWS SAM documentation.	DevOps engineer

Set up AWS credentials.	To set AWS credentials so that the AWS SAM CLI can make calls to AWS services on your behalf, run the `aws configure` command and follow the prompts.
	`$aws configure`
	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: your_secret_access_key`
		DevOps engineer

Task	Description	Skills required

	`Default region name [None]:`
	`Default output format [None]:`
	
	For more information on setting up your credentials, see Authentication and access credentials.	

Initialize the AWS SAM project

Task	Description	Skills required

Clone the AWS SAM code repository.	1. Clone the aws sam nested stack sample repository for this pattern by entering the following command.
	`git clone https://github.com/aws-samples/aws-sam-nested-stack-sample.git`
	2. Navigate into the cloned directory by entering the following command.
	`cd aws-sam-nested-stack-sample`
		DevOps engineer

Deploy templates to initialize the project.	To initialize the project, run the SAM `init` command. When prompted to choose a template source, choose	DevOps engineer

Task	Description	Skills required
	`Custom Template Location`.	

Compile and build the SAM template code

Task	Description	Skills required
Review the AWS SAM application templates.	Review the templates for the nested applications. This example uses the following nested application templates: • `auth.yaml` – This template sets up authentication-related resources, such as Amazon Cognito and AWS Systems Manager Parameter Store. • `product-mock.yaml` – This template deploys product-related resources, such as Lambda functions and Amazon API Gateway. • `shoppingcart-service.yaml` – This template sets up shopping cart–related resources, such as AWS Identity and Access Management (IAM), DynamoDB tables, and Lambda functions.	DevOps engineer
Review the parent template.	Review the template that will invoke the nested application	DevOps engineer

Task	Description	Skills required
	templates. In this example, the parent template is `template.yml` . All separate applications are nested in the single parent template `template.yml` .	
Compile and build the AWS SAM template code.	Using the AWS SAM CLI, run the following command. `sam build`	DevOps engineer

Deploy the AWS SAM template

Task	Description	Skills required
Deploy the applications.	To launch the SAM template code that creates the nested application CloudFormation stacks and deploys code in the AWS environment, run the following command. `sam deploy --guided --stack-name shopping-cart-nested-stack --capabilities CAPABILITY_IAM CAPABILITY_AUTO_EXPAND` The command will prompt with a few questions. Answer all questions with `y`.	DevOps engineer

Verify the deployment

Task	Description	Skills required
Verify the stacks.	To review the AWS CloudFormation stacks and AWS resources that were defined in the AWS SAM templates, do the following: 1. Log in to the AWS Management Console, and navigate to the **CloudFormation** console. 2. Verify that the parent and child stacks are listed. In this example, `sam-shopping-cart` is the parent stack that calls the nested Auth, Product and Shopping stacks. The product stack gives the Product API Gateway URL link as an output.	DevOps engineer

Related resources

References

- AWS Serverless Application Model (AWS SAM)
- AWS SAM on GitHub
- Serverless Shopping Cart Microservice (AWS example application)

Tutorials and videos

- Build a Serverless App
- AWS Online Tech Talks: Serverless Application Building and Deployments with AWS SAM

Additional information

After all the code is in place, the example has the following directory structure:

- sam_stacks – This folder contains the `shared.py` layer. A layer is a file archive that contains libraries, a custom runtime, or other dependencies. With layers, you can use libraries in your function without needing to include them in a deployment package.
- *product-mock-service* – This folder contains all product-related Lambda functions and files.
- *shopping-cart-service* – This folder contains all shopping-related Lambda functions and files.

Implement the serverless saga pattern by using AWS Step Functions

Tabby Ward, Joe Kern, and Rohan Mehta, Amazon Web Services

Summary

In a microservices architecture, the main goal is to build decoupled and independent components to promote agility, flexibility, and faster time to market for your applications. As a result of decoupling, each microservice component has its own data persistence layer. In a distributed architecture, business transactions can span multiple microservices. Because these microservices cannot use a single atomicity, consistency, isolation, durability (ACID) transaction, you might end up with partial transactions. In this case, some control logic is needed to undo the transactions that have already been processed. The distributed saga pattern is typically used for this purpose.

The saga pattern is a failure management pattern that helps establish consistency in distributed applications and coordinates transactions between multiple microservices to maintain data consistency. When you use the saga pattern, every service that performs a transaction publishes an event that triggers subsequent services to perform the next transaction in the chain. This continues until the last transaction in the chain is complete. If a business transaction fails, saga orchestrates a series of compensating transactions that undo the changes that were made by the preceding transactions.

This pattern demonstrates how to automate the setup and deployment of a sample application (which handles travel reservations) with serverless technologies such as AWS Step Functions, AWS Lambda, and Amazon DynamoDB. The sample application also uses Amazon API Gateway and Amazon Simple Notification Service (Amazon SNS) to implement a saga execution coordinator. The pattern can be deployed with an infrastructure as code (IaC) framework such as the AWS Cloud Development Kit (AWS CDK), the AWS Serverless Application Model (AWS SAM), or Terraform.

For more information about the saga pattern and other data persistence patterns, see the guide [Enabling data persistence in microservices] on the AWS Prescriptive Guidance website.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Permissions to create an AWS CloudFormation stack. For more information, see [Controlling access] in the CloudFormation documentation.

- IaC framework of your choice (AWS CDK, AWS SAM, or Terraform) configured with your AWS account so that you can use the framework CLI to deploy the application.
- NodeJS, used to build the application and run it locally.
- A code editor of your choice (such as Visual Studio Code, Sublime, or Atom).

Product versions

- NodeJS version 14
- AWS CDK version 2.37.1
- AWS SAM version 1.71.0
- Terraform version 1.3.7

Limitations

Event sourcing is a natural way to implement the saga orchestration pattern in a microservices architecture where all components are loosely coupled and don't have direct knowledge of one another. If your transaction involves a small number of steps (three to five), the saga pattern might be a great fit. However complexity increases with the number of microservices and the number of steps.

Testing and debugging can become difficult when you're using this design, because you have to have all services running in order to simulate the transaction pattern.

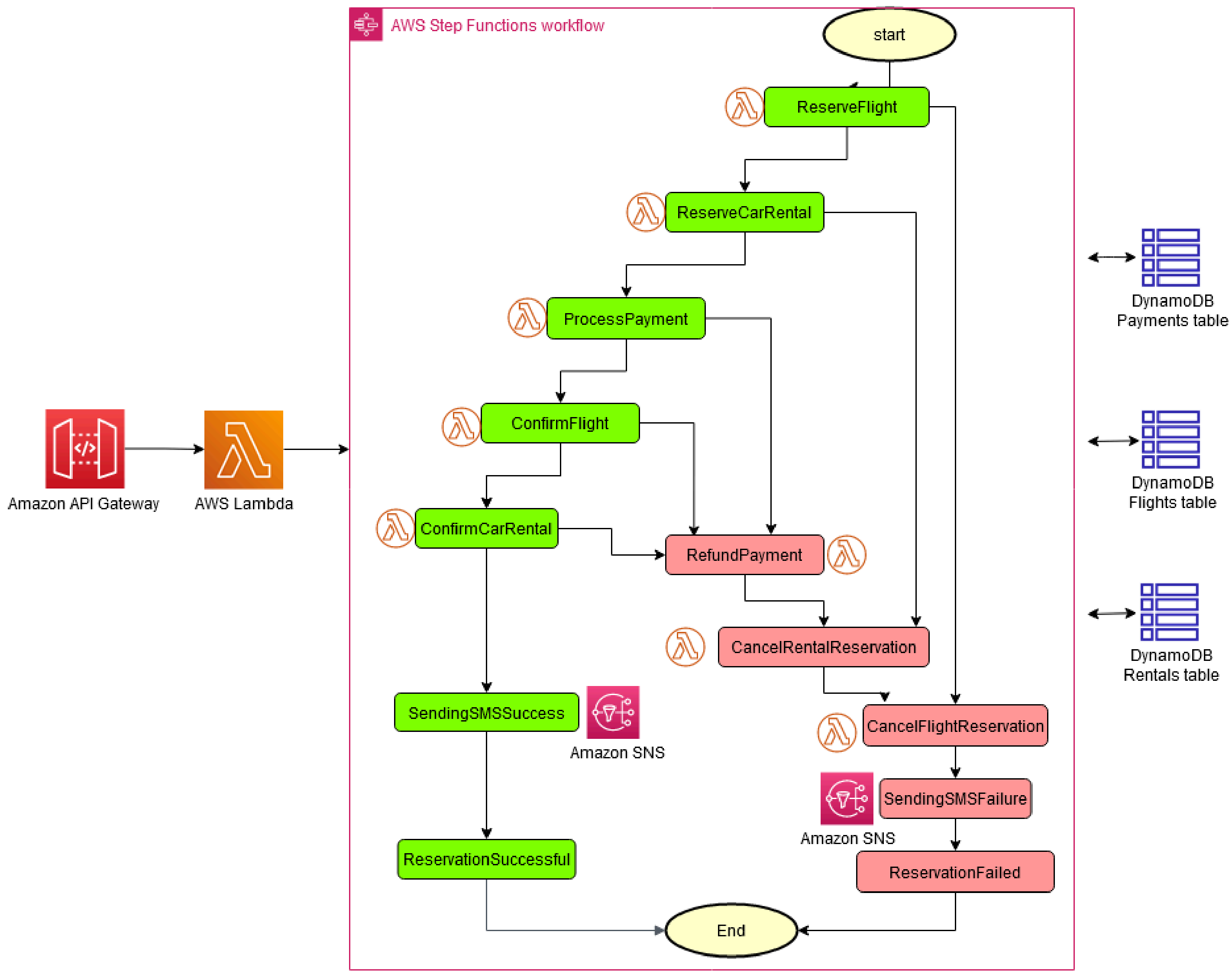
Architecture

Target architecture

The proposed architecture uses AWS Step Functions to build a saga pattern to book flights, book car rentals, and process payments for a vacation.

The following workflow diagram illustrates the typical flow of the travel reservation system. The workflow consists of reserving air travel ("ReserveFlight"), reserving a car ("ReserveCarRental"), processing payments ("ProcessPayment"), confirming flight reservations ("ConfirmFlight"), and confirming car rentals ("ConfirmCarRental") followed by a success notification when these steps are complete. However, if the system encounters any errors in running any of these transactions, it starts to fail backward. For example, an error with payment processing ("ProcessPayment") triggers a refund ("RefundPayment"), which then triggers a cancellation of the rental car and flight ("CancelRentalReservation" and "CancelFlightReservation"), which ends the entire transaction with a failure message.

This pattern deploys separate Lambda functions for each task that is highlighted in the diagram as well as three DynamoDB tables for flights, car rentals, and payments. Each Lambda function creates, updates, or deletes the rows in the respective DynamoDB tables, depending on whether a transaction is confirmed or rolled back. The pattern uses Amazon SNS to send text (SMS) messages to subscribers, notifying them of failed or successful transactions.



Automation and scale

You can create the configuration for this architecture by using one of the IaC frameworks. Use one of the following links for your preferred IaC.

- [Deploy with AWS CDK]
- [Deploy with AWS SAM]
- [Deploy with Terraform]

Tools

AWS services

- AWS Step Functions is a serverless orchestration service that lets you combine AWS Lambda functions and other AWS services to build business-critical applications. Through the Step Functions graphical console, you see your application's workflow as a series of event-driven steps.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can use DynamoDB to create a database table that can store and retrieve any amount of data, and serve any level of request traffic.
- AWS Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.
- Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers.
- AWS Cloud Development Kit (AWS CDK) is a software development framework for defining your cloud application resources by using familiar programming languages such as TypeScript, JavaScript, Python, Java, and C#/.Net.
- AWS Serverless Application Model (AWS SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings.

Code

The code for a sample application that demonstrates the saga pattern, including the IaC template (AWS CDK, AWS SAM, or Terraform), the Lambda functions, and the DynamoDB tables can be found in the following links. Follow the instructions in the first epic to install these.

- Deploy with AWS CDK

- Deploy with AWS SAM
- Deploy with Terraform

Epics

Install packages, compile, and build

Task	Description	Skills required
Install the NPM packages.	Create a new directory, navigate to that directory in a terminal, and clone the GitHub repository of your choice from the *Code* section earlier in this pattern. In the root folder that has the `package.json` file, run the following command to download and install all Node Package Manager (NPM) packages: `npm install`	Developer, Cloud architect
Compile scripts.	In the root folder, run the following command to instruct the TypeScrip t transpiler to create all necessary JavaScript files: `npm run build`	Developer, Cloud architect
Watch for changes and recompile.	In the root folder, run the following command in a separate terminal window to watch for code changes,	Developer, Cloud architect

Task	Description	Skills required
	and compile the code when it detects a change: ```npm run watch```	
Run unit tests (AWS CDK only).	If you're using the AWS CDK, in the root folder, run the following command to perform the Jest unit tests: ```npm run test```	Developer, Cloud architect

Deploy resources to the target AWS account

Task	Description	Skills required
Deploy the demo stack to AWS.	**Important** The application is AWS Region-agnostic. If you use a profile, you must declare the Region explicitly in either the [AWS Command Line Interface (AWS CLI) profile](#) or through [AWS CLI environment variables.](#) In the root folder, run the following command to create a deployment assembly and	Developer, Cloud architect

Task	Description	Skills required

	to deploy it to the default AWS account and Region.
	
	AWS CDK:
	
	`cdk bootstrap`
	`cdk deploy`
	
	AWS SAM:
	
	`sam build`
	`sam deploy --guided`
	
	Terraform:
	
	`terraform init`
	`terraform apply`
	
	This step might take several minutes to complete.
	This command uses the default credentials that were configured for the AWS CLI.
	
	Note the API Gateway URL that is displayed on the console after deployment is complete. You will need this information to test the saga execution flow.	

Task	Description	Skills required

Compare the deployed stack with the current state.	In the root folder, run the following command to compare the deployed stack with the current state after making changes to the source code:
	
	AWS CDK:
	
	`cdk diff`
	
	AWS SAM:
	
	`sam deploy`
	
	Terraform:
	
	`terraform plan`
		Developer, Cloud architect

Test the execution flow

Task	Description	Skills required
Test the saga execution flow.	Navigate to the API Gateway URL that you noted in the earlier step, when you deployed the stack. This URL triggers the state machine to start. For more information about how to manipulate the flow of the state machine by passing different URL parameters, see the Additional information section.	Developer, Cloud architect

Task	Description	Skills required
	To view the results, sign in to the AWS Management Console and navigate to the Step Functions console. Here, you can see every step of the saga state machine. You can also view the DynamoDB table to see the records inserted, updated, or deleted. If you refresh the screen frequently, you can watch the transaction status change from `pending` to `confirmed` . You can subscribe to the SNS topic by updating the code in the `stateMachine.ts` file with your cell phone number to receive SMS messages upon successful or failed reservations. For more information, see *Amazon SNS* in the [Additional information](#) section.	

Clean up

Task	Description	Skills required
Clean up resources.	To clean up the resources deployed for this application, you can use one of the following commands.	App developer, Cloud architect

Task	Description	Skills required
	AWS CDK: `cdk destroy` AWS SAM: `sam delete` Terraform: `terraform destroy`	

Related resources

Technical papers

- Implementing Microservices on AWS
- Serverless Application Lens
- Enabling Data Persistence in Microservices

AWS service documentation

- Getting started with the AWS CDK
- Getting started with AWS SAM
- AWS Step Functions
- Amazon DynamoDB
- AWS Lambda
- Amazon API Gateway
- Amazon SNS

Tutorials

- Hands-on Workshops for Serverless Computing

Additional information

Code

For testing purposes, this pattern deploys API Gateway and a test Lambda function that triggers the Step Functions state machine. With Step Functions, you can control the functionality of the travel reservation system by passing a `run_type` parameter to mimic failures in "ReserveFlight," "ReserveCarRental," "ProcessPayment," "ConfirmFlight," and "ConfirmCarRental."

The `saga` Lambda function (`sagaLambda.ts`) takes input from the query parameters in the API Gateway URL, creates the following JSON object, and passes it to Step Functions for execution:

```
let input = {
"trip_id": tripID, //  value taken from query parameter, default is AWS request ID
"depart_city": "Detroit",
"depart_time": "2021-07-07T06:00:00.000Z",
"arrive_city": "Frankfurt",
"arrive_time": "2021-07-09T08:00:00.000Z",
"rental": "BMW",
"rental_from": "2021-07-09T00:00:00.000Z",
"rental_to": "2021-07-17T00:00:00.000Z",
"run_type": runType // value taken from query parameter, default is "success"
};
```

You can experiment with different flows of the Step Functions state machine by passing the following URL parameters:

- **Successful Execution** — https://{api gateway url}
- **Reserve Flight Fail** — https://{api gateway url}?**runType=failFlightsReservation**
- **Confirm Flight Fail** — https://{api gateway url}?**runType=failFlightsConfirmation**
- **Reserve Car Rental Fail** — https://{api gateway url}?**runType=failCarRentalReservation**
- **Confirm Car Rental Fail** — https://{api gateway url}?**runType=failCarRentalConfirmation**
- **Process Payment Fail** — https://{api gateway url}?**runType=failPayment**
- **Pass a Trip ID** — https://{api gateway url}?**tripID=**{by default, trip ID will be the AWS request ID}

IaC templates

The linked repositories include IaC templates that you can use to create the entire sample travel reservation application.

- [Deploy with AWS CDK]
- [Deploy with AWS SAM]
- [Deploy with Terraform]

DynamoDB tables

Here are the data models for the flights, car rentals, and payments tables.

```
Flight Data Model:
 var params = {
      TableName: process.env.TABLE_NAME,
      Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: flightReservationID},
        'trip_id' : {S: event.trip_id},
        'id': {S: flightReservationID},
        'depart_city' : {S: event.depart_city},
        'depart_time': {S: event.depart_time},
        'arrive_city': {S: event.arrive_city},
        'arrive_time': {S: event.arrive_time},
        'transaction_status': {S: 'pending'}
      }
    };

Car Rental Data Model:
var params = {
      TableName: process.env.TABLE_NAME,
      Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: carRentalReservationID},
        'trip_id' : {S: event.trip_id},
        'id': {S: carRentalReservationID},
        'rental': {S: event.rental},
        'rental_from': {S: event.rental_from},
        'rental_to': {S: event.rental_to},
        'transaction_status': {S: 'pending'}
      }
    };

Payment Data Model:
var params = {
      TableName: process.env.TABLE_NAME,
```

```
      Item: {
        'pk' : {S: event.trip_id},
        'sk' : {S: paymentID},
        'trip_id' : {S: event.trip_id},
        'id': {S: paymentID},
        'amount': {S: "750.00"}, // hard coded for simplicity as implementing any 
monetary transaction functionality is beyond the scope of this pattern
        'currency': {S: "USD"},
        'transaction_status': {S: "confirmed"}
      }
    };
```

Lambda functions

The following functions will be created to support the state machine flow and execution in Step Functions:

- **Reserve Flights**: Inserts a record into the DynamoDB Flights table with a `transaction_status` of `pending`, to book a flight.
- **Confirm Flight**: Updates the record in the DynamoDB Flights table, to set `transaction_status` to `confirmed`, to confirm the flight.
- **Cancel Flights Reservation**: Deletes the record from the DynamoDB Flights table, to cancel the pending flight.
- **Reserve Car Rentals**: Inserts a record into the DynamoDB CarRentals table with a `transaction_status` of `pending`, to book a car rental.
- **Confirm Car Rentals**: Updates the record in the DynamoDB CarRentals table, to set `transaction_status` to `confirmed`, to confirm the car rental.
- **Cancel Car Rentals Reservation:** Deletes the record from the DynamoDB CarRentals table, to cancel the pending car rental.
- **Process Payment**: Inserts a record into the DynamoDB Payment table for the payment.
- **Cancel Payment**: Deletes the record from the DynamoDB Payments table for the payment.

Amazon SNS

The sample application creates the following topic and subscription for sending SMS messages and notifying the customer about successful or failed reservations. If you want to receive text messages while testing the sample application, update the SMS subscription with your valid phone number in the state machine definition file.

AWS CDK snippet (add the phone number in the second line of the following code):

```
const topic = new  sns.Topic(this, 'Topic');
topic.addSubscription(new subscriptions.SmsSubscription('+11111111111'));
const snsNotificationFailure = new tasks.SnsPublish(this ,'SendingSMSFailure', {
topic:topic,
integrationPattern: sfn.IntegrationPattern.REQUEST_RESPONSE,
message: sfn.TaskInput.fromText('Your Travel Reservation Failed'),
});

const snsNotificationSuccess = new tasks.SnsPublish(this ,'SendingSMSSuccess', {
topic:topic,
integrationPattern: sfn.IntegrationPattern.REQUEST_RESPONSE,
message: sfn.TaskInput.fromText('Your Travel Reservation is Successful'),
});
```

AWS SAM snippet (replace the +1111111111 strings with your valid phone number):

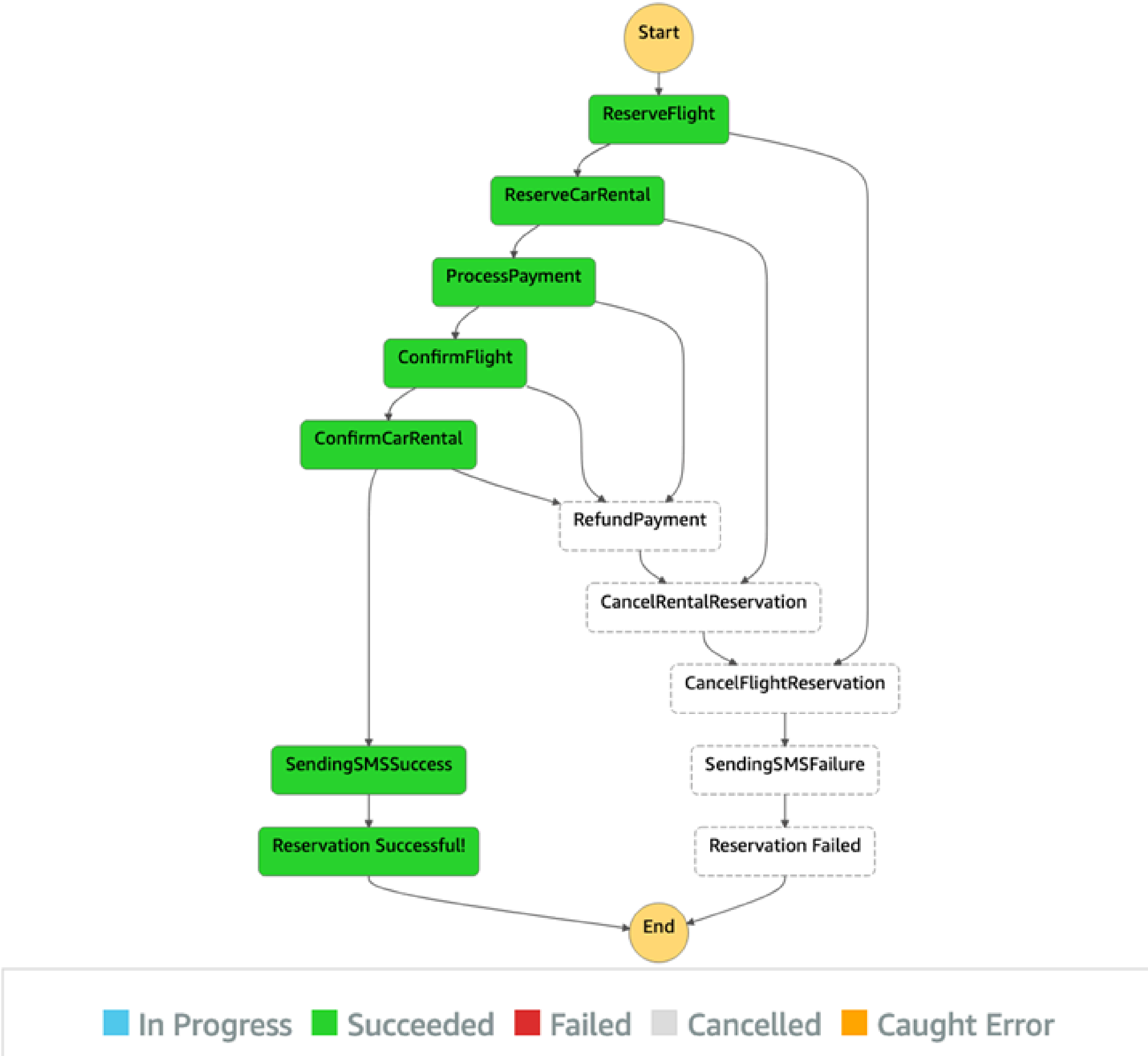
```
  StateMachineTopic11111111111:
    Type: 'AWS::SNS::Subscription'
    Properties:
      Protocol: sms
      TopicArn:
        Ref: StateMachineTopic
      Endpoint: '+11111111111'
    Metadata:
      'aws:sam:path': SamServerlessSagaStack/StateMachine/Topic/+11111111111/Resource
```

Terraform snippet (replace the +111111111 string with your valid phone number):

```
resource "aws_sns_topic_subscription" "sms-target" {
  topic_arn = aws_sns_topic.topic.arn
  protocol  = "sms"
  endpoint  = "+11111111111"
}
```

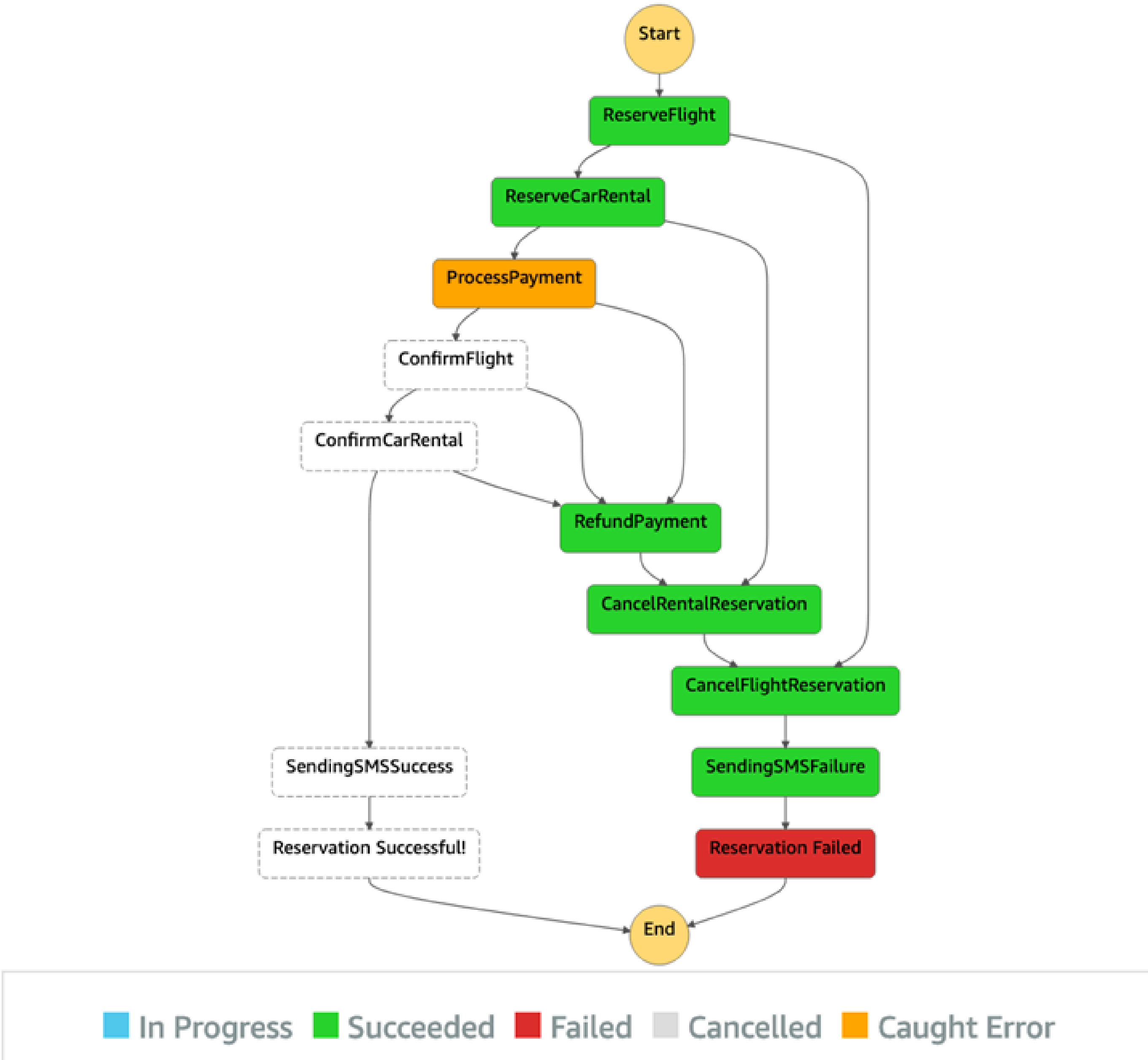
Successful reservations

The following flow illustrates a successful reservation with "ReserveFlight," "ReserveCarRental," and "ProcessPayment" followed by "ConfirmFlight" and "ConfirmCarRental." The customer is notified about the successful booking through SMS messages that are sent to the subscriber of the SNS topic.



Failed reservations

This flow is an example of failure in the saga pattern. If, after booking flights and car rentals, "ProcessPayment" fails, steps are canceled in reverse order. The reservations are released, and the customer is notified of the failure through SMS messages that are sent to the subscriber of the SNS topic.


Start
ReserveFlight
ReserveCarRental
ProcessPayment
ConfirmFlight
ConfirmCarRental
RefundPayment
CancelRentalReservation
CancelFlightReservation
SendingSMSSuccess
SendingSMSFailure
Reservation Successful!
Reservation Failed
End
In Progress
Succeeded
Failed
Cancelled
Caught Error

Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK

Dr. Rahul Sharad Gaikwad, Amazon Web Services

Summary

[Amazon ECS Anywhere](https://aws.amazon.com/ecs/anywhere/) is an extension of the Amazon Elastic Container Service (Amazon ECS). You can use ECS Anywhere to deploy native Amazon ECS tasks in an on-premises or customer-managed environment. This feature helps reduce costs and mitigate complex local container orchestration and operations. You can use ECS Anywhere to deploy and run container applications in both on-premises and cloud environments. It removes the need for your team to learn multiple domains and skill sets, or to manage complex software on their own.

This pattern demonstrates the steps to set up ECS Anywhere by using AWS Cloud Development Kit (AWS CDK) stacks.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. (See Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.)
- AWS CDK Toolkit, installed and configured. (See AWS CDK Toolkit in the AWS CDK documentation, and follow the instructions to install version 2 globally.)
- Node package manager (npm), installed and configured for the AWS CDK in TypeScript. (See Downloading and installing Node.js and npm in the npm documentation.)

Limitations

- For limitations and considerations, see External instances (Amazon ECS Anywhere) in the Amazon ECS documentation.

Product versions

- AWS CDK Toolkit version 2

- npm version 7.20.3 or later
- Node.js version 16.6.1 or later

Architecture

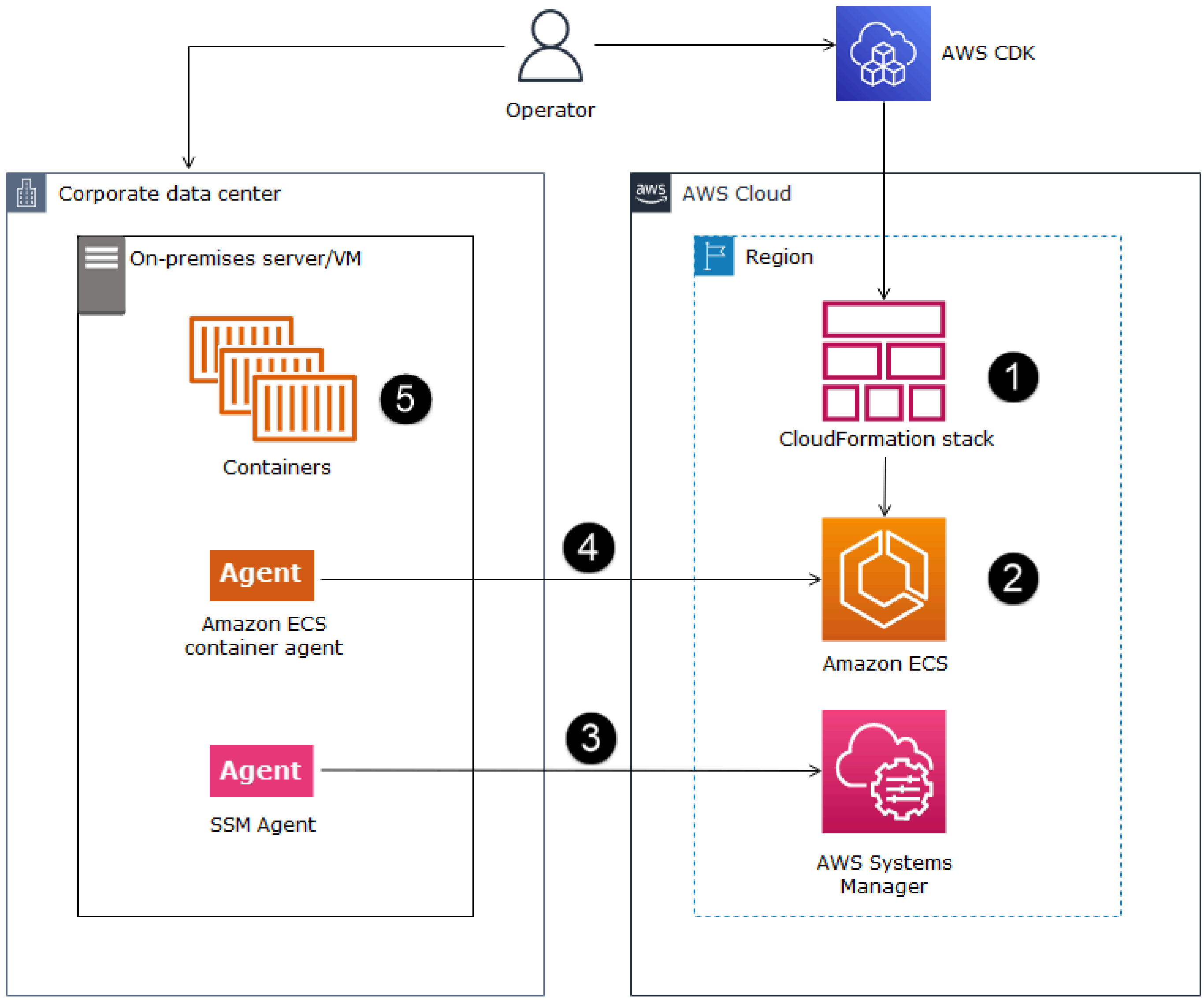
Target technology stack

- AWS CloudFormation
- AWS CDK
- Amazon ECS Anywhere
- AWS Identity and Access Management (IAM)

Target architecture

The following diagram illustrates a high-level system architecture of ECS Anywhere setup using the AWS CDK with TypeScript, as implemented by this pattern.

1. When you deploy the AWS CDK stack, it creates a CloudFormation stack on AWS.
2. The CloudFormation stack provisions an Amazon ECS cluster and related AWS resources.
3. To register an external instance with an Amazon ECS cluster, you must install AWS Systems Manager Agent (SSM Agent) on your virtual machine (VM) and register the VM as an AWS Systems Manager managed instance.
4. You must also install the Amazon ECS container agent and Docker on your VM to register it as an external instance with the Amazon ECS cluster.
5. When the external instance is registered and configured with the Amazon ECS cluster, it can run multiple containers on your VM, which is registered as an external instance.



Automation and scale

The [GitHub repository](https://github.com) that is provided with this pattern uses the AWS CDK as an infrastructure as code (IaC) tool to create the configuration for this architecture. AWS CDK helps you orchestrate resources and set up ECS Anywhere.

Tools

- [AWS Cloud Development Kit (AWS CDK)](https://aws.amazon.com/cdk/) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.

Code

The source code for this pattern is available on GitHub, in the Amazon ECS Anywhere CDK Samples repository. To clone and use the repository, follow the instructions in the next section.

Epics

Verify AWS CDK configuration

Task	Description	Skills required
Verify the AWS CDK version.	Verify the version of the AWS CDK Toolkit by running the following command: `cdk --version` This pattern requires AWS CDK version 2. If you have an earlier version of the AWS CDK, follow the instructions in the AWS CDK documentation to update it.	DevOps engineer

Set up AWS credentials.	To set up credentials, run the `aws configure` command and follow the prompts:
	`$aws configure`
	`AWS Access Key ID [None]: <your-access-key-ID>`
	`AWS Secret Access Key [None]: <your-secret-access-key>`
		DevOps engineer

Task	Description	Skills required
	`Default region name [None]: <your-Region-name>` `Default output format [None]:`	

Bootstrap the AWS CDK environment

Task	Description	Skills required
Clone the AWS CDK code repository.	Clone the GitHub code repository for this pattern by using the command: `git clone https://github.com/aws-samples/amazon-ecs-anywhere-cdk-samples.git`	DevOps engineer
Bootstrap the environment.	To deploy the AWS CloudFormation template to the account and AWS Region that you want to use, run the following command: `cdk bootstrap <account-number>/<Region>` For more information, see Bootstrapping in the AWS CDK documentation.	DevOps engineer

Build and deploy the project

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install package dependencies and compile TypeScript files.</td><td>Install the package dependencies and compile the TypeScript files by running the following commands:<pre><code>$cd amazon-ecs-anywhere-cdk-samples
$npm install
$npm fund</code></pre>These commands install all the packages from the sample repository.

Important
If you get any errors about missing packages, use one of the following commands:<pre><code>$npm ci</code></pre>—or—<pre><code>$npm install -g @aws-cdk/<package_name></code></pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	For more information, see [npm ci](#) and [npm install](#) in the npm documentation.	
Build the project.	To build the project code, run the command: `npm run build` For more information about building and deploying the project, see [Your first AWS CDK app](#) in the AWS CDK documentation.	DevOps engineer
Deploy the project.	To deploy the project code, run the command: `cdk deploy`	DevOps engineer
Verify stack creation and output.	Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation, and choose the `EcsAnywhereStack` stack. The **Outputs** tab shows the commands to run on your external VM.	DevOps engineer

Set up an on-premises machine

Task	Description	Skills required
Set up your VM by using Vagrant.	For demonstration purposes, you can use [HashiCorp](#)	DevOps engineer

Task	Description	Skills required
	Vagrant to create a VM. Vagrant is an open-source utility for building and maintaining portable virtual software development environments. Create a Vagrant VM by running the `vagrant up` command from the root directory where Vagrantfile is placed. For more information, see the Vagrant documentation.	

Task	Description	Skills required

Register your VM as an external instance.	1. Log in to the Vagrant VM by using the `vagrant ssh` command. For more information, see the [Vagrant documentation]().
	
	 2. Create an activation code and ID that you can use to register your VM with AWS Systems Manager and to activate your external instance. The output from this command includes `ActivationId` and `ActivationCode` values:
	
	 `aws ssm create-activation --iam-role EcsAnywhereInstanceRole | tee ssm-activation.json`
	
	 3. Export the activation ID and code values:
	
	 `export ACTIVATION_ID=<activation-ID>`
	 `export ACTIVATION_CODE=<activation-code>`
	
	 4. Download the installation script to your on-premises server or VM:
		DevOps engineer

Task	Description	Skills required

	```curl -o "ecs-anywhere-install.sh" "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh" && sudo chmod +x ecs-anywhere-install.sh```
	
	5. Run the installation script on your on-premises server or VM:
	
	```sudo ./ecs-anywhere-install.sh \ --cluster test-ecs-anywhere \ --activation-id $ACTIVATION_ID \ --activation-code $ACTIVATION_CODE \ --region <Region>```
	
	For more information about setting up and registering your VM, see Registering an external instance to a cluster in the Amazon ECS documentation.	

Task	Description	Skills required

Verify the status of ECS Anywhere and the external VM.	To verify whether your virtual box is connected to the Amazon ECS control plane and running, use the following commands:
	`aws ssm describe-instance-information`
	`aws ecs list-container-instances --cluster $CLUSTER_NAME`
		DevOps engineer

Clean up

Task	Description	Skills required
Clean up and delete resources .	After you walk through this pattern, you should remove the resources you created to avoid incurring any further charges. To clean up, run the command: `cdk destroy`	DevOps engineer

Related resources

- Amazon ECS Anywhere Documentation
- Amazon ECS Anywhere Demo
- Amazon ECS Anywhere Workshop Samples

Modernize ASP.NET Web Forms applications on AWS

Vijai Anand Ramalingam and Sreelaxmi Pai, Amazon Web Services

Summary

This pattern describes the steps for modernizing a legacy, monolith ASP.NET Web Forms application by porting it to ASP.NET Core on AWS.

Porting ASP.NET Web Forms applications to ASP.NET Core helps you take advantage of the performance, cost savings, and robust ecosystem of Linux. However, it can be a significant manual effort. In this pattern, the legacy application is modernized incrementally by using a phased approach, and then containerized in the AWS Cloud.

Consider a legacy, monolith application for a shopping cart. Let's assume that it was created as an ASP.NET Web Forms application and consists of .aspx pages with a code-behind (`aspx.cs`) file. The modernization process consists of these steps:

1. Break the monolith into microservices by using the appropriate decomposition patterns. For more information, see the guide [Decomposing monoliths into microservices] on the AWS Prescriptive Guidance website.
2. Port your legacy ASP.NET Web Forms (.NET Framework) application to ASP.NET Core in .NET 5 or later. In this pattern, you use Porting Assistant for .NET to scan your ASP.NET Web Forms application and identify incompatibilities with ASP.NET Core. This reduces the manual porting effort.
3. Redevelop the Web Forms UI layer by using React. This pattern doesn't cover UI redevelopment. For instructions, see [Create a New React App] in the React documentation.
4. Redevelop the Web Forms code-behind file (business interface) as an ASP.NET Core web API. This pattern uses NDepend reports to help identify required files and dependencies.
5. Upgrade shared/common projects, such as Business Logic and Data Access, in your legacy application to .NET 5 or later by using Porting Assistant for .NET.
6. Add AWS services to complement your application. For example, you can use [Amazon CloudWatch Logs] to monitor, store, and access your application's logs, and [AWS Systems Manager] to store your application settings.
7. Containerize the modernized ASP.NET Core application. This pattern creates a Docker file that targets Linux in Visual Studio and uses Docker Desktop to test it locally. This step assumes that your legacy application is already running on an on-premises or Amazon Elastic Compute Cloud

(Amazon EC2) Windows instance. For more information, see the pattern Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance.

8. Deploy the modernized ASP.NET core application to Amazon Elastic Container Service (Amazon ECS). This pattern doesn't cover the deployment step. For instructions, see the Amazon ECS Workshop.

Note

This pattern doesn't cover UI development, database modernization, or container deployment steps.

Prerequisites and limitations

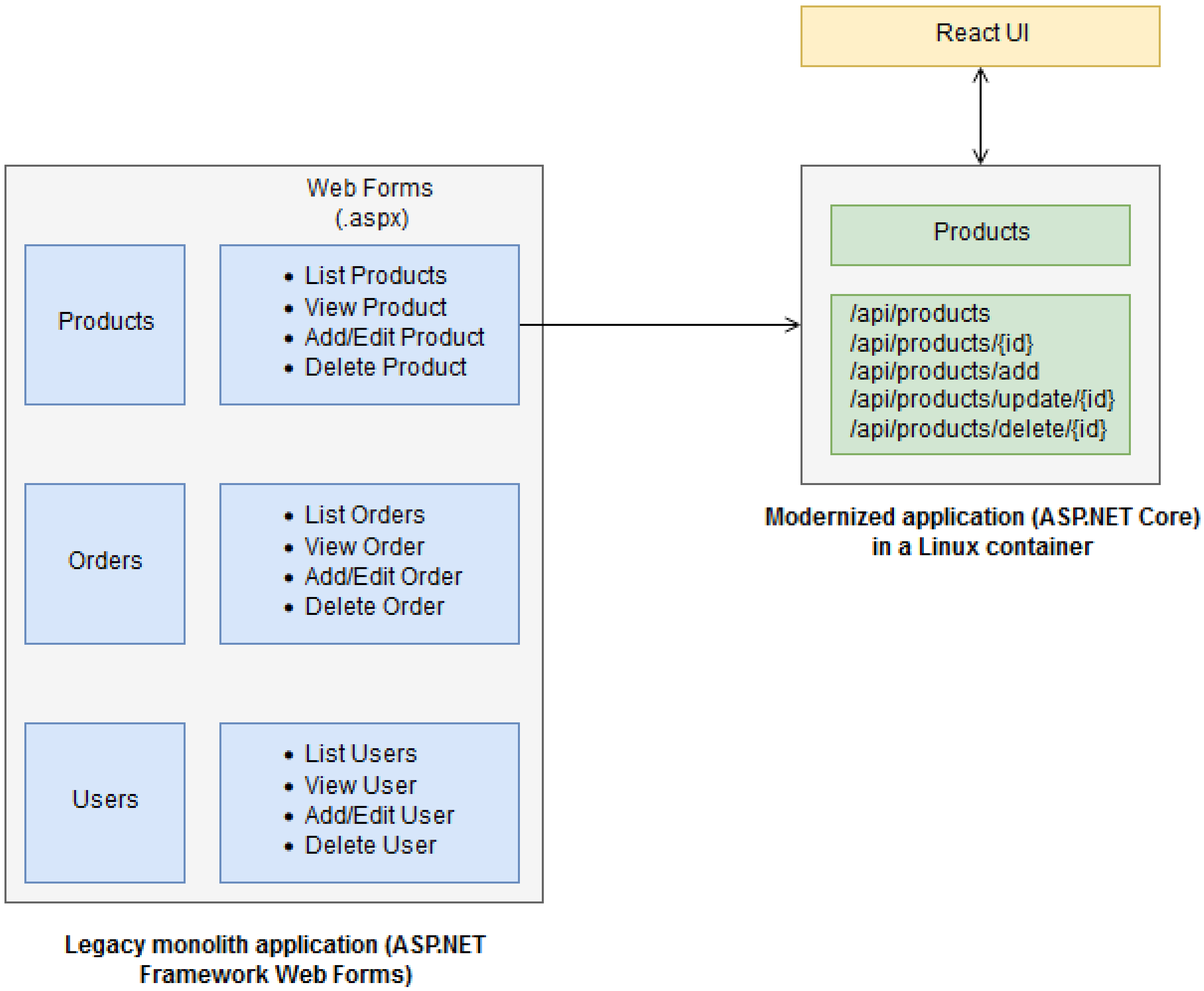
Prerequisites

- Visual Studio or Visual Studio Code, downloaded and installed.
- Access to an AWS account using the AWS Management Console and the AWS Command Line Interface (AWS CLI) version 2. (See the instructions for configuring the AWS CLI.)
- The AWS Toolkit for Visual Studio (see setup instructions).
- Docker Desktop, downloaded and installed.
- .NET SDK, downloaded and installed.
- NDepend tool, downloaded and installed. To install the NDepend extension for Visual Studio, run `NDepend.VisualStudioExtension.Installer` (see instructions). You can select Visual Studio 2019 or 2022, depending on your requirements.
- Porting Assistant for .NET, downloaded and installed.

Architecture

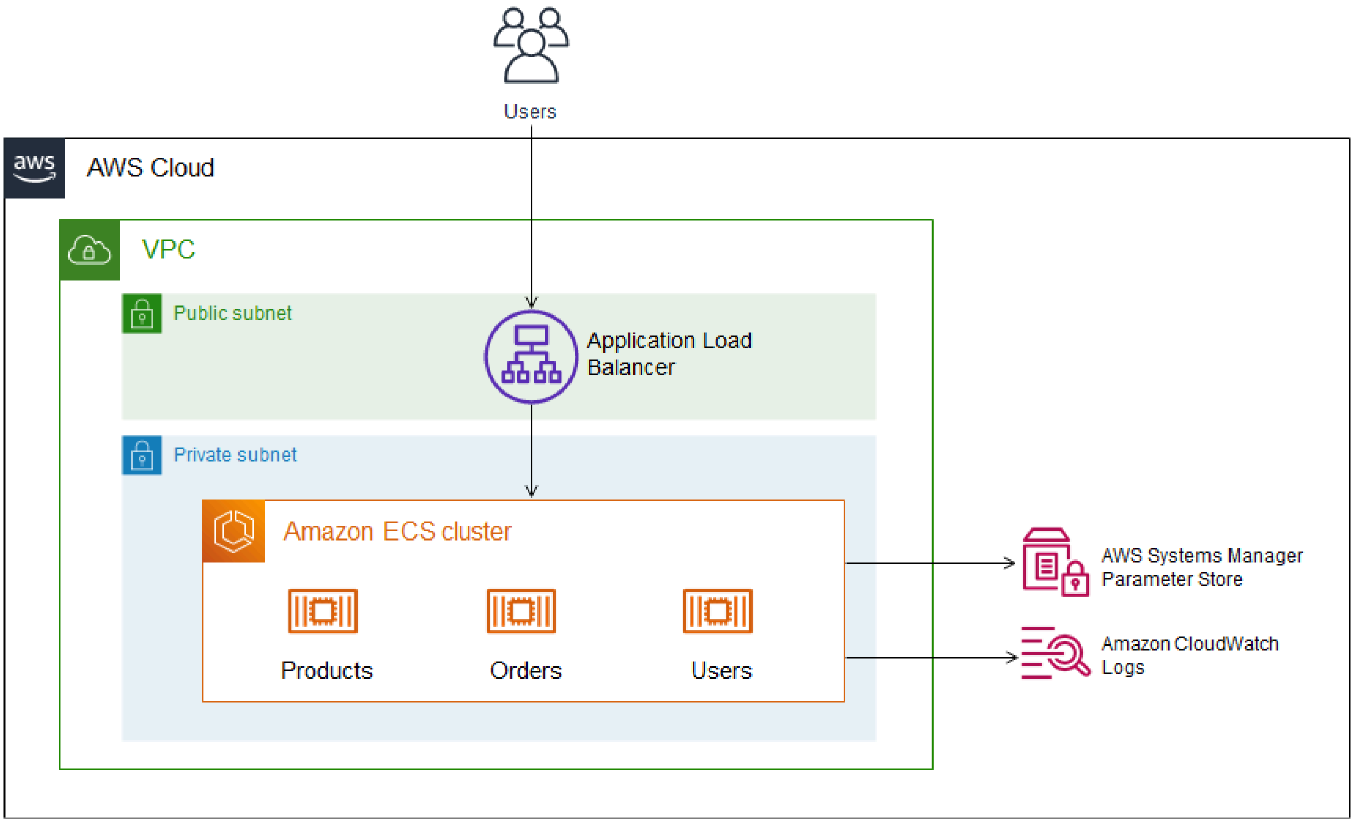
Modernizing the shopping cart application

The following diagram illustrates the modernization process for a legacy ASP.NET shopping cart application.



Target architecture

The following diagram illustrates the architecture of the modernized shopping cart application on AWS. ASP.NET Core web APIs are deployed to an Amazon ECS cluster. Logging and configuration services are provided by Amazon CloudWatch Logs and AWS Systems Manager.



Tools

AWS services

- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service for running, stopping, and managing containers on a cluster. You can run your tasks and services on a serverless infrastructure that is managed by AWS Fargate. Alternatively, for more control over your infrastructure, you can run your tasks and services on a cluster of EC2 instances that you manage.
- Amazon CloudWatch Logs – Amazon CloudWatch Logs centralizes the logs from all your systems, applications, and AWS services that you use. You can view and monitor the logs, search them for specific error codes or patterns, filter them based on specific fields, or archive them securely for future analysis.
- AWS Systems Manager — AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS

resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting (or taking corrective action) on any policy violations it detects.

Tools

- Visual Studio or Visual Studio Code – Tools for building .NET applications, web APIs, and other programs.
- AWS Toolkit for Visual Studio – An extension for Visual Studio that helps develop, debug, and deploy .NET applications that use AWS services.
- Docker Desktop – A tool that simplifies building and deploying containerized applications.
- NDepend – An analyzer that monitors .NET code for dependencies, quality issues, and code changes.
- Porting Assistant for .NET – An analysis tool that scans .NET code to identify incompatibilities with .NET Core and to estimate the migration effort.

Epics

Port your legacy application to .NET 5 or later version

Task	Description	Skills required
Upgrade your.NET Framework legacy application to .NET 5.	You can use Porting Assistant for .NET to convert your legacy ASP.NET Web Forms application to .NET 5 or later. Follow the instructions in the Porting Assistant for .NET documentation.	App developer
Generate NDepend reports.	When you modernize your ASP.NET Web Forms applicati on by decomposing it into microservices, you might not need all the .cs files from the legacy application. You can use NDepend to generate	App developer

Task	Description	Skills required
	a report for any code-behind (.cs) file, to get all the callers and callees. This report helps you identify and use only the required files in your microservices. After you install NDepend (see the [Prerequisites](#) section), open the solution (.sln file) for your legacy application in Visual Studio and follow these steps: 1. Build the legacy application in Visual Studio. 2. On the Visual Studio menu bar, choose **NDepend, Attach new NDepend project to current VS solution**. 3. Choose **Analyze .NET assemblies**. 4. When analysis is complete, navigate to the project in Solution Explorer. Right-click any code-behind file (for example, `listproducts.aspx.cs`) that you want to generate the report for, and then choose **Show on Dependency Graph**. 5. In the navigation bar, choose **Callers and callees,**	

Task	Description	Skills required
	and then choose **Edit code query**. 6. In the **Queries and Rules Edit pane**, choose the download arrow, and then choose **Export to Excel**. This process generates a report for the code-behind file that lists all callers and callees. For more information about the dependency graph, see the [NDepend documentation]().	

Task	Description	Skills required
Create a new .NET 5 solution.	To create a new .NET 5 (or later) structure for your modernized ASP.NET Core web APIs: 1. Open Visual Studio. 2. Create a new, blank solution. 3. Create new projects that target .NET 5 (or later), based on your legacy application. For examples of legacy and new projects for a shopping cart application, see the Additional information section. 4. Use the NDepend report from the previous step to identify all required files. Copy these files from the application you upgraded earlier and add them to the new solution. 5. Build the solution and fix all issues. For more information about creating projects and solutions, see the Visual Studio documentation.	App developer

Task	Description	Skills required
	Note As you build the solution and verify functionality, you might identify several additional files to be added to the solution, in addition to the files that NDepend identified.	

Update your application code

Task	Description	Skills required
Implement web APIs with ASP.NET Core.	Let's assume that one of the microservices that you identified in your legacy monolith shopping cart application is *Products*. You created a new ASP.NET Core web API project for *Products* in the previous epic. In this step, you identify and modernize all the web forms (.aspx pages) that are related to *Products*. Let's assume that *Products* consists of four web forms, as illustrated earlier in the [Architecture](#) section: • List Products • View Product	App developer

Task	Description	Skills required
	• Add/Edit Product • Delete Product You should analyze each web form, identify all the requests that are sent to the database to perform some logic, and get responses. You can implement each request as a web API endpoint. Given its web forms, *Products* can have the following possible endpoints: • `/api/products` • `/api/products/{id}` • `/api/products/add` • `/api/products/update/{id}` • `/api/products/delete/{id}` As mentioned previously, you can also reuse all the other projects that you upgraded to .NET 5, including Business Logic, Data Access, and shared/common projects.	

Task	Description	Skills required
Configure Amazon CloudWatc h Logs.	You can use Amazon CloudWatch Logs to monitor, store, and access your application's logs. You can log data into Amazon CloudWatc h Logs by using an AWS SDK. You can also integrate .NET applications with CloudWatc h Logs by using popular .NET logging frameworks such as NLog, Log4Net, and ASP.NET Core logging framework. For more information about this step, see the blog post Amazon CloudWatch Logs and .NET Logging Framework s.	App developer

Task	Description	Skills required
Configure AWS Systems Manager Parameter Store.	You can use AWS Systems Manager Parameter Store to store application settings such as connection strings separately from your application's code. The NuGet package Amazon.Extensions.Configuration.SystemsManager simplifies how your application loads these settings from the AWS Systems Manager Parameter Store into the .NET Core configuration system. For more information about this step, see the blog post .NET Core configuration provider for AWS Systems Manager.	App developer

Add authentication and authorization

Task	Description	Skills required
Use a shared cookie for authentication.	Modernizing a legacy monolith application is an iterative process and requires the monolith and its modernized version to co-exist. You can use a shared cookie to achieve seamless authentication between the two versions.	App developer

Task	Description	Skills required
	The legacy ASP.NET applicati on continues to validate user credentials and issues the cookie while the modernize d ASP.NET Core application validates the cookie. For instructions and sample code, see the [sample GitHub project]().	

Build and run the container locally

Task	Description	Skills required
Create a Docker image by using Visual Studio.	In this step, you create a Docker file by using the Visual Studio for .NET Core web API. 1. Open Visual Studio. 2. In Solution Explorer, from the context (right-click) menu of your project, choose **Add, Docker Support.** 3. Select **Linux** as the target operating system**.** Visual Studio creates a Docker file for your project. For a sample Docker file, see [Visual Studio Container Tools for Docker]() on the Microsoft website.	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Build and run the container by using Docker Desktop.</td><td>Now you can build, create and run the container in Docker Desktop.

1. Open a Command Prompt window. Navigate to the solution folder where the Docker file is located. Run the following command to create the Docker image:

<pre>docker build -t aspnetcorewebapiimage -f Dockerfile .</pre>
2. Run the following command to view all Docker images:

<pre>docker images</pre>
3. Run the following command to create and run a container:

<pre>docker run -d -p 8080:80 --name aspnetcorewebapicontainer aspnetcorewebapiimage</pre>
4. Open Docker Desktop, and then choose Containers/Apps. You can see a new container named <code>aspnetcorewebapicontainer</code> running.</td><td>App developer</td></tr>
</table>

Related resources

- Run an ASP.NET Core web API Docker container on an Amazon EC2 Linux instance (AWS Prescriptive Guidance)
- Amazon ECS Workshop
- Perform ECS blue/green deployments through CodeDeploy using AWS CloudFormation (AWS CloudFormation documentation)
- Getting started with NDepend (NDepend documentation)
- Porting Assistant for .NET

Additional information

The following tables provide examples of sample projects for a legacy shopping cart application and the equivalent projects in your modernized ASP.NET Core application.

Legacy solution:

Project name	Project template	Target framework
Business Interface	Class Library	.NET Framework
BusinessLogic	Class Library	.NET Framework
WebApplication	ASP.NET Framework Web Application	.NET Framework
UnitTests	NUnit Test Project	.NET Framework
Shared ->Common	Class Library	.NET Framework
Shared ->Framework	Class Library	.NET Framework

New solution:

Project name	Project template	Target framework
BusinessLogic	Class Library	.NET 5.0

<WebAPI>	ASP.NET Core Web API	.NET 5.0
<WebAPI>.UnitTests	NUnit 3 Test Project	.NET 5.0
Shared ->Common	Class Library	.NET 5.0
Shared ->Framework	Class Library	.NET 5.0

Run event-driven and scheduled workloads at scale with AWS Fargate

HARI OHM PRASATH RAJAGOPAL, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern describes how to run scheduled and event-driven workloads at scale on the Amazon Web Services (AWS) Cloud by using AWS Fargate.

In the use case that this pattern sets up, code is scanned for AWS sensitive information, such as the AWS account number and credentials, whenever a pull request is submitted. The pull request initiates a Lambda function. The Lambda function invokes a Fargate task that takes care of the code scan. Lambda is initiated whenever a new pull request is raised. If the scan finds any sensitive information, Amazon Simple Notification Service (Amazon SNS) sends the scan results in an email message.

This pattern is helpful in the following business use cases:

- If your business must run many scheduled and event-driven workloads that cannot be run by AWS Lambda because of limitations around runtime (a 15-minute limit) or memory
- If you want AWS to manage the instances provisioned for these workloads

When you use this pattern, you have the option of creating a new virtual private cloud (VPC). This pattern also uses AWS CodeCommit.

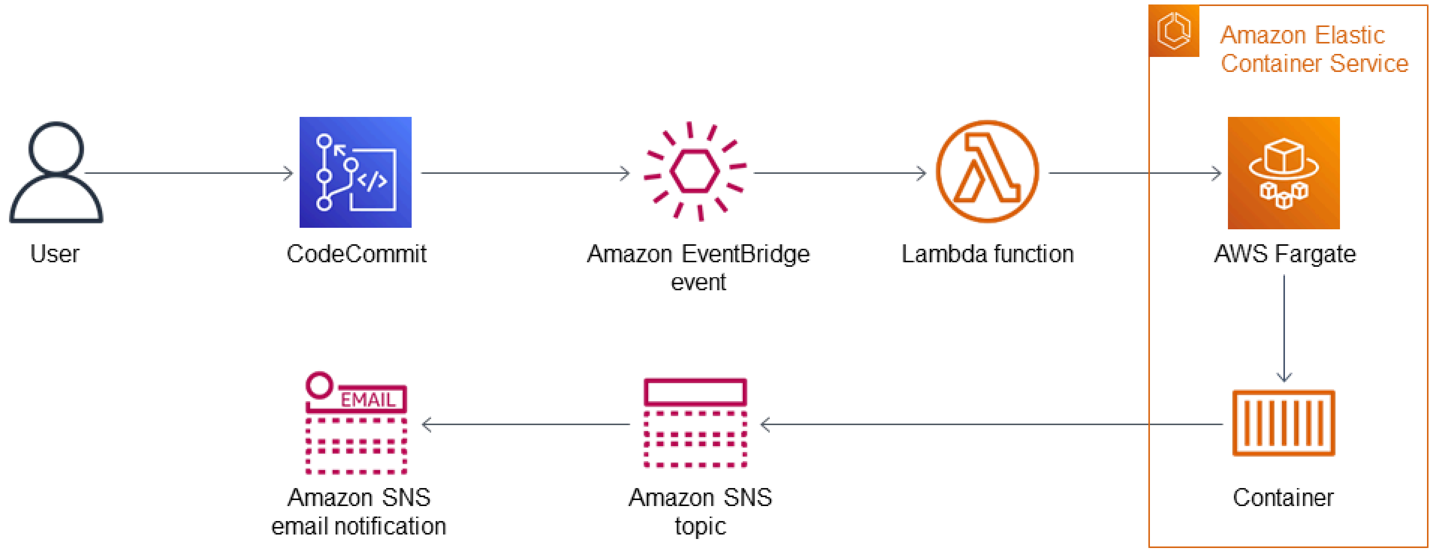
Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS CodeCommit for hosting the code base and creating pull requests
- AWS Command Line Interface (AWS CLI) version 1.7 or later, installed and configured on macOS, Linux, or Windows
- Workloads running in containers

- Apache Maven executable set up in classpath

Architecture



The overall flow includes the following steps.

1. Whenever a new pull request is submitted in CodeCommit, a Lambda function is initiated. The Lambda function listens through the `CodeCommit Pull Request State Change` event through Amazon EventBridge.
2. The Lambda function submits a new Fargate task with the following environment parameters for checking out the code and scanning it.

```
RUNNER # <<TaskARN>>
SNS_TOPIC # <<SNSTopicARN>>
SUBNET # <<Subnet in which Fargate task gets launched>>
```

   If the scan finds sensitive information in the code, Fargate pushes a new message to the Amazon SNS topic.
3. An SNS subscriber reads the message from the topic and sends an email message.

Technology

- AWS CodeCommit
- Amazon Elastic Container Registry (Amazon ECR)

- Amazon Elastic Container Service (Amazon ECS)
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon SNS
- Docker

Tools

Tools

- AWS CLI – AWS Command Line Interface (CLI) is a unified tool to manage your AWS services.
- AWS CodeCommit – AWS CodeCommit is a fully managed source control service that hosts secure Git-based repositories. Using CodeCommit, teams can collaborate on code in a secure and highly scalable environment.
- Amazon ECR – Amazon Elastic Container Registry (Amazon ECR) is a fully managed registry that developers can use to store, manage, and deploy Docker container images.
- Amazon ECS – Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service. You can use Amazon ECS to run, stop, and manage containers on a cluster.
- AWS Fargate – AWS Fargate is a technology that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel. Clients that subscribe to the SNS topic receive published messages using a supported protocol, such Lambda, email, mobile push notifications, and mobile text messages (SMS).
- Docker – Docker helps you build, test, and deliver applications in packages called containers.
- Git client – Command line or desktop tool to check out the required artifacts
- Maven – Apache Maven is a project management tool for centrally managing a project's build, reporting, and documentation.

Epics

Set up the local repository

Task	Description	Skills required
Download the code.	In the **Attachments** section, download the .zip file and extract the files.	Developer, AWS system administrator
Set up the repo.	Run `mvn clean install` on the root folder.	Developer, AWS system administrator

Create an Amazon ECR image and push the image

Task	Description	Skills required
Create an Amazon ECR repository and log in.	Open the Amazon ECR console. In the navigatio n pane, choose **Repositor ies**, and then choose **Create repository**. For help with this and other stories, see the **Related resources** section.	Developer, AWS system administrator
Push your container image.	Open the repository, choose **View push commands**, and log in to Docker. After you are logged in, run the commands, with the required substitut ions, that are under **Push the container image** in the **Additional information** section. This uploads the Docker container image that is used to perform code scanning. After the upload	Developer, AWS system administrator

Task	Description	Skills required
	is complete, copy the URL of the latest build in the Amazon ECR repository.	

Create the CodeCommit repository

Task	Description	Skills required
Create the CodeCommit repository.	To create a new AWS CodeCommit repository, run the command under **Create the CodeCommit repository** in the **Additional informati on** section.	Developer, AWS system administrator

Create the VPC (optional)

Task	Description	Skills required
Create a VPC.	If you want to use a new VPC rather than an existing one, run the commands under **Create a VPC** in the **Additional information** section. The AWS Cloud Development Kit (AWS CDK) script will output the IDs of the VPC and subnet that were created.	Developer, AWS system administrator

Create the Amazon ECS cluster and Fargate task

Task	Description	Skills required
Create the cluster and the task.	To create an Amazon ECS cluster and Fargate task definition, run the commands under **Create the cluster and task** in the **Additional information** section. Make sure that the correct VPC ID and Amazon ECR repo URI are passed in as a parameter while running the shell script. The script creates a Fargate task definition that points to the Docker image (responsible for scanning). The script then creates a job and an associate d execution role.	Developer, AWS system administrator
Verify the Amazon ECS cluster.	Open the Amazon ECS console. In the navigation pane, choose **Clusters**, and choose the newly created Amazon ECS cluster named **Fargate-Job-Cluster**. After this, choose **Task definition** in the navigation pane, and confirm that there is a new task definition with the prefix `awscdkfargateecsTa skDef` .	Developer, AWS system administrator

Create the SNS topic and subscriber

Task	Description	Skills required
Create an SNS topic.	To create an SNS topic, run the command under **Create the SNS topic** in the **Additional informati on** section. After creation is successful, note the `SNS ARN`, which is used in the next step.	Developer, AWS system administrator
Create the SNS subscriber.	To create an email subscribe r for the SNS topic, run the command under **Create the SNS subscriber** in the **Additional information** section. Make sure to replace `TopicARN` and `Email address` used in the CLI command. To receive email notifications, make sure to confirm the email address that is used as a subscriber.	Developer, AWS system administrator

Create the Lambda function and CodeCommit trigger

Task	Description	Skills required
Create the function and the trigger.	To create a Lambda function with a CodeCommit trigger, run the command under **Lambda function and CodeCommit trigger** in the **Additional information** section. Make sure to replace	Developer, AWS system administrator

Task	Description	Skills required
	the parameters with the corresponding values before running the command. The script creates the Lambda function and configures it to be invoked when a new pull request is made.	

Test the application

Task	Description	Skills required
Test the application.	If you check in any AWS sensitive information to CodeCommit repo, the Lambda function should be initiated. The Lambda function initiates the Fargate task, which scans the code and sends the scan results in an email notification.	Developer, AWS system administrator

Related resources

- Creating an Amazon ECR repository
- Pushing Docker images to Amazon ECR

Additional information

Push the container image

```
> cd 1-ecr-image-push
> ./run.sh <<ecr-repository>>
```

Create the CodeCommit repository

```
aws codecommit create-repository --repository-name test-repo --repository-description
 "My Test repository"
```

Create a VPC

```
> cd 2-create-vpc
> ./run.sh
```

Output

```
aws-batch-cdk-vpc-efs-launch-template.privatesubnet = subnet-<<id>>
aws-batch-cdk-vpc-efs-launch-template.publicsubnet = subnet-<<id>>
aws-batch-cdk-vpc-efs-launch-template.vpcid = vpc-<<id>>
```

Create the cluster and task

```
> export CDK_DEFAULT_ACCOUNT = <<aws_account_id>>
> export CDK_DEFAULT_REGION = <<aws_region>>
> cd 3-create-ecs-task
> ./run.sh <<vpc-id>> <<ecr-repo-uri>>
```

Output

```
aws-cdk-fargate-ecs.CLUSTERNAME = Fargate-Job-Cluster
aws-cdk-fargate-ecs.ClusterARN = <<cluster_arn>>
aws-cdk-fargate-ecs.ContainerARN = Fargate-Container
aws-cdk-fargate-ecs.TaskARN = <<task_arn>>
aws-cdk-fargate-ecs.TaskExecutionRole = <<execution_role_arn>>
aws-cdk-fargate-ecs.TaskRole = <<task_role_arn>>
```

Create the SNS topic

```
aws sns create-topic --name code-commit-topic
```

Create the SNS subscriber

```
aws sns subscribe \
```

```
    --topic-arn <<topic_arn>> \
    --protocol email \
    --notification-endpoint <<email_address>>
```

Lambda function and CodeCommit trigger

```
> export CDK_DEFAULT_ACCOUNT = <<aws_account_id>>
> export CDK_DEFAULT_REGION = <<aws_region>>
> cd 5-Lambda-CodeCommit-Trigger
> ./run.sh <<taskarn>> <<snstopicarn>> subnet-<<id>> <<codecommitarn>>
```

Output

```
aws-cdk-fargate-lambda-event.Cloudwatchrule = <<cloudwatchrule>>
aws-cdk-fargate-lambda-event.CodeCommitLambda = AWS-Code-Scanner-Function
aws-cdk-fargate-lambda-event.LambdaRole = <<lambdaiamrole>>
```

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Tenant onboarding in SaaS architecture for the silo model using C# and AWS CDK

Tabby Ward, Susmitha Reddy Gankidi, and Vijai Anand Ramalingam, Amazon Web Services

Summary

Software as a service (SaaS) applications can be built with a variety of different architectural models. The *silo model* refers to an architecture where tenants are provided dedicated resources.

SaaS applications rely on a frictionless model for introducing new tenants into their environment. This often requires the orchestration of a number of components to successfully provision and configure all the elements needed to create a new tenant. This process, in SaaS architecture, is referred to as tenant on-boarding. On-boarding should be fully automated for every SaaS environment by utilizing infrastructure as code in your on-boarding process.

This pattern guides you through an example of creating a tenant and provisioning a basic infrastructure for the tenant on Amazon Web Services (AWS). The pattern uses C# and the AWS Cloud Development Kit (AWS CDK).

Because this pattern creates a billing alarm, we recommend deploying the stack in the US East (N. Virginia), or us-east-1, AWS Region. For more information, see the AWS documentation.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An AWS Identity and Access Management (IAM) principal with sufficient IAM access to create AWS resources for this pattern. For more information, see IAM roles.
- Install Amazon Command Line Interface (AWS CLI) and configure AWS CLI to perform AWS CDK deployment.
- Visual Studio 2022 downloaded and installed or Visual Studio Code downloaded and installed.
- AWS Toolkit for Visual Studio set up.
- .NET Core 3.1 or later (required for C# AWS CDK applications)
- Amazon.Lambda.Tools installed.

Limitations

- AWS CDK uses AWS CloudFormation, so AWS CDK applications are subject to CloudFormation service quotas. For more information, see AWS CloudFormation quotas.
- The tenant CloudFormation stack is created with a CloudFormation service role `infra-cloudformation-role` with wildcard characters on actions (`sns*` and `sqs*`) but with resources locked down to the `tenant-cluster` prefix. For a production use case, evaluate this setting and provide only required access to this service role. The `InfrastructureProvision` Lambda function also uses a wildcard character (`cloudformation*`) to provision the CloudFormation stack but with resources locked down to the `tenant-cluster` prefix.
- This example code's docker build uses `--platform=linux/amd64` to force `linux/amd64` based images. This is to ensure that the final image artifacts will be suitable for Lambda, which by default uses x86-64 architecture. If you need to change the target Lambda architecture, be sure to change both the Dockerfiles and the AWS CDK codes. For more information, see the blog post Migrating AWS Lambda functions to Arm-based AWS Graviton2 processors.
- The stack deletion process will not clean up CloudWatch Logs (log groups and logs) generated by the stack. You must manually clean up the logs through the AWS Management Console Amazon CloudWatch console or the through the API.

This pattern is set up as an example. For production use, evaluate the following setups and make changes based on your business requirements:

- The AWS Simple Storage Service (Amazon S3) bucket in this example does not have versioning enabled for simplicity. Evaluate and update the setup as needed.
- This example sets up Amazon API Gateway REST API endpoints without authentication, authorization, or throttling for simplicity. For production use, we recommend integrating the system with the business security infrastructure. Evaluate this setting and add required security settings as needed.
- For this tenant infrastructure example, Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Queue Service (Amazon SQS) have only minimum setups. The AWS Key Management Service (AWS KMS) for each tenant opens to Amazon CloudWatch and Amazon SNS services in the account to consume based on the AWS KMS key policy. The setup is only an example placeholder. Adjust the setups as needed based on your business use case.
- The entire setup, which includes but isn't limited to API endpoints and backend tenant provisioning and deletion by using AWS CloudFormation, covers only the basic happy path case. Evaluate and update the setup with the necessary retry logic, additional error handling logic, and security logic based on your business needs.

- The example code is tested with up-to-date cdk-nag to check for policies at the time of this writing. New policies might be enforced in the future. These new policies might require you to manually modify the stack based on the recommendations before the stack can be deployed. Review the existing code to ensure that it aligns with your business requirements.
- The code relies on the AWS CDK to generate a random suffix instead of relying on static assigned physical names for most created resources. This setup is to ensure that these resources are unique and do not conflict with other stacks. For more information, see the AWS CDK documentation. Adjust this based on your business requirements.
- This example code packages .NET Lambda artifacts into Docker based images and runs with the Lambda provided Container image runtime. The container image runtime has advantages for standard transfer and store mechanisms (container registries) and more accurate local test environments (through the container image). You can switch the project to use Lambda provided .NET runtimes to reduce the build time of the Docker images, but you will then need to set up transfer and store mechanisms and ensure that the local setup matches the Lambda setup. Adjust the code to align with users' business requirements.

Product versions

- AWS CDK version 2.45.0 or later
- Visual Studio 2022

Architecture

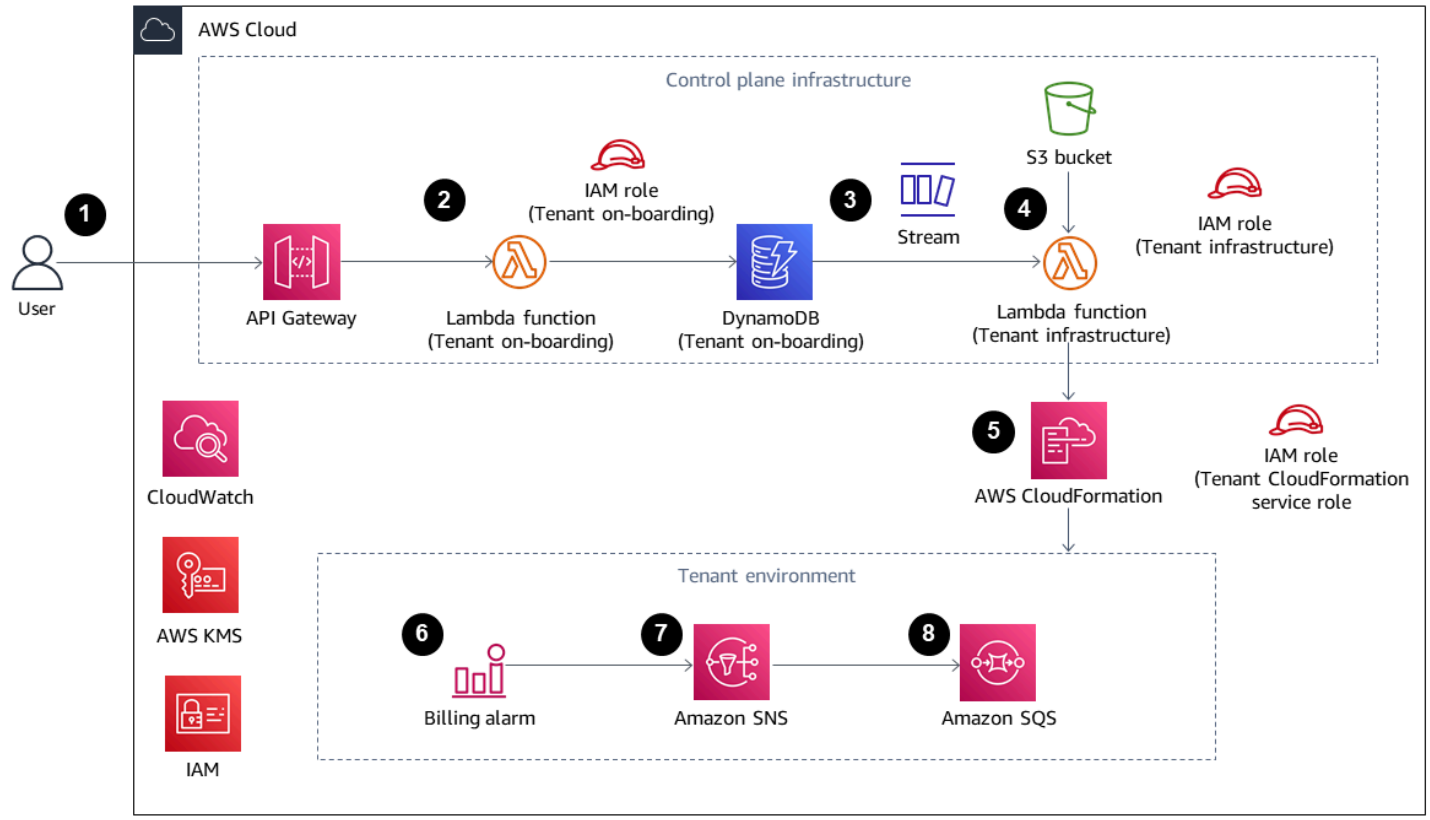
Technology stack

- Amazon API Gateway
- AWS CloudFormation
- Amazon CloudWatch
- Amazon DynamoDB
- AWS Identity and Access Management (IAM)
- AWS KMS
- AWS Lambda
- Amazon S3
- Amazon SNS

- Amazon SQS

Architecture

The following diagram shows the tenant stack creation flow. For more information about the control-plane and tenant technology stacks, see the *Additional information* section.



Tenant stack creation flow

1. User sends a POST API request with new tenant payload (tenant name, tenant description) in JSON to a REST API hosted by Amazon API Gateway. The API Gateway processes the request and forwards it to the backend Lambda Tenant On-boarding function. In this example, there is no authorization or authentication. In a production setup, this API should be integrated with the SaaS infrastructure security system.
2. The Tenant On-boarding function verifies the request. Then it attempts to store the tenant record, which includes the tenant name, generated tenant universally unique identifier (UUID), and tenant description, into the Amazon DynamoDB Tenant On-boarding table.
3. After DynamoDB stores the record, a DynamoDB stream initiates the downstream Lambda Tenant Infrastructure function.

4. The Tenant Infrastructure Lambda function acts based the on received DynamoDB stream. If the stream is for the INSERT event, the function uses the stream's NewImage section (latest update record, Tenant Name field) to invoke CloudFormation to create a new tenant infrastructure using the template that is stored in the S3 bucket. The CloudFormation template requires the Tenant Name parameter.
5. AWS CloudFormation creates the tenant infrastructure based on the CloudFormation template and input parameters.
6. Each tenant infrastructure setup has a CloudWatch alarm, a billing alarm, and an alarm event.
7. The alarm event becomes a message to an SNS topic, which is encrypted by the tenant's AWS KMS key.
8. The SNS topic forwards the received alarm message to the SQS queue, which is encrypted by the tenant's AWS KMS for encryption key.

Other systems can be integrated with Amazon SQS to perform actions based on messages in queue. In this example, to keep the code generic, incoming messages remain in queue and require manual deletion.

Tenant stack deletion flow

1. User sends a DELETE API request with new tenant payload (tenant name, tenant description) in JSON to the REST API hosted by Amazon API Gateway, which will process the request and forward to Tenant On-boarding function. In this example, there is no authorization or authentication. In a production setup, this API will be integrated with the SaaS infrastructure security system.
2. The Tenant On-boarding function will verify the request and then attempt to delete the tenant record (tenant name) from the Tenant On-boarding table.
3. After DynamoDB deletes the record successfully (the record exists in the table and is deleted), a DynamoDB stream initiates the downstream Lambda Tenant Infrastructure function.
4. The Tenant Infrastructure Lambda function acts based on the received DynamoDB stream record. If the stream is for the REMOVE event, the function uses the record's OldImage section (record information and Tenant Name field, before the latest change, which is delete) to initiate deletion of an existing stack based on that record information.
5. AWS CloudFormation deletes the target tenant stack according to the input.

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CDK Toolkit is a command line cloud development kit that helps you interact with your AWS Cloud Development Kit (AWS CDK) app.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- AWS Toolkit for Visual Studio is a plugin for the Visual Studio integrated development environment (IDE). The Toolkit for Visual Studio supports developing, debugging, and deploying .NET applications that use AWS services.

Other tools

- Visual Studio is an IDE that includes compilers, code completion tools, graphical designers, and other features that support software development.

Code

The code for this pattern is in the Tenant onboarding in SaaS Architecture for Silo Model APG Example repository.

Epics

Set up AWS CDK

Task	Description	Skills required
Verify Node.js installation.	To verify that Node.js is installed on your local machine, run the following command. `node --version`	AWS administrator, AWS DevOps
Install AWS CDK Toolkit.	To install AWS CDK Toolkit on your local machine, run the following command. `npm install -g aws-cdk` If npm is not installed, you can install it from the Node.js site.	AWS administrator, AWS DevOps
Verify the AWS CDK Toolkit version.	To verify that the AWS CDK Toolkit version is installed correctly on your machine, run the following command.	AWS administrator, AWS DevOps

Task	Description	Skills required
	`cdk --version`	

Review the code for the tenant onboarding control plane

Task	Description	Skills required
Clone the repository.	Clone the [repository](), and navigate to the `\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example` folder. In Visual Studio 2022, open the `\src\TenantOnboardingInfra.sln` solution. Open the `TenantOnboardingInfraStack.cs` file and review the code. The following resources are created as part of this stack: • DynamoDB table • S3 bucket (Upload the CloudFormation template to the S3 bucket.) • Lambda execution role • Lambda function • API Gateway API • Event source to Lambda function	AWS administrator, AWS DevOps

Task	Description	Skills required
Review the CloudFormation template.	In the `\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\template` folder, open `infra.yaml`, and review the CloudFormation template. This template will be hydrated with the tenant name retrieved from the tenant onboarding DynamoDB table. The template provisions the tenant-specific infrastructure. In this example, it provisions the AWS KMS key, Amazon SNS , Amazon SQS, and the CloudWatch alarm.	App developer, AWS DevOps

Task	Description	Skills required
Review the tenant onboardin g function.	Open `Function.cs` , and review the code for the tenant onboarding function, which is created with the Visual Studio AWS Lambda Project (.NET Core- C#) template with the .NET 6 (Container Image) blueprint. Open the `Dockerfile` , and review the code. The `Dockerfile` is a text file that consists of instructions for building the Lambda container image. Note that the following NuGet packages are added as dependencies to the `TenantOnboardingFunction` project: • `Amazon.Lambda.APIGatewayEvents` • `AWSSDK.DynamoDBv2` • `Newtonsoft.Json`	App developer, AWS DevOps

Task	Description	Skills required
Review the Tenant InfraProvisioning function.	Navigate to `\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\src\InfraProvisioningFunction` . Open `Function.cs` , and review the code for the tenant infrastructure provisioning function, which is created with the Visual Studio AWS Lambda Project (.NET Core- C#) template with the .NET 6 (Container Image) blueprint. Open the `Dockerfile` , and review the code. Note that the following NuGet packages are added as dependencies to the `InfraProvisioningFunction` project: • `Amazon.Lambda.DynamoDBEvents` • `AWSSDK.DynamoDBv2` • `AWSSDK.Cloudformation`	App developer, AWS DevOps

Deploy the AWS resources

Task	Description	Skills required
Build the solution.	To build the solution, perform the following steps: 1. In Visual Studio 2022, open the `\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\src\TenantOnboardingInfra.sln` solution. 2. Open the context (right-click) menu for the solution, and choose **Build solution**. **Note** Make sure that you update the `Amazon.CDK.Lib` `NuGet` package to the latest version in `\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example\src\TenantOnboardingInfra`	App developer

Task	Description	Skills required
	project before you build the solution.	

Bootstrap the AWS CDK environment.	Open the Windows command prompt and navigate to the AWS CDK app root folder where the `cdk.json` file is available (`\tenant-onboarding-in-saas-architecture-for-silo-model-apg-example`). Run the following command for bootstrapping.
	`cdk bootstrap`
	If you have created an AWS profile for the credentials, use the command with your profile.
	`cdk bootstrap --profile <profile name>`
		AWS administrator, AWS DevOps

Task	Description	Skills required

List the AWS CDK stacks.	To list all the stacks to be created as part of this project, run the following command.
	`cdk ls`
	`cdk ls --profile <profile name>`
	If you have created an AWS profile for the credentials, use the command with your profile.
	`cdk ls --profile <profile name>`
		AWS administrator, AWS DevOps

Review which AWS resources will be created.	To review all the AWS resources that will be created as part of this project, run the following command.
	`cdk diff`
	If you have created an AWS profile for the credentials, use the command with your profile.
	`cdk diff --profile <profile name>`
		AWS administrator, AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy all the AWS resources by using AWS CDK.</td><td>To deploy all the AWS resources run the following command.

```
cdk deploy --all --require-approval never
```

If you have created an AWS profile for the credentials, use the command with your profile.

```
cdk deploy --all --require-approval never --profile <profile name>
```

After the deployment is complete, copy the API URL from the outputs section in the command prompt, which is shown in the following example.

```
Outputs:
TenantOnboardingInfraStack.TenantOnboardingAPIEndpoint42E526D7 = https://j2qmp8ds21i1i.execute-api.us-west-2.amazonaws.com/prod/
```
</td><td>AWS administrator, AWS DevOps</td></tr>
</table>

Verify the functionality

Task	Description	Skills required

Create a new tenant.	To create the new tenant, send the following curl request.
	`curl -X POST <TenantOnboardingAPIEndpoint* from CDK Output>tenant -d '{"Name":"Tenant123", "Description":"Stack for Tenant123"}'`
	Change the place holder `<TenantOnboardingAPIEndpoint* from CDK Output>` to the actual value from AWS CDK, as shown in the following example.
	`curl -X POST https://j2qmp8ds21i1i.execute-api.us-west-2.amazonaws.com/prod/tenant -d '{"Name":"Tenant123", "Description":"test12"}'`
	The following example shows the output.
	`{"message": "A new tenant added - 5/4/2022 7:11:30 AM"}`
		App developer, AWS administrator, AWS DevOps

Task	Description	Skills required
Verify the newly created tenant details in DynamoDB.	To verify the newly created tenant details in DynamoDB, perform the following steps. 1. Open AWS Management Console, and navigate to the Amazon DynamoDB service. 2. In the left navigation, choose **Explore items**, and choose the `TenantOnboarding` table. **Note** The tenant name will be prepended with `tenantcluster-`. For more information, see the *Additional information* section. 3. Verify that a new item is created with the tenant details.	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required
Verify the stack creation for the new tenant.	Verify that the new stack was successfully created and provisioned with infrastru cture for the newly created tenant according to the CloudFormation template. 1. Open the CloudFormation console. 2. In the left navigation, choose **Stacks**, and verify that a stack with the tenant name was successfu lly created. 3. Choose the newly created tenant stack, and then choose the **Resources** tab. Note the alarm resource and the Amazon SQS resource. 4. Open a new terminal with AWS credentials configure d, and point to the correct Region. To raise a test alarm, enter the following code, replacing `<alarm resource name>` with the alarm resource name noted in step 3. `aws cloudwatch set-alarm-state --alarm-name <alarm resource name> --state-value`	App developer, AWS administrator, AWS DevOps

Task	Description	Skills required

	<pre>ALARM --state-reason
	'Test setup'</pre>
	The following example shows the code with an alarm resource name.
	<pre>aws cloudwatch set-
	alarm-state --alarm-
	name tenantcluster-
	tenant123-alarm --
	state-value ALARM --
	state-reason 'Test
	 setup'</pre>
	5. Open the console and navigate to the Amazon SQS console. Choose the Amazon SQS resource name identified in step 3. Follow the [AWS documentation instructions](#) to receive and delete the test message from the alarm that was raised in step 4.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Delete the tenant stack.</td><td>

To delete the tenant stack, send the following curl request.

```
curl -X DELETE <TenantOnboardingAPIEndpoint* from CDK Output>tenant/<Tenant Name from previous step>
```

Change the place holder `<TenantOnboardingAPIEndpoint* from CDK Output>` to the actual value from AWS CDK, and change `<Tenant Name from previous step>` to the actual value from the previous tenant creation step, as shown in the following example.

```
curl -X DELETE https://j2qmp8ds21i1i.execute-api.us-west-2.amazonaws.com/prod/tenant/Tenant123
```

The following example shows the output.

```
{"message": "Tenant destroyed - 5/4/2022 7:14:48 AM"}
```

</td><td>App developer, AWS DevOps, AWS administrator</td></tr>
</table>

Task	Description	Skills required
Verify the stack deletion for the existing tenant.	To verify that the existing tenant stack got deleted, perform the following steps: 1. Open console and navigate to the CloudFormation console. 2. In the left navigation, verify that the existing stack with the tenant name is no longer in console (if the CloudFormation console is set up to show only **Active** stacks) or is in the process of being deleted. If the stack is no longer in the CloudForm ation console, use the dropdown list to change the console's setting from **Active** to **Deleted** to see the deleted stack and verify that the stack was deleted successfully.	App developer, AWS administrator, AWS DevOps

Clean up

Task	Description	Skills required
Destroy the environment.	Before the stack clean up, ensure the following: • All records in DynamoDB are removed either through	AWS administrator, AWS DevOps

Task	Description	Skills required
	the previous tenant deletion operation or through the DynamoDB console or API. Each tenant record deletion will initiate the cleanup of its AWS CloudFormation counterpa rt. • All tenant-based AWS CloudFormation stacks are cleaned up (in case the DynamoDB trigger cleanup logic fails) on the AWS CloudFormation console. After testing is done, AWS CDK can be used to destroy all the stacks and related resources by running the following command. `cdk destroy --all;` If you created an AWS profile for the credentials, use the profile. Confirm the stack deletion prompt to delete the stack.	

Task	Description	Skills required
Clean up Amazon CloudWatch Logs.	The stack deletion process will not clean up CloudWatc h Logs (log groups and logs) that were generated by the stack. Manually clean up the CloudWatch resources by using the CloudWatch console or the API.	App developer, AWS DevOps, AWS administrator

Related resources

- [AWS CDK .NET Workshop]
- [Working with the AWS CDK in C#]
- [CDK .NET Reference]

Additional information

Control-plane technology stack

The CDK code written in .NET is used to provision the control-plane infrastructure, which consists of the following resources:

1. **API Gateway**

   Serves as the REST API entry point for the control-plane stack.

2. **Tenant on-boarding Lambda function**

   This Lambda function is initiated by API Gateway using the m method.

   A POST method API request results in (`tenant name`, `tenant description`) being inserted into the DynamoDB `Tenant Onboarding` table.

   In this code example, the tenant name is also used as part of the tenant stack name and the names of resources within that stack. This is to make these resources easier to identify. This tenant name must be unique across the setup to avoid conflicts or errors. Detailed input validation setup is explained in the IAM roles documentation and the *Limitations* section.

The persistence process to the DynamoDB table will succeed only if the tenant name is not used in any other record in the table.

The tenant name in this case is the partition key for this table, because only the partition key can be used as a `PutItem` condition expression.

If the tenant name was never recorded before, the record will be saved into the table successfully.

However, if the tenant name is already used by an existing record in the table, the operation will fail and initiate a DynamoDB `ConditionalCheckFailedException` exception. The exception will be used to return a failure message (`HTTP BadRequest`) indicating that the tenant name already exists.

A `DELETE` method API request will remove the record for a specific tenant name from the `Tenant Onboarding` table.

The DynamoDB record deletion in this example will succeed even if the record does not exist.

If the target record exists and is deleted, it will create a DynamoDB stream record. Otherwise, no downstream record will be created.

3. **Tenant on-boarding DynamoDB, with Amazon DynamoDB Streams enabled**

   This records the tenant metadata information, and any record save or deletion will send a stream downstream to the `Tenant Infrastructure` Lambda function.

4. **Tenant infrastructure Lambda function**

   This Lambda function is initiated by the DynamoDB stream record from the previous step. If the record is for an `INSERT` event, it invokes AWS CloudFormation to create a new tenant infrastructure with the CloudFormation template that is stored in an S3 bucket. If the record is for `REMOVE`, it initiates deletion of an existing stack based on the stream record's `Tenant Name` field.

5. **S3 bucket**

   This is for storing the CloudFormation template.

6. **IAM roles for each Lambda function and a service role for CloudFormation**

Each Lambda function has its unique IAM role with least-privilege permissions to achieve its task. For example, the `Tenant On-boarding` Lambda function has read/write access to DynamoDB, and the `Tenant Infrastructure` Lambda function can only read the DynamoDB stream.

A custom CloudFormation service role is created for tenant stack provisioning. This service role contains additional permissions for CloudFormation stack provisioning (for example, the AWS KMS key). This divides roles between Lambda and CloudFormation to avoid all permissions on a single role (Infrastructure Lambda role).

Permissions that allow powerful actions (such as creating and deleting CloudFormation stacks) are locked down and allowed only on resources that start with `tenantcluster-`. The exception is AWS KMS, because of its resource naming convention. The ingested tenant name from the API will be prepended with `tenantcluster-` along with other validation checks (alphanumeric with dash only, and limited to less than 30 characters to fit into most AWS resource naming). This ensures that the tenant name will not accidentally result in disruption of core infrastructure stacks or resources.

Tenant technology stack

A CloudFormation template is stored in the S3 bucket. The template provisions the tenant-specific AWS KMS key, a CloudWatch alarm, an SNS topic, an SQS queue, and an SQS policy.

The AWS KMS key is used for data encryption by Amazon SNS and Amazon SQS for their messages. The security practices for AwsSolutions-SNS2 and AwsSolutions-SQS2 recommend that you set up Amazon SNS and Amazon SQS with encryption. However, CloudWatch alarms don't work with Amazon SNS when using an AWS managed key, so you must use a customer managed key in this case. For more information, see the AWS Knowledge Center.

The SQS policy is used on the Amazon SQS queue to allow the created SNS topic to deliver the message to the queue. Without the SQS policy, the access will be denied. For more information, see the Amazon SNS documentation.

Decompose monoliths into microservices by using CQRS and event sourcing

Rodolfo Jr. Cerrada, Dmitry Gulin, and Tabby Ward, Amazon Web Services

Summary

This pattern combines two patterns, using both the command query responsibility separation (CQRS) pattern and the event sourcing pattern. The CQRS pattern separates responsibilities of the command and query models. The event sourcing pattern takes advantage of asynchronous event-driven communication to improve the overall user experience.

You can use CQRS and Amazon Web Services (AWS) services to maintain and scale each data model independently while refactoring your monolith application into microservices architecture. Then you can use the event sourcing pattern to synchronize data from the command database to the query database.

This pattern uses example code that includes a solution (*.sln) file that you can open using the latest version of Visual Studio. The example contains Reward API code to showcase how CQRS and event sourcing work in AWS serverless and traditional or on-premises applications.

To learn more about CQRS and event sourcing, see the Additional information section.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Amazon CloudWatch
- Amazon DynamoDB tables
- Amazon DynamoDB Streams
- AWS Identity and Access Management (IAM) access key and secret key; for more information, see the video in the *Related resources* section
- AWS Lambda
- Familiarity with Visual Studio
- Familiarity with AWS Toolkit for Visual Studio; for more information, see the *AWS Toolkit for Visual Studio demo* video in the *Related resources* section

Product versions

- Visual Studio 2019 Community Edition.
- AWS Toolkit for Visual Studio 2019.
- .NET Core 3.1. This component is an option in the Visual Studio installation. To include .NET Core during installation, select **NET Core cross-platform development**.

Limitations

- The example code for a traditional on-premises application (ASP.NET Core Web API and data access objects) does not come with a database. However, it comes with the `CustomerData` in-memory object, which acts as a mock database. The code provided is enough for you to test the pattern.

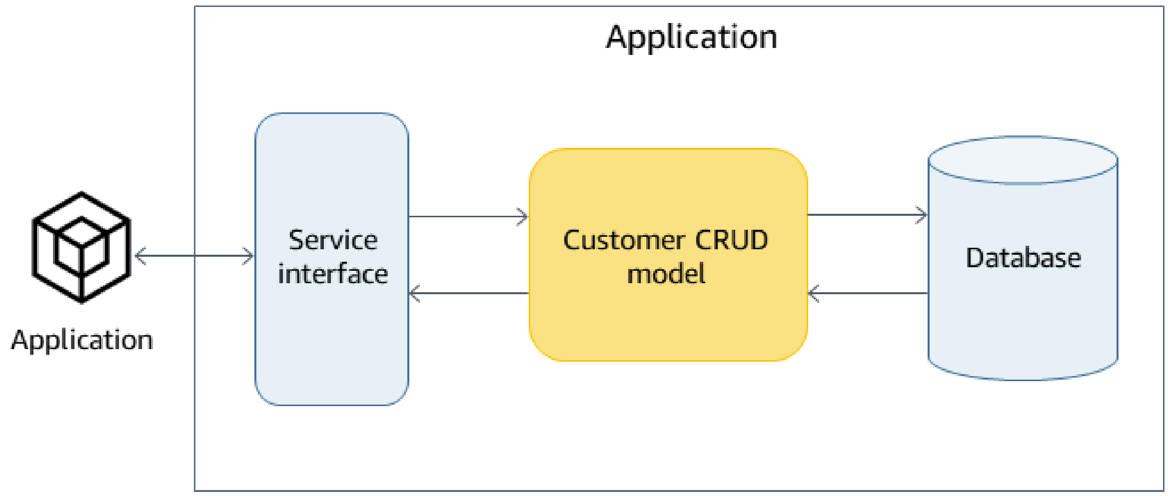
Architecture

Source technology stack

- ASP.NET Core Web API project
- IIS Web Server
- Data access object
- CRUD model

Source architecture

In the source architecture, the CRUD model contains both command and query interfaces in one application. For example code, see `CustomerDAO.cs` (attached).

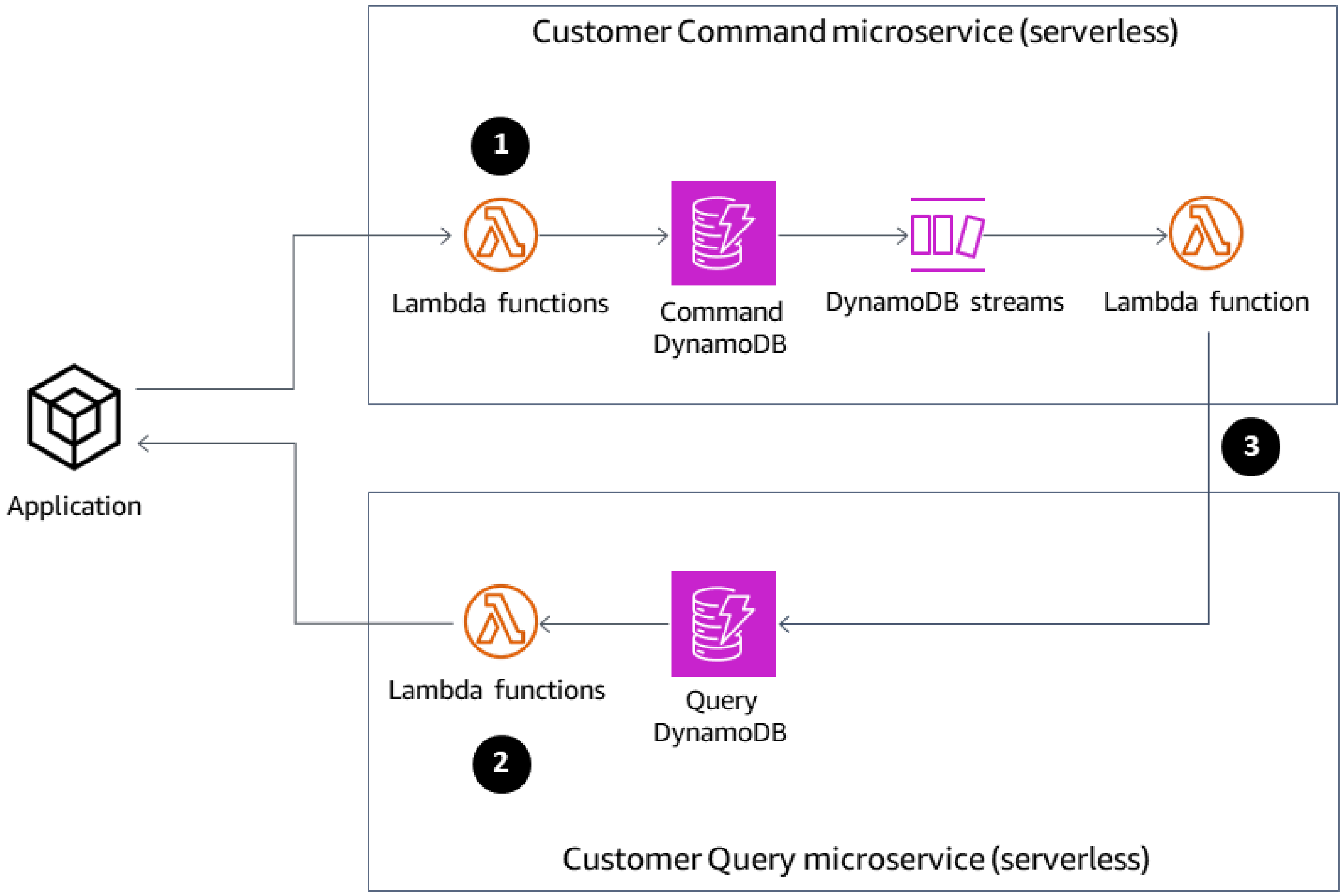


Target technology stack

- Amazon DynamoDB
- Amazon DynamoDB Streams
- AWS Lambda
- (Optional) Amazon API Gateway
- (Optional) Amazon Simple Notification Service (Amazon SNS)

Target architecture

In the target architecture, the command and query interfaces are separated. The architecture shown in the following diagram can be extended with API Gateway and Amazon SNS. For more information, see the Additional information section.



1. Command Lambda functions perform write operations, such as create, update, or delete, on the database.
2. Query Lambda functions perform read operations, such as get or select, on the database.
3. This Lambda function processes the DynamoDB streams from the Command database and updates the Query database for the changes.

Tools

Tools

- Amazon DynamoDB – Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
- Amazon DynamoDB Streams – DynamoDB Streams captures a time-ordered sequence of item-level modifications in any DynamoDB table. It then stores this information in a log for up to 24 hours. Encryption at rest encrypts the data in DynamoDB streams.

- [AWS Lambda](#) – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- [AWS Management Console](#) – The AWS Management Console is a web application that comprises a broad collection of service consoles for managing AWS services.
- [Visual Studio 2019 Community Edition](#) – Visual Studio 2019 is an integrated development environment (IDE). The Community Edition is free for open-source contributors. In this pattern, you will use Visual Studio 2019 Community Edition to open, compile, and run example code. For viewing only, you can use any text editor or [Visual Studio Code](#).
- [AWS Toolkit for Visual Studio](#) – The AWS Toolkit for Visual Studio is a plugin for the Visual Studio IDE. The AWS Toolkit for Visual Studio makes it easier for you to develop, debug, and deploy .NET applications that use AWS services.

Code

The example code is attached. For instructions on deploying the example code, see the *Epics* section.

Epics

Open and build the solution

Task	Description	Skills required
Open the solution.	1. Download the example source code (`CQRS-ES Code.zip`) from the *Attachments* section, and extract the files. 2. In the Visual Studio IDE, choose **File**, **Open**, **Project Solution**, and navigate to the folder where you extracted the source code. 3. Choose **AWS.APG.CQRSES.sln**, and then	App developer

Task	Description	Skills required
	choose **Open**. The entire solution is loaded into Visual Studio.	
Build the solution.	Open the context (right-click) menu for the solution, and then choose **Build Solution**. This will build and compile all the projects in the solution. It should compile successfully. Visual Studio Solution Explorer should show the directory structure. • `CQRS On-Premises Code Sample` contains an example of using CQRS on premises. • `CQRS AWS Serverless` contains all the CQRS and event-sourcing example code using AWS serverless services.	App developer

Build the DynamoDB tables

Task	Description	Skills required
Provide credentials.	If you don't have an access key yet, see the video in the *Related resources* section. 1. In Solution Explorer, expand **CQRS AWS**	App developer, Data engineer, DBA

Task	Description	Skills required
	Serverless, and then expand the **Build** solution folder. 2. Expand the **AwS.APG.CQRSES.Build** project and view the `Program.cs` file. 3. Scroll to the top of `Program.cs` and look for `Program()` . 4. Replace `YOUR ACCESS KEY` with your account access key, and replace `YOUR SECRET KEY`with your account secret key. Note that in a production environment, you would not hardcode your keys. Instead, you could use [AWS Secrets Manager](https://aws.amazon.com/secrets-manager/) to store and retrieve the credentials.	
Build the project.	To build the project, open the context (right-click) menu for the **AwS.APG.CQRSES.Build** project, and then choose **Build**.	App developer, Data engineer, DBA

Task	Description	Skills required
Build and populate the tables.	To build the tables and populate them with seed data, open the context (right-click) menu for the **AwS.APG.CQRSES.Build** project, and then choose **Debug**, **Start New Instance**.	App developer, Data engineer, DBA
Verify the table construction and the data.	To verify, navigate to **AWS Explorer**, and expand **Amazon DynamoDB**. It should display the tables. Open each table to display the example data.	App developer, Data engineer, DBA

Run local tests

Task	Description	Skills required
Build the CQRS project.	1. Open the solution, and navigate to the **CQRS AWS Services/CQRS/Tests** solution folder. 2. In the **AWS.APG.CQRSES.CQRSLambda.Tests** project, open **BaseFunctionTest.cs**, and replace *AccessKey* and *SecretKey* with the IAM keys that you created. 3. Save the changes. 4. To compile and build the test project, open the context (right-click) menu	App developer, Test engineer

Task	Description	Skills required
	for the project, and then choose **Build**.	
Build the event-sourcing project.	1. Navigate to the **CQRS AWS Services/Event Source/Tests** solution folder. 2. In the **AWS.APG.CQRSES.EventSourceLambda.Tests** project, open **BaseFunctionTest.cs**, and replace *AccessKey* and *SecretKey* with the IAM keys that you created. 3. Save the changes. 4. To compile and build the test project, open the context (right-click) menu for the project, and then choose **Build**.	App developer, Test engineer
Run the tests.	To run all tests, choose **View**, **Test Explorer**, and then choose **Run All Tests In View**. All tests should pass, which is indicated by a green check mark icon.	App developer, Test engineer

Publish the CQRS Lambda functions to AWS

Task	Description	Skills required
Publish the first Lambda function.	1. In Solution Explorer, open the context (right-click) menu for the AWS.APG.C	App developer, DevOps engineer

Task	Description	Skills required
	QRSES.CommandCreateLambda project, and then choose **Publish to AWS Lambda**. 2. Select the profile that you want to use and the AWS Region where you want to deploy the Lambda function, and the function name. 3. For the remaining fields, keep the default values, and choose **Next**. 4. In **Role Name** dropdown list, select **AWSLambdaFullAccess**. 5. To provide your account keys, choose **Add**, and enter `AcessKey` as the variable and your access key as the value. Then choose **Add** again, enter `SecretKey` as the variable and your secret key as the value. 6. For the remaining fields, keep the default values, and choose **Upload**. After the Lambda test function uploads, it appears in Visual Studio automatically. 7. Repeat steps 1-6 for the following projects:	

Task	Description	Skills required
	• AWS.APG.CQRSES.CommandDeleteLambda • AWS.APG.CQRSES.CommandUpdateLambda • AWS.APG.CQRSES.CommandAddRewardLambda • AWS.APG.CQRSES.CommandRedeemRewardLambda • AWS.APG.CQRSES.QueryCustomerListLambda • AWS.APG.CQRSES.QueryRewqardLambda	
Verify the function upload.	(Optional) You can verify that the function was successfully loaded by navigating to AWS Explorer and expanding **AWS Lambda**. To open the test window, choose the Lambda function (double-click).	App developer, DevOps engineer

Task	Description	Skills required
Test the Lambda function.	1. Enter the request data, or copy an example request data from **Test data** in the [Additional information](#) section. Make sure that you select data that is for the function you are testing. 2. To run the test, choose **Invoke**. The response and any errors are displayed in the **Response** text box, and logs are shown in the **Logs** text box or in CloudWatch Logs. 3. To verify the data, in AWS Explorer, choose the DynamoDB table (double-click). All CQRS Lambda projects are found under the `CQRS AWS Serverless\CQRS\Command Microservice` and `CQRS AWS Serverless\CQRS\Command Microservice` solution folders. For the solution directory and projects, see **Source code directory** in the [Additional information](#) section.	App developer, DevOps engineer

Task	Description	Skills required
Publish the remaining functions.	Repeat the previous steps for the following projects: • AWS.APG.CQRSES.CommandDeleteLambda • AWS.APG.CQRSES.CommandUpdateLambda • AWS.APG.CQRSES.CommandAddRewardLambda • AWS.APG.CQRSES.CommandRedeemRewardLambda • AWS.APG.CQRSES.QueryCustomerListLambda • AWS.APG.CQRSES.QueryRewqardLambda	App developer, DevOps engineer

Set up the Lambda function as an event listener

Task	Description	Skills required
Publish the Customer and Reward Lambda event handlers.	To publish each event handler, follow the steps in the preceding epic. The projects are under the `CQRS AWS Serverless\Event Source\Customer Event` and `CQRS AWS Serverless\Event Source\Reward Event` solution folders. For more information, see *Source code*	App developer

Task	Description	Skills required
	directory in the Additional information section.	

Task	Description	Skills required
Attach the event-sourcing Lambda event listener.	1. Log in to the AWS Management Console using the same account you use when you publish the Lambda projects. 2. For the **Region**, select **US East 1** or the Region where you deployed the Lambda functions in the previous epic. 3. Navigate to the **Lambda** service. 4. Select the `EventSourceCustomer` Lambda function. 5. Choose **Add Trigger**. 6. In the **Trigger configuration** dropdown list, select **DynamoDB**. 7. In the **DynamoDB table** dropdown list, select **cqrses-customer-cmd**. 8. In the **Starting position** dropdown list, select *Trim horizon* from . Trim horizon means that the DynamoDB trigger will start reading at the last (untrimmed) stream record, which is the oldest record in the shard. 9. Select the **Enable trigger** check box.	App developer

Task	Description	Skills required
	10For the remaining fields, keep the default values, and choose **Add**. After the listener is successfully attached to the DynamoDB table, it will be displayed on the Lambda designer page.	
Publish and attach the EventSourceReward Lambda function.	To publish and attach the `EventSourceReward` Lambda function, repeat the steps in the previous two stories, selecting **cqrses-reward-cmd** from the **DynamoDB table** dropdown list.	App developer

Test and validate the DynamoDB streams and Lambda trigger

Task	Description	Skills required
Test the stream and the Lambda trigger.	1. In Visual Studio, navigate to AWS Explorer. 2. Expand AWS Lambda, and choose the **CommandRedeemReward** function (double-click). In the function window that opens, you can test the function. 3. In the **Request** text box, enter the request data in JavaScript Object Notation	App developer

Task	Description	Skills required
	(JSON) format. For an example request, see **Test data** in the [Additional information](#) section. 4. Choose **Invoke**.	
Validate, using the DynamodDB reward query table.	1. Open the **cqrses-reward-query** table. 2. Check the points of the customer that redeemed the reward. The redeemed points should be subtracted from the customer's total aggregated points.	App developer
Validate, using CloudWatch Logs.	1. Navigate to CloudWatch and choose **Log groups**. 2. The **/aws/lambda/EventSourceReward** log group contains the logs for the `EventSourceReward` trigger. All Lambda calls are logged, including the messages you placed in `context.Logger.LogLine` and `Console.Writeline` in the Lambda code.	App developer

Task	Description	Skills required
Validate the EventSourceCustomer trigger.	To validate the `EventSourceCustomer` trigger, repeat the steps in this epic, using the `EventSourceCustomer` trigger's respective customer table and CloudWatch logs.	App developer

Related resources

References

- Visual Studio 2019 Community Edition downloads
- AWS Toolkit for Visual Studio download
- AWS Toolkit for Visual Studio User Guide
- Serverless on AWS
- DynamoDB Use Cases and Design Patterns
- Martin Fowler CQRS
- Martin Fowler Event Sourcing

Videos

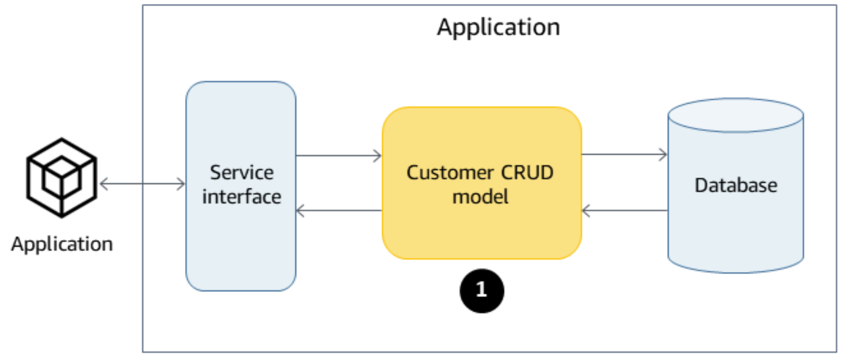
- AWS Toolkit for Visual Studio demo
- How do I create an access key ID for a new IAM user?

Additional information

CQRS and event sourcing

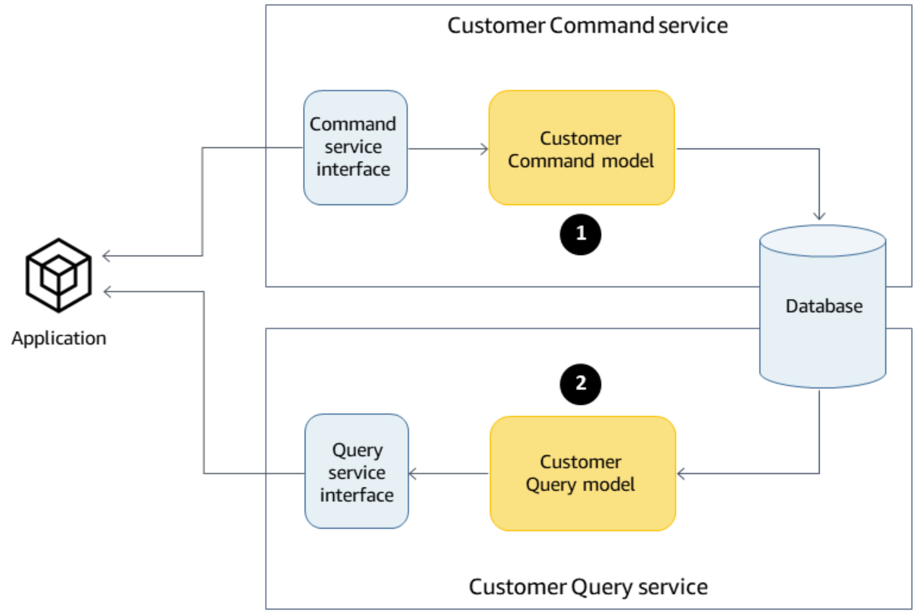
CQRS

The CQRS pattern separates a single conceptual operations model, such as a data access object single CRUD (create, read, update, delete) model, into command and query operations models. The command model refers to any operation, such as create, update, or delete, that changes the state. The query model refers to any operation that returns a value.



1. The Customer CRUD model includes the following interfaces:
   - `Create Customer()`
   - `UpdateCustomer()`
   - `DeleteCustomer()`
   - `AddPoints()`
   - `RedeemPoints()`
   - `GetVIPCustomers()`
   - `GetCustomerList()`
   - `GetCustomerPoints()`

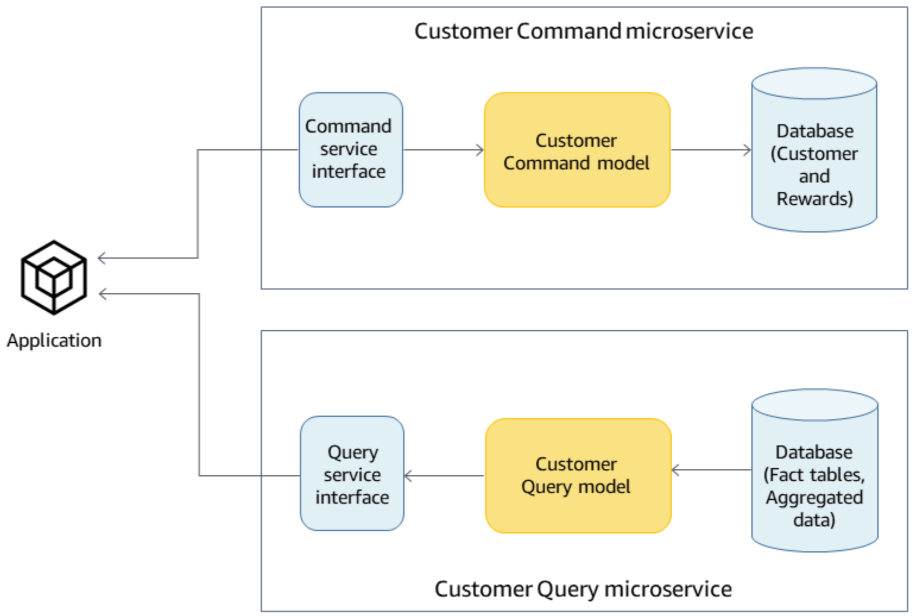
As your requirements become more complex, you can move from this single-model approach. CQRS uses a command model and a query model to separate the responsibility for writing and reading data. That way, the data can be independently maintained and managed. With a clear separation of responsibilities, enhancements to each model do not impact the other. This separation improves maintenance and performance, and it reduces the complexity of the application as it grows.



1. Interfaces in the Customer Command model:
   - `Create Customer()`
   - `UpdateCustomer()`
   - `DeleteCustomer()`
   - `AddPoints()`
   - `RedeemPoints()`
2. Interfaces in the Customer Query model:
   - `GetVIPCustomers()`
   - `GetCustomerList()`
   - `GetCustomerPoints()`
   - `GetMonthlyStatement()`

For example code, see *Source code directory*.

The CQRS pattern then decouples the database. This decoupling leads to the total independence of each service, which is the main ingredient of microservice architecture.



Using CQRS in the AWS Cloud, you can further optimize each service. For example, you can set different compute settings or choose between a serverless or a container-based microservice. You can replace your on-premises caching with Amazon ElastiCache. If you have an on-premises publish/subscribe messaging, you can replace it with Amazon Simple Notification Service (Amazon SNS). Additionally, you can take advantage of pay-as-you-go pricing and the wide array of AWS services that you pay only for what you use.

CQRS includes the following benefits:

- Independent scaling – Each model can have its scaling strategy adjusted to meet the requirements and demand of the service. Similar to high-performance applications, separating read and write enables the model to scale independently to address each demand. You can also add or reduce compute resources to address the scalability demand of one model without affecting the other.
- Independent maintenance – Separation of query and command models improves the maintainability of the models. You can make code changes and enhancements to one model without affecting the other.
- Security – It's easier to apply the permissions and policies to separate models for read and write.
- Optimized reads – You can define a schema that is optimized for queries. For example, you can define a schema for the aggregated data and a separate schema for the fact tables.
- Integration – CQRS fits well with event-based programming models.

- Managed complexity – The separation into query and command models is suited to complex domains.

When using CQRS, keep in mind the following caveats:

- The CQRS pattern applies only to a specific portion of an application and not the whole application. If implemented on a domain that does not fit the pattern, it can reduce productivity, increase risk, and introduce complexity.
- The pattern works best for frequently used models that have an imbalance read and write operations.
- For read-heavy applications, such as large reports that take time to process, CQRS gives you the option to select the right database and create a schema to store your aggregated data. This improves the response time of reading and viewing the report by processing the report data only one time and dumping it in the aggregated table.
- For the write-heavy applications, you can configure the database for write operations and allow the command microservice to scale independently when the demand for write increases. For examples, see the `AWS.APG.CQRSES.CommandRedeemRewardLambda` and `AWS.APG.CQRSES.CommandAddRewardLambda` microservices.

Event sourcing

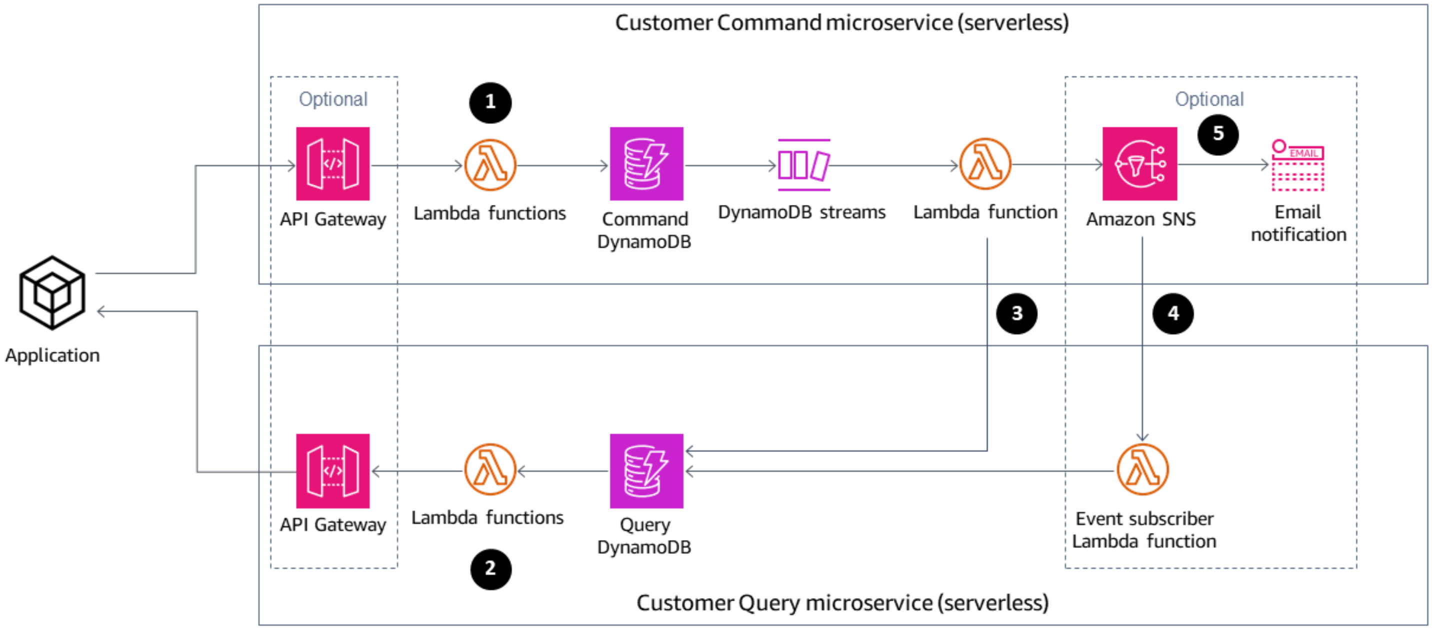
The next step is to use event sourcing to synchronize the query database when a command is run. For example, consider the following events:

- A customer reward point is added that requires the customer total or aggregated reward points in the query database to be updated.
- A customer's last name is updated in the command database, which requires the surrogate customer information in the query database to be updated.

In the traditional CRUD model, you ensure consistency of data by locking the data until it finishes a transaction. In event sourcing, the data are synchronized through publishing a series of events that will be consumed by a subscriber to update its respective data.

The event-sourcing pattern ensures and records a full series of actions taken on the data and publishes it through a sequence of events. These events represent a set of changes to the data that subscribers of that event must process to keep their record updated. These events are consumed by the subscriber, synchronizing the data on the subscriber's database. In this case, that's the query database.

The following diagram shows event sourcing used with CQRS on AWS.



1. Command Lambda functions perform write operations, such as create, update, or delete, on the database.
2. Query Lambda functions perform read operations, such as get or select, on the database.
3. This Lambda function processes the DynamoDB streams from the Command database and updates the Query database for the changes. You can also use this function also to publish a message to Amazon SNS so that its subscribers can process the data.
4. (Optional) The Lambda event subscriber processes the message published by Amazon SNS and updates the Query database.
5. (Optional) Amazon SNS sends email notification of the write operation.

On AWS, the query database can be synchronized by DynamoDB Streams. DynamoDB captures a time-ordered sequence of item-level modifications in a DynamobDB table in near-real time and durably stores the information within 24 hours.

Activating DynamoDB Streams enables the database to publish a sequence of events that makes the event sourcing pattern possible. The event sourcing pattern adds the event subscriber. The event subscriber application consumes the event and processes it depending on the subscriber's responsibility. In the previous diagram, the event subscriber pushes the changes to the Query

DynamoDB database to keep the data synchronized. The use of Amazon SNS, the message broker, and the event subscriber application keeps the architecture decoupled.

Event sourcing includes the following benefits:

- Consistency for transactional data
- A reliable audit trail and history of the actions, which can be used to monitor actions taken in the data
- Allows distributed applications such as microservices to synchronize their data across the environment
- Reliable publication of events whenever the state changes
- Reconstructing or replaying of past states
- Loosely coupled entities that exchange events for migration from a monolithic application to microservices
- Reduction of conflicts caused by concurrent updates; event sourcing avoids the requirement to update objects directly in the data store
- Flexibility and extensibility from decoupling the task and the event
- External system updates
- Management of multiple tasks in a single event

When using event sourcing, keep in mind the following caveats:

- Because there is some delay in updating data between the source subscriber databases, the only way to undo a change is to add a compensating event to the event store.
- Implementing event sourcing has a learning curve since its different style of programming.

Test data

Use the following test data to test the Lambda function after successful deployment.

CommandCreate Customer

```
{  "Id":1501,  "Firstname":"John",  "Lastname":"Done",  "CompanyName":"AnyCompany",
 "Address": "USA",  "VIP":true }
```

CommandUpdate Customer

```
{  "Id":1501,  "Firstname":"John",  "Lastname":"Doe",  "CompanyName":"Example Corp.",
 "Address": "Seattle, USA",  "VIP":true }
```

CommandDelete Customer

Enter the customer ID as request data. For example, if the customer ID is 151, enter 151 as request data.

```
151
```

QueryCustomerList

This is blank. When it is invoked, it will return all customers.

CommandAddReward

This will add 40 points to customer with ID 1 (Richard).

```
{
  "Id":10101,
  "CustomerId":1,
  "Points":40
}
```

CommandRedeemReward

This will deduct 15 points to customer with ID 1 (Richard).

```
{
  "Id":10110,
  "CustomerId":1,
  "Points":15
}
```

QueryReward

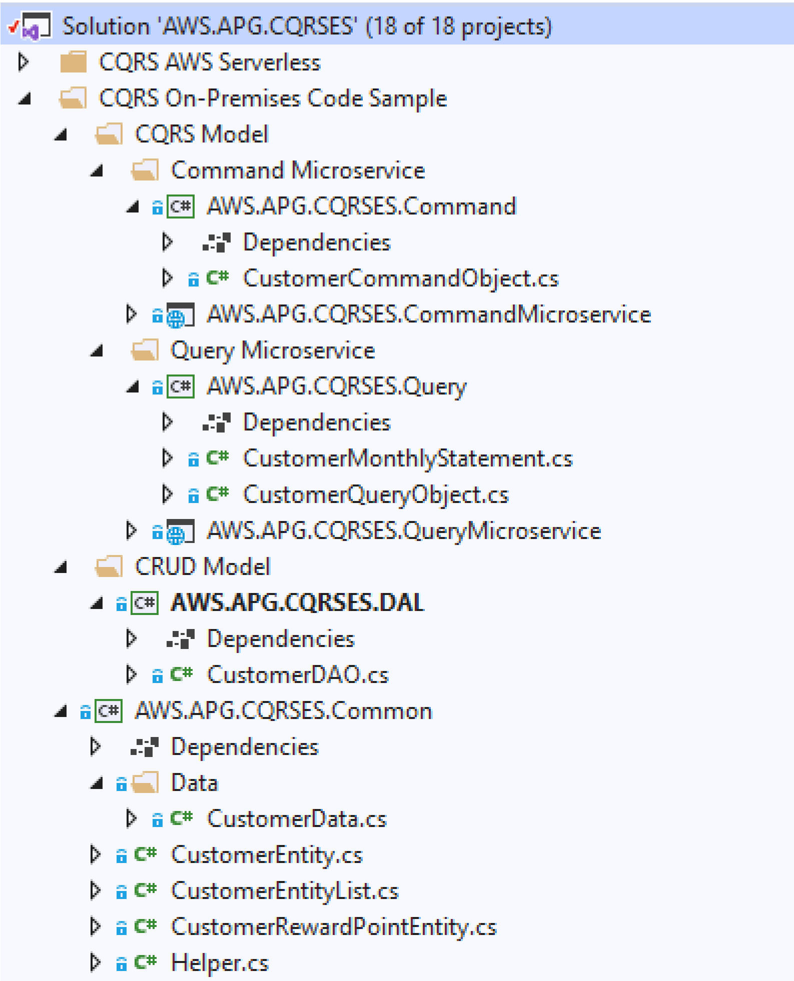
Enter the ID of the customer. For example, enter 1 for Richard, 2 for Arnav, and 3 for Shirley.

```
2
```

Source code directory

Use the following table as a guide to the directory structure of the Visual Studio solution.

CQRS On-Premises Code Sample solution directory



Customer CRUD model

CQRS On-Premises Code Sample\CRUD Model\AWS.APG.CQRSES.DAL project

CQRS version of the Customer CRUD model

- Customer command: `CQRS On-Premises Code Sample\CQRS Model\Command Microservice\AWS.APG.CQRSES.Command`project
- Customer query: `CQRS On-Premises Code Sample\CQRS Model\Query Microservice \AWS.APG.CQRSES.Query` project

Command and Query microservices

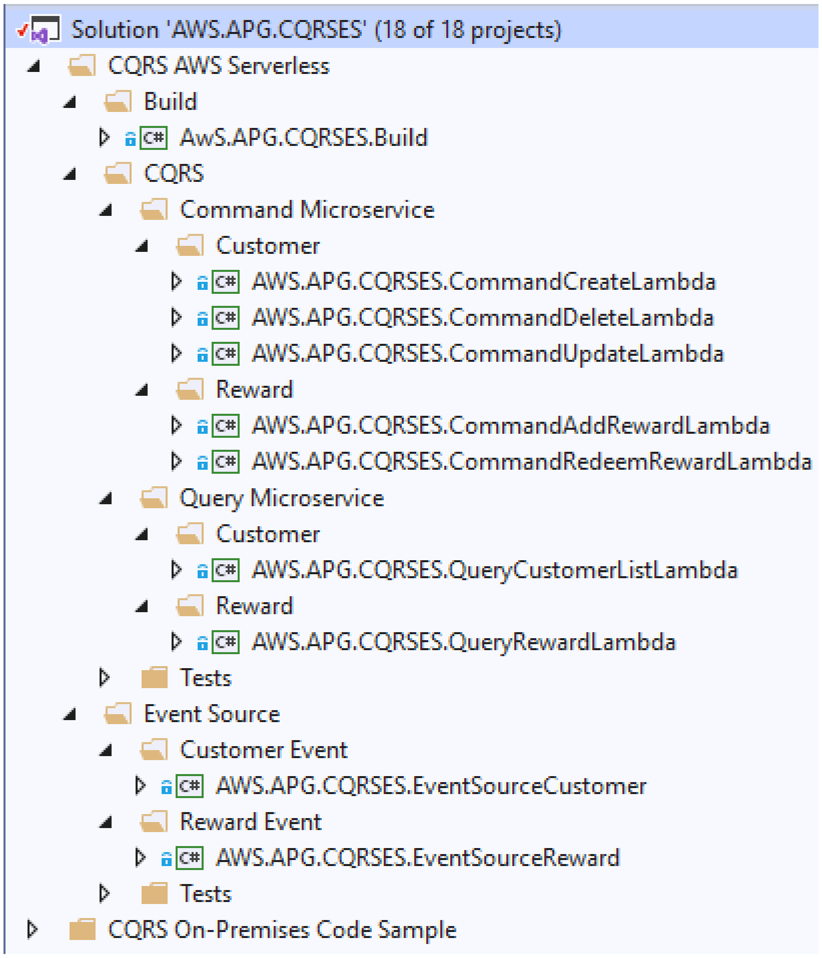
The Command microservice is under the solution folder `CQRS On-Premises Code Sample\CQRS Model\Command Microservice`:

- `AWS.APG.CQRSES.CommandMicroservice` ASP.NET Core API project acts as the entry point where consumers interact with the service.
- `AWS.APG.CQRSES.Command` .NET Core project is an object that hosts command-related objects and interfaces.

The query microservice is under the solution folder `CQRS On-Premises Code Sample\CQRS Model\Query Microservice`:

- `AWS.APG.CQRSES.QueryMicroservice` ASP.NET Core API project acts as the entry point where consumers interact with the service.
- `AWS.APG.CQRSES.Query` .NET Core project is an object that hosts query-related objects and interfaces.

CQRS AWS Serverless code solution directory



This code is the AWS version of the on-premises code using AWS serverless services.

In C# .NET Core, each Lambda function is represented by one .NET Core project. In this pattern's example code, there is a separate project for each interface in the command and query models.

CQRS using AWS services

You can find the root solution directory for CQRS using AWS serverless services is in the `CQRS AWS Serverless\CQRS`folder. The example includes two models: Customer and Reward.

The command Lambda functions for Customer and Reward are under `CQRS\Command Microservice\Customer` and `CQRS\Command Microservice\Reward` folders. They contain the following Lambda projects:

- Customer command: `CommandCreateLambda`, `CommandDeleteLambda`, and `CommandUpdateLambda`
- Reward command: `CommandAddRewardLambda` and `CommandRedeemRewardLambda`

The query Lambda functions for Customer and Reward are found under the `CQRS\Query Microservice\Customer` and `CQRS\QueryMicroservice\Reward`folders. They contain the `QueryCustomerListLambda` and `QueryRewardLambda` Lambda projects.

CQRS test project

The test project is under the `CQRS\Tests` folder. This project contains a test script to automate testing the CQRS Lambda functions.

Event sourcing using AWS services

The following Lambda event handlers are initiated by the Customer and Reward DynamoDB streams to process and synchronize the data in query tables.

- The `EventSourceCustomer` Lambda function is mapped to the Customer table (`cqrses-customer-cmd`) DynamoDB stream.
- The `EventSourceReward` Lambda function is mapped to the Reward table (`cqrses-reward-cmd`) DynamoDB stream.

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

More patterns

- Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer
- Automate adding or updating Windows registry entries using AWS Systems Manager
- Automate cross-Region failover and failback by using DR Orchestrator Framework
- Automatically build and deploy a Java application to Amazon EKS using a CI/CD pipeline
- Automatically build CI/CD pipelines and Amazon ECS clusters for microservices using AWS CDK
- Back up and archive mainframe data to Amazon S3 using BMC AMI Cloud Data
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager
- Build an enterprise data mesh with Amazon DataZone, AWS CDK, and AWS CloudFormation
- Chain AWS services together using a serverless approach
- Containerize mainframe workloads that have been modernized by Blu Age
- Continuously deploy a modern AWS Amplify web application from an AWS CodeCommit repository
- Convert and unpack EBCDIC data to ASCII on AWS by using Python
- Convert mainframe data files with complex record layouts using Micro Focus
- Create a pipeline and AMI using CodePipeline and HashiCorp Packer
- Deploy artifact updates to on-premises Amazon EC2 instances using AWS CodePipeline
- Create a portal for micro-frontends by using AWS Amplify, Angular, and Module Federation
- Deploy and debug Amazon EKS clusters
- Deploy containers by using Elastic Beanstalk
- Emulate Oracle DR by using a PostgreSQL-compatible Aurora global database
- Generate data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Generate Db2 z/OS data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Identify duplicate container images automatically when migrating to an Amazon ECR repository
- Implement AI-powered Kubernetes diagnostics and troubleshooting with K8sGPT and Amazon Bedrock integration
- Implement Microsoft Entra ID-based authentication in an AWS Blu Age modernized mainframe application
- Implement path-based API versioning by using custom domains in Amazon API Gateway

- Incrementally migrate from Amazon RDS for Oracle to Amazon RDS for PostgreSQL using Oracle SQL Developer and AWS SCT
- Integrate Stonebranch Universal Controller with AWS Mainframe Modernization
- Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions
- Migrate an AWS member account from AWS Organizations to AWS Control Tower
- Migrate and replicate VSAM files to Amazon RDS or Amazon MSK using Connect from Precisely
- Migrate from SAP ASE to Amazon RDS for SQL Server using AWS DMS
- Migrate Oracle external tables to Amazon Aurora PostgreSQL-Compatible
- Modernize the CardDemo mainframe application by using AWS Transform
- Modernize mainframe batch printing workloads on AWS by using Rocket Enterprise Server and LRS VPSX/MFI
- Modernize mainframe online printing workloads on AWS by using Micro Focus Enterprise Server and LRS VPSX/MFI
- Modernize mainframe output management on AWS by using Rocket Enterprise Server and LRS PageCenterX
- Move mainframe files directly to Amazon S3 using Transfer Family
- Optimize multi-account serverless deployments by using the AWS CDK and GitHub Actions workflows
- Automate blue/green deployments of Amazon Aurora global databases by using IaC principles
- Replicate mainframe databases to AWS by using Precisely Connect
- Run Amazon ECS tasks on Amazon WorkSpaces with Amazon ECS Anywhere
- Send telemetry data from AWS Lambda to OpenSearch for real-time analytics and visualization
- Set up a Helm v3 chart repository in Amazon S3
- Set up AWS CloudFormation drift detection in a multi-Region, multi-account organization
- Structure a Python project in hexagonal architecture using AWS Lambda
- Test AWS infrastructure by using LocalStack and Terraform Tests
- Upgrade SAP Pacemaker clusters from ENSA1 to ENSA2
- Use Amazon Q Developer as a coding assistant to increase your productivity
- Validate Account Factory for Terraform (AFT) code locally

Mainframes

Topics

- Access AWS services from IBM z/OS by installing the AWS CLI
- Back up and archive mainframe data to Amazon S3 using BMC AMI Cloud Data
- Build COBOL Db2 programs by using AWS Mainframe Modernization and AWS CodeBuild
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager
- Build an advanced mainframe file viewer in the AWS Cloud
- Containerize mainframe workloads that have been modernized by Blu Age
- Convert and unpack EBCDIC data to ASCII on AWS by using Python
- Convert mainframe files from EBCDIC format to character-delimited ASCII format in Amazon S3 using AWS Lambda
- Convert mainframe data files with complex record layouts using Micro Focus
- Deploy an environment for containerized Blu Age applications by using Terraform
- Generate Db2 z/OS data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Generate data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight
- Implement Microsoft Entra ID-based authentication in an AWS Blu Age modernized mainframe application
- Integrate Stonebranch Universal Controller with AWS Mainframe Modernization
- Migrate and replicate VSAM files to Amazon RDS or Amazon MSK using Connect from Precisely
- Modernize the CardDemo mainframe application by using AWS Transform
- Modernize mainframe output management on AWS by using Rocket Enterprise Server and LRS PageCenterX
- Modernize mainframe batch printing workloads on AWS by using Rocket Enterprise Server and LRS VPSX/MFI
- Mainframe modernization: DevOps on AWS with Rocket Software Enterprise Suite
- Modernize mainframe online printing workloads on AWS by using Micro Focus Enterprise Server and LRS VPSX/MFI
- Move mainframe files directly to Amazon S3 using Transfer Family
- Secure and streamline user access in a Db2 federation database on AWS by using trusted contexts

- [Transfer large-scale Db2 z/OS data to Amazon S3 in CSV files]
- [More patterns]

Access AWS services from IBM z/OS by installing the AWS CLI

Souma Ghosh, Paulo Vitor Pereira, and Phil de Valence, Amazon Web Services

Summary

The AWS Command Line Interface (AWS CLI) is an open source tool for managing multiple AWS services by using commands in a command line shell. With minimal configuration, you can run commands from command line sessions such as the command prompt, terminal, and bash shell to implement functionality that's equivalent to that provided by the browser-based AWS Management Console.

All AWS infrastructure as a service (IaaS) administration, management, and access functions in the AWS Management Console are available in the AWS API and AWS CLI. You can install the AWS CLI on an IBM z/OS mainframe to directly access, manage, and interact with AWS services from z/OS. The AWS CLI enables users and applications to perform various tasks, such as:

- Transferring files or datasets between z/OS and Amazon Simple Storage Service (Amazon S3) object storage and viewing content of buckets
- Starting and stopping different AWS resources; for example, starting a batch job in an AWS Mainframe Modernization environment
- Calling an AWS Lambda function to implement common business logic
- Integrating with artificial intelligence and machine learning (AI/ML) and analytics services

This pattern describes how to install, configure, and use the AWS CLI on z/OS. You can install it globally, so it's available to all z/OS users, or at a user level. The pattern also details how to use the AWS CLI in an interactive command line session from z/OS Unix System Services (USS) or as a batch job.

Prerequisites and limitations

Prerequisites

- **Network communication from z/OS to AWS**

  By default, the AWS CLI sends requests to AWS services by using HTTPS on TCP port 443. To use the AWS CLI successfully, you must be able to make outbound connections on TCP port 443. You can use any of the following z/OS USS commands (some of these might not be installed in your environment) to test network connectivity from z/OS to AWS:

```
ping amazonaws.com
dig amazonaws.com
traceroute amazonaws.com
curl -k https://docs.aws.amazon.com/cli/v1/userguide/cli-chap-welcome.html
```

- **AWS credentials**

  In order to communicate with AWS Cloud services from z/OS, the AWS CLI requires you to configure some credentials with privileges to access the target AWS account. For programmatic commands to AWS, you can use access keys, which consist of an access key ID and secret access key. If you don't have access keys, you can create them from the AWS Management Console. As a best practice, do not use the access keys for the AWS account root user for any task unless the root user is required. Instead, create a new administrator IAM user and prepare for least-privilege permissions to set up the user with access keys. After you create the user, you can create an access key ID and secret access key for this user.

  > **Warning**
  >
  > AWS Identity and Access Management (IAM) users have long-term credentials that present a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed.

- **IBM Python for z/OS**

  The AWS CLI requires Python 3.8 or later. IBM has enabled Python to run on z/OS with IBM Open Enterprise Python for z/OS. IBM Open Enterprise Python is available at no charge through Shopz SMP/E, or you can download the PAX file from the IBM website. For instructions, see the installation and configuration documentation for IBM Open Enterprise Python for z/OS.

Limitations

- The installation instructions provided in this pattern are applicable to **AWS CLI version 1 only**. The latest version of the AWS CLI is version 2. However, this pattern uses the older version because the installation methods are different for version 2, and the binary executables available for version 2 aren't compatible with the z/OS system.

Product versions

- AWS CLI version 1
- Python 3.8 or later

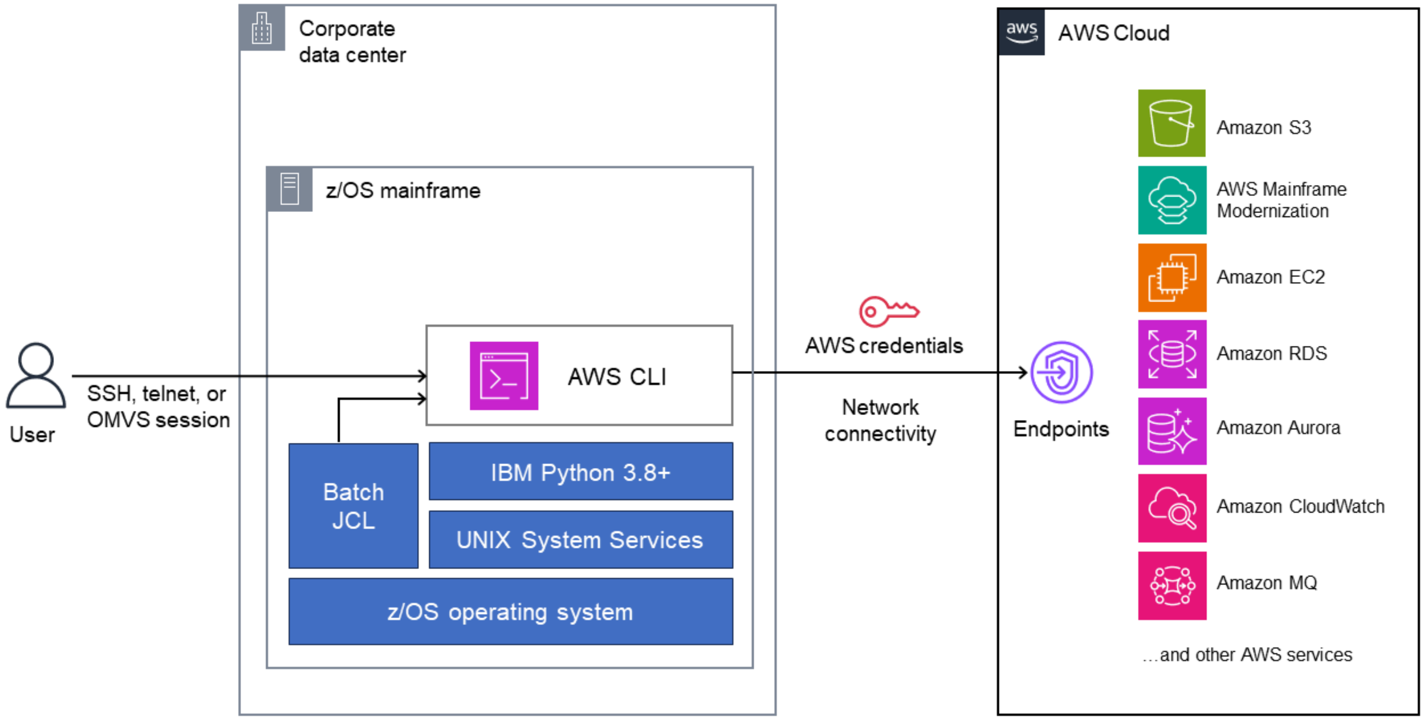
Architecture

Technology stack

- Mainframe running z/OS
- Mainframe z/OS UNIX System Services (USS)
- Mainframe Open MVS (OMVS) – z/OS UNIX shell environment command interface
- Mainframe disk, such as a direct-access storage device (DASD)
- AWS CLI

Target architecture

The following diagram shows an AWS CLI deployment on IBM z/OS. You can invoke the AWS CLI from an interactive user session, such as SSH, and telnet sessions. You can also invoke it from a batch job by using job control language (JCL), or from any program that can call a z/OS Unix shell command.



The AWS CLI communicates with AWS service endpoints over a TCP/IP network. This network connection can happen over the internet or through a private AWS Direct Connect connection from the customer data center to AWS Cloud data centers. The communication is authenticated with AWS credentials and encrypted.

Automation and scale

You can explore the capabilities of an AWS service with the AWS CLI and develop USS shell scripts to manage your AWS resources from z/OS. You can also run AWS CLI commands and shell scripts from the z/OS batch environment, and you can automate batch jobs to run on a specific schedule by integrating with mainframe schedulers. AWS CLI commands or scripts can be coded inside parameters (PARMs) and procedures (PROCs), and can be scaled by following the standard approach of calling the PARM or PROC from different batch jobs with different parameters.

Tools

- [AWS Command Line Interface (AWS CLI)](#) is an open source tool that helps you interact with AWS services through commands in your command-line shell.

Best practices

- For security reasons, restrict the access permissions to the USS directory where the AWS access key details are stored. Allow access to only the users or programs that use the AWS CLI.
- Do not use the AWS account root user access keys for any task. Instead, create a new administrator IAM user for yourself and set it up with access keys.

> **Warning**
>
> IAM users have long-term credentials that present a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed.

Epics

Install AWS CLI version 1 on z/OS USS

Task	Description	Skills required
Install Python 3.8 or later.	1. Log in to the z/OS USS command prompt interface by using one of these methods: • Use a Time Sharing Option (TSO) OMVS command from the Interactive System Productivity Facility (ISPF) panel, or • Use SSH or telnet to connect to the IP of the mainframe logical partition (LPAR).	Mainframe z/OS administr ator

Task	Description	Skills required
	This pattern assumes that `cliuser` is the userid used to log in to the USS environment and `/u/cliuser/` is the home directory for the user. You can set the user home directory differently in your z/OS environment depending on your installation requirements. 2. Follow the installation guide for IBM Open Enterprise Python for z/OS to install Python 3.8 or later if it isn't already installed.	

Task	Description	Skills required

Set USS environment variables.	Add environment variables to the profile. You can add these either to the `/u/cliuser/.profile` file for an individual user (`cliuser`) or to the `/etc/profile` file for all users.
	
	Note
	This pattern assumes that Python has been installed in the `/u/awscli/python` directory. If your installation directory is different, update the code accordingly.
	
	`# Python configuration`
	`export BPXKAUTOC`
	`VT='ON'`
	`export CEERUNOPT`
	`S='FILETAG(AUTOCVT`
	`,AUTOTAG) POSIX(ON)'`
	`export TAGREDIR_ERR=txt`
	`export TAGREDIR_IN=txt`
	`export TAGREDIR_`
	`OUT=txt`
	
	`# AWS CLI configura`
	`tion`
	`export PATH=/u/c`
	`liuser/python/bin:`
	`$PATH`
		Mainframe z/OS administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre>export PYTHONPATH=/u/cliuser/python:$PYTHONPATH</pre></td><td></td></tr>
<tr><td>Test the Python installation.</td><td>Run the python command:<pre>python --version</pre>The output should confirm that you have Python 3.8 or later installed correctly.</td><td>Mainframe z/OS administrator</td></tr>
<tr><td>Verify or install pip.</td><td>1. The pip command is usually installed automatically when you install Python from the IBM website. To verify, run the command:<pre>pip ––version</pre>If pip is installed, this command should show the installed version.
2. If the pip command isn't found, install pip by running the following command:<pre>python -m ensurepip --upgrade</pre>For more installation options, see the pip documentation.</td><td>Mainframe z/OS administrator</td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install AWS CLI version 1.</td><td>

1. To install the AWS CLI, run the command:

```
python -m pip install
 awscli
```

The output should be similar to the following.

```
Successfully
 installed PyYAML-6.
0.1 awscli-1.32.23
 botocore-1.34.23
 colorama-0.4.4
 docutils-0.16
 jmespath-1.0.1
pyasn1-0.5.1 python-
dateutil-2.8.2
 rsa-4.7.2 s3transfe
r-0.10.0 urllib3-2
.0.7
```

2. Change the permission of the **aws** executable by running the following command. Make sure to update the placeholder directory `<python_installation_dir>` with your Python installation path.

```
chmod 744 <python_i
nstallation_dir>/b
in/aws
```

</td><td>Mainframe z/OS administrator</td></tr>
</table>

Task	Description	Skills required

	3. Run the following command to test the AWS CLI installation:
	`aws --version`
	The output should show the versions of the AWS CLI, Python, and botocore, similar to the following.
	`aws-cli/1.32.3 Python/3.9.5 OS/390/27.00 botocore/1.34.3`	

Configure AWS CLI access from z/OS

Task	Description	Skills required
Configure the AWS access keys, default Region, and output.	The AWS CLI documentation describes different options for setting up AWS access. You can choose a configuration according to your organizat ion's standards. This example uses the short-term credentia l configuration. 1. Configure the AWS CLI with the following command: `aws configure`	AWS administrator, Mainframe z/OS administr ator, Mainframe z/OS developer

Task	Description	Skills required

	2. Provide the details for the following items when prompted. The access key ID and secret access key values are from the keys you obtained when you set AWS credentials in the [Prerequisites](#) steps.
	
	`AWS Access Key ID [None]: ASIAIOSFODNN7EXAMPLE`
	`AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`
	`Default region name [None]: us-east-1`
	`Default output format [None]:`
	`aws configure set aws_session_token IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONGSTRINGEXAMPLE`
	
	This configuration, including access keys, is stored in the `/u/cliuser/.aws` folder. For security reasons, restrict this folder to allow access only to the users or programs that use the AWS CLI.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Test the AWS CLI.</td><td>1. Run the following command at the command prompt to test the AWS CLI with a simple command:

<code>aws s3 ls</code>

The output should list all the S3 buckets for the configured AWS account without any errors.
2. Follow the instructions in the next two epics to transfer data from USS to Amazon S3. You can choose one of these two options:
• Option 1 (next epic): Interactively transfer an EBCDIC comma-separated value (CSV) file to Amazon S3 and query the file from Amazon Athena.
• Option 2: Transfer an EBCDIC fixed-length dataset to Amazon S3 as a batch job.</td><td>Mainframe z/OS administrator, Mainframe z/OS developer</td></tr>
</table>

Option 1 – Transfer data from USS to Amazon S3 interactively from a USS session

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Download and transfer the sample CSV file.</td><td>1. Download <code>sales-records.csv</code> from the Attachments section. This file provides a sample CSV file with sales records.
2. Transfer the file to z/OS USS.
3. Verify that the <code>/u/cliuser/sales-records.csv</code> file is readable in EBCDIC format in USS by using a text editor of your choice.</td><td>App developer, Mainframe z/OS developer</td></tr>
<tr><td>Create an S3 bucket and upload the CSV file.</td><td>1. Create an S3 bucket to store the CSV file.
<pre>aws s3 mb s3://<s3_bucket_name></pre>where <code><s3_bucket_name></code> is the unique name for a bucket; for example:
<pre>aws s3 mb s3://DOC-EXAMPLE-BUCKET1</pre>2. Upload the CSV file from z/OS USS to the S3 bucket:</td><td>App developer, Mainframe z/OS developer</td></tr>
</table>

Task	Description	Skills required
	``` aws s3 cp <csv_file_path> s3://<s3_bucket_name> ``` For example: ``` aws s3 cp /u/cliuser/sales-records.csv s3://DOC-EXAMPLE-BUCKET1 ``` 3. List the contents of the S3 bucket and confirm that they include the uploaded file: ``` aws s3 ls s3://<s3_bucket_name> ``` For example: ``` aws s3 ls s3://DOC-EXAMPLE-BUCKET1 ```	

Task	Description	Skills required
View the S3 bucket and uploaded file.	1. Sign in to the AWS Management Console and open the Amazon S3 console. 2. Navigate to see the new S3 bucket and the uploaded object. For more information about uploading objects, see Getting started with Amazon S3 in the Amazon S3 documentation.	General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Run a SQL query on an Amazon Athena table.</td><td>
1. Open the Amazon Athena console.
2. Create a new table (for example, <code>DOC-EXAMPLE-BUCKET</code>) by using the CSV data from Amazon S3. For more information, see Querying Amazon S3 Inventory with Amazon Athena in the Amazon S3 documentation.
3. Run the <code>SELECT</code> query against the table to view the data.
<pre>SELECT * FROM
 <table_name>;</pre>
For example:
<pre>SELECT * FROM DOC-
EXAMPLE-BUCKET;</pre>
The output of the SQL query will display the contents of your CSV file.</td><td>General AWS, App developer</td></tr>
</table>

Option 2 – Transfer data from USS to Amazon S3 by using batch JCL

Task	Description	Skills required
Upload the sample file.	1. Download `sales-records-fixed.txt` from	Mainframe z/OS developer

Task	Description	Skills required
	the Attachments section. This is a sample file with sales records. Rename the text file; for example, to `USER.DATA.FIXED` . 2. Transfer the file to z/OS as a fixed-blocked (FB), 256 record length (LRECL), physical sequential (PS) dataset. 3. Use the dataset list utility to verify that the `USER.DATA.FIXED` dataset is readable in EBCDIC format under ISPF option 3.4. See the Additional information section for example output.	

Task	Description	Skills required

Create batch JCL.	Code the batch JCL as follows to create the destination S3 bucket, upload the dataset, and list the bucket content. Make sure to replace the directory name, file names, and bucket name to your own values.
	
	`//AWSCLICP JOB ACTINFO1,'IBMUSER' ,CLASS=A,MSGCLASS= H,MSGLEVEL=(1,1), // NOTIFY=&SYSUID,TIM E=1440`
	`//*---------------------------------------------------`
	`//* Sample job for AWS CLI`
	`//*---------------------------------------------------`
	`//USSCMD EXEC PGM=BPXBA TCH`
	`//STDERR  DD SYSOUT=*`
	`//STDOUT  DD SYSOUT=*`
	`//STDENV  DD *`
	`export PATH=/u/c liuser/python/bin: $PATH`
	`//STDPARM DD *`
	`SH`
	`export _BPXK_AUT OCVT=ON;`
	`aws s3 mb s3://DOC- EXAMPLE-BUCKET2;`
		Mainframe z/OS developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

```
 cp "//'USER.DATA.FIXE
D'" /tmp/tmpfile;
 aws s3 cp /tmp/tmpf
ile s3://DOC-EXAMPLE-
BUCKET2/USER.DATA.F
IXED;
 rm /tmp/tmpfile;
 aws s3 ls s3://DOC-
EXAMPLE-BUCKET2;
/*
```

</td><td></td></tr>
<tr><td>Submit the batch JCL job.</td><td>

1. Submit the JCL job that you coded in the previous step.
2. Check the status of the job in System Display and Search Facility (SDSF). If successful, the job should end with return code 0.
3. The standard output (STDOUT) from the job log shows the successful bucket creation, dataset upload, and list of bucket contents. For a sample screen illustration, see the [Additional information](#) section.

</td><td>Mainframe z/OS developer</td></tr>
</table>

Task	Description	Skills required
View the dataset uploaded to the S3 bucket.	1. Sign in to the AWS Management Console and open the [Amazon S3 console](). 2. Navigate to see the uploaded file in the test bucket. 3. You can further process the `USER.DATA.FIXED` file or analyze it by using analytics services such as Amazon Redshift.	General AWS

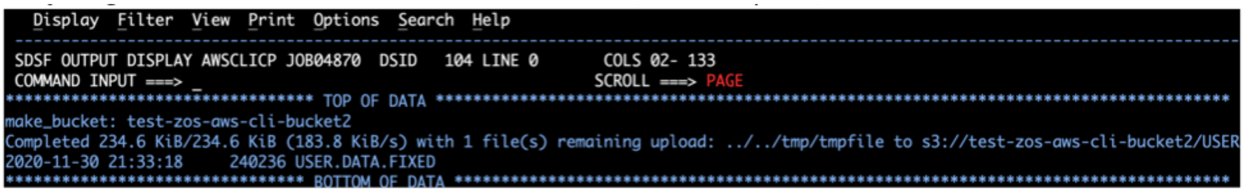
Related resources

- [AWS CLI version 1 documentation]()
- [AWS Mainframe Modernization CLI Command Reference]()
- [AWS Mainframe Modernization]()

Additional information

USER.DATA.FIXED in ISPF option 3.4 (dataset list utility)

SYSOUT of the submitted batch job



Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Back up and archive mainframe data to Amazon S3 using BMC AMI Cloud Data

Santosh Kumar Singh, Gilberto Biondo, and Maggie Li, Amazon Web Services

Mikhael Liberman, Model9 Mainframe Software

Summary

This pattern demonstrates how to back up and archive mainframe data directly to Amazon Simple Storage Service (Amazon S3), and then recall and restore that data to the mainframe by using BMC AMI Cloud Data (previously known as Model9 Manager). If you are looking for a way to modernize your backup and archive solution as part of a mainframe modernization project or to meet compliance requirements, this pattern can help meet those goals.

Typically, organizations that run core business applications on mainframes use a virtual tape library (VTL) to back up data stores such as files and logs. This method can be expensive because it consumes billable MIPS, and the data stored on tapes outside the mainframe is inaccessible. To avoid these issues, you can use BMC AMI Cloud Data to quickly and cost-effectively transfer operational and historical mainframe data directly to Amazon S3. You can use BMC AMI Cloud Data to back up and archive data over TCP/IP to AWS while taking advantage of IBM z Integrated Information Processor (zIIP) engines to reduce cost, parallelism, and transfer times.

Prerequisites and limitations

Prerequisites

- An active AWS account
- BMC AMI Cloud Data with a valid license key
- TCP/IP connectivity between the mainframe and AWS
- An AWS Identity and Access Management (IAM) role for read/write access to an S3 bucket
- Mainframe security product (RACF) access in place to run BMC AMI Cloud processes
- A BMC AMI Cloud z/OS agent (Java version 8 64-bit SR5 FP16 or later) that has available network ports, firewall rules permitting access to S3 buckets, and a dedicated z/FS file system
- Requirements met for the BMC AMI Cloud management server

Limitations

- BMC AMI Cloud Data stores its operational data in a PostgreSQL database that runs as a Docker container on the same Amazon Elastic Compute Cloud (Amazon EC2) instance as the management server. Amazon Relational Database Service (Amazon RDS) is not currently supported as a backend for BMC AMI Cloud Data. For more information about the latest product updates, see What's New? in the BMC documentation.
- This pattern backs up and archives z/OS mainframe data only. BMC AMI Cloud Data backs up and archives only mainframe files.
- This pattern doesn’t convert data into standard open formats such as JSON or CSV. Use an additional transformation service such as BMC AMI Cloud Analytics (previously known as Model9 Gravity) to convert the data into standard open formats. Cloud-native applications and data analytics tools can access the data after it's is written to the cloud.

Product versions

- BMC AMI Cloud Data version 2.x

Architecture

Source technology stack

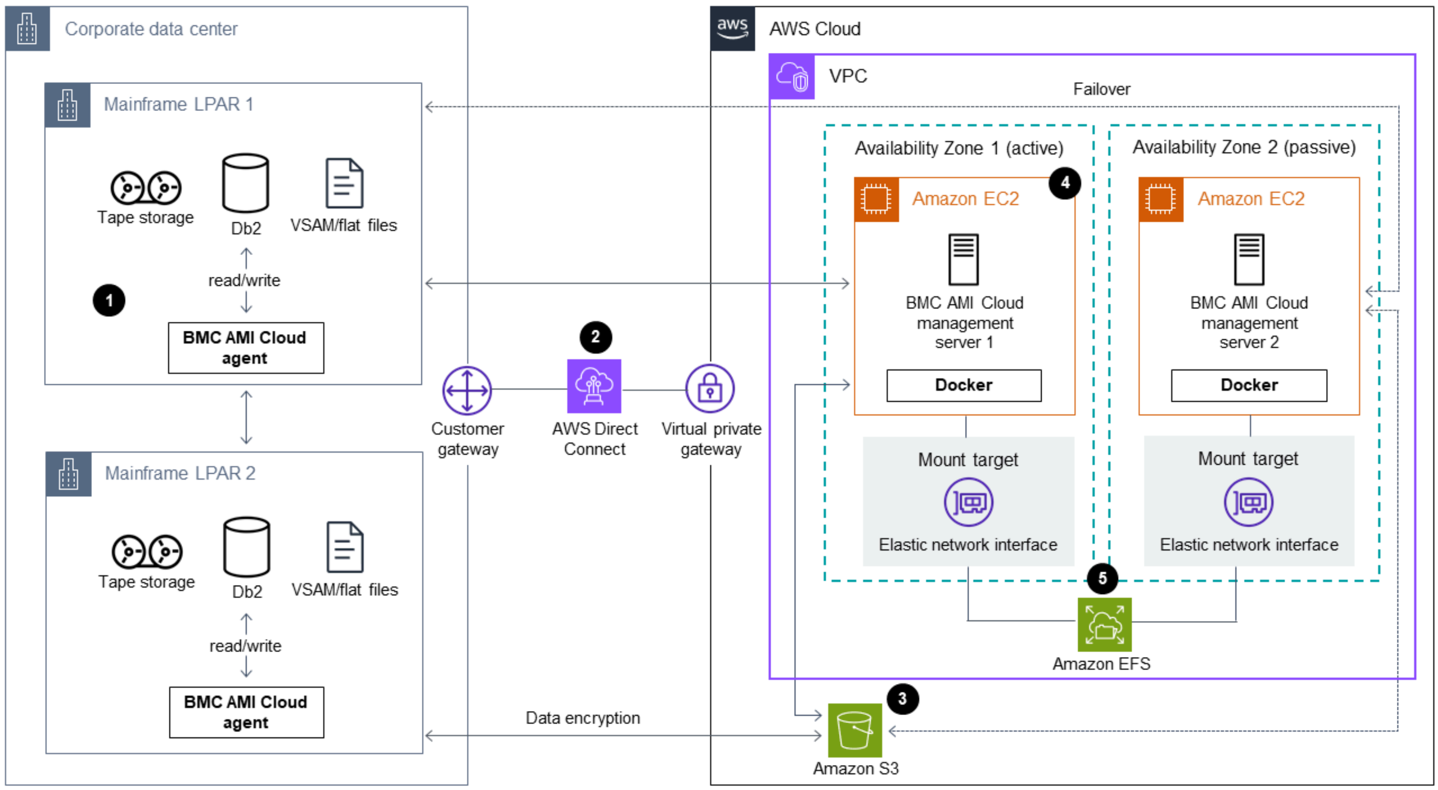
- Mainframe running z/OS
- Mainframe files such as datasets and z/OS UNIX System Services (USS) files
- Mainframe disk, such as a direct-access storage device (DASD)
- Mainframe tape (virtual or physical tape library)

Target technology stack

- Amazon S3
- Amazon EC2 instance in a virtual private cloud (VPC)
- AWS Direct Connect
- Amazon Elastic File System (Amazon EFS)

Target architecture

The following diagram shows a reference architecture where BMC AMI Cloud Data software agents on a mainframe drive the legacy data backup and archive processes that store the data in Amazon S3.



The diagram shows the following workflow:

1. BMC AMI Cloud Data software agents run on mainframe logical partitions (LPARs). The software agents read and write mainframe data from DASD or tape directly to Amazon S3 over TCP/IP.
2. AWS Direct Connect sets up a physical, isolated connection between the on-premises network and AWS. For enhanced security, run a site-to-site VPN on top of AWS Direct Connect to encrypt data in transit.
3. The S3 bucket stores mainframe files as object storage data, and BMC AMI Cloud Data agents directly communicate with the S3 buckets. Certificates are used for HTTPS encryption of all communications between the agent and Amazon S3. Amazon S3 data encryption is used to encrypt and protect the data at rest.
4. BMC AMI Cloud Data management servers run as Docker containers on EC2 instances. The instances communicate with agents running on mainframe LPARs and S3 buckets.
5. Amazon EFS is mounted on both active and passive EC2 instances to share the Network File System (NFS) storage. This is to make sure that metadata related to a policy created on the

management server isn't lost in the event of a failover. In the event of a failover by the active server, the passive server can be accessed without any data loss. If the passive server fails, the active server can be accessed without any data loss.

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve nearly any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.
- AWS Direct Connect links your internal network to a AWS Direct Connect location over a standard Ethernet fiber-optic cable. With this connection, you can create virtual interfaces directly to public AWS services while bypassing internet service providers in your network path.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

BMC tools

- BMC AMI Cloud management server is a GUI application that runs as a Docker container on an Amazon Linux Amazon Machine Image (AMI) for Amazon EC2. The management server provides the functionality to manage BMC AMI Cloud activities such as reporting, creating and managing policies, running archives, and performing backups, recalls, and restores.
- BMC AMI Cloud agent runs on an on-premises mainframe LPAR that reads and writes files directly to object storage by using TCP/IP. A started task runs on a mainframe LPAR and is responsible for reading and writing backup and archive data to and from Amazon S3.
- BMC AMI Cloud Mainframe Command Line Interface (M9CLI) provides you with a set of commands to perform BMC AMI Cloud actions directly from TSO/E or in batch operations, without the dependency on the management server.

Epics

Create an S3 bucket and IAM policy

Task	Description	Skills required
Create an S3 bucket.	Create an S3 bucket to store the files and volumes that you want to back up and archive from your mainframe environment.	General AWS

Create an IAM policy.	All BMC AMI Cloud management servers and agents require access to the S3 bucket that you created in the previous step.
	
	To grant the required access, create the following IAM policy:
	
	`{`
	`    "Version":`
	` "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Sid":`
	` "Listfolder",`
	`            "Action": [`
	
	` "s3:ListBucket",`
	
	` "s3:GetBucketLocat`
	`ion",`
	
	` "s3:ListBucketVers`
	`ions"`
	`            ],`
	`            "Effect":`
	` "Allow",`
		General AWS

Task	Description	Skills required

	``
	              "Resource":
	 [
	
	 "arn:aws:s3:::<Bucket
	 Name>"
	              ]
	          },
	          {
	              "Sid":
	 "Objectaccess",
	              "Effect":
	 "Allow",
	              "Action": [
	
	 "s3:PutObject",
	
	 "s3:GetObjectAcl",
	
	 "s3:GetObject",
	
	 "s3:DeleteObjectVe
	rsion",
	
	 "s3:DeleteObject",
	
	 "s3:PutObjectAcl",
	
	 "s3:GetObjectVersion"
	              ],
	              "Resource":
	 [
	
	 "arn:aws:s3:::<Bucket
	 Name>/*"
	              ]
	          }
	     ]
	}
	``	

Get the BMC AMI Cloud software license and download the software

Task	Description	Skills required
Get a BMC AMI Cloud software license.	To get a software license key, contact the [BMC AMI Cloud team](). The output of the z/OS `D M=CPU` command is required for generating a license.	Build lead
Download the BMC AMI Cloud software and license key.	Obtain the installation files and license key by following the instructions in the [BMC documentation]().	Mainframe infrastructure administrator

Install the BMC AMI Cloud software agent on the mainframe

Task	Description	Skills required
Install the BMC AMI Cloud software agent.	1. Before you start the installation process, verify that [minimum software and hardware requirements]() for the agent have been met. 2. To install the agent, follow the instructions in the [BMC documentation](). 3. After the agent starts running on the mainframe LPAR, check for the `ZM91000I MODEL9 BACKUP AGENT INITIALIZED` message in the spool. Verify that the	Mainframe infrastructure administrator

Task	Description	Skills required
	connectivity is successfully established between the agent and the S3 bucket by looking for the `Object store connectivity has been established successfully` message in the agent's STDOUT.	

Set up a BMC AMI Cloud management server on an EC2 instance

Task	Description	Skills required
Create Amazon EC2 Linux 2 instances.	Launch two Amazon EC2 Linux 2 instances in different Availability Zones by following the instructions from [Step 1: Launch an instance](#) in the Amazon EC2 documentation. The instance must meet the following recommended hardware and software requirements: • CPU – Minimum 4 cores • RAM – Minimum 8 GB • Drive – 40 GB • Recommended EC2 instance – C5.xlarge • OS – Linux	Cloud architect, Cloud administrator

Task	Description	Skills required
	• Software – Docker, unzip, vi/VIM • Network bandwidth – Minimum 1 GB For more information, see the BMC documentation.	
Create an Amazon EFS file system.	Create an Amazon EFS file system by following the instructions from Step 1: Create your Amazon EFS file system in the Amazon EFS documentation. When creating the file system, do the following: • Choose the Standard storage class. • Choose the same VPC that you used to launch your EC2 instances.	Cloud administrator, Cloud architect

Task	Description	Skills required

Install Docker and configure the management server.	**Connect to your EC2 instances:**
	
	Connect to your EC2 instances by following the instructions from Connect to your Linux instance in the Amazon EC2 documentation.
	
	Configure your EC2 instances :
	
	For each EC2 instance, do the following:
	
	1. To install Docker, run the command:
	`sudo yum install docker`
	2. To start Docker, run the command:
	`sudo service docker start`
	3. To validate the status of Docker, run the command:
	`sudo service docker status`
	4. In the `/etc/selinux` folder, change the `config` file to `SELINUX=permissive` .
		Cloud architect, Cloud administrator

Task	Description	Skills required
	5. Upload the `model9-v2.x.y_build_build-id-server.zip` and `VerificationScripts.zip` files (that you downloaded earlier) to a temporary folder in one of the EC2 instances (for example, into the `/var/tmp` folder in your instance). 6. To go to the `tmp` folder, run the command: `cd/var/tmp` 7. To unzip the verification script, run the command: `unzip VerificationScripts.zip` 8. To change the directory, run the command: `cd /var/tmp/sysutils/PrereqsScripts` 9. To run the verification script, run the command: `./M9VerifyPrereqs.sh` 10After the verification script prompts for the input,	

Task	Description	Skills required
	enter the Amazon S3 URL and port number. Then, enter the z/OS IP/DNS and port number. **Note** The script runs a check to confirm that EC2 instance can connect with the S3 bucket and agent that's running on the mainframe. If a connection is established, a success message is displayed.	

Task	Description	Skills required

Install the management server software.	1. Create a folder and subfolder in the root directory (for example, `/data/model9`) in the EC2 instance that you plan to make the active server.
	2. To install the `amazon-efs-utils` package and to mount the Amazon EFS file system created earlier, run the following commands:
	`sudo yum install -y amazon-efs-utils`
	`sudo mount -t efs -o tls <File System ID>:/ /data/model9`
	3. To update the EC2 instance's `/etc/fstab` file with an entry for the Amazon EFS file system (so that Amazon EFS is automatically remounted when Amazon EC2 reboots), run the command:
	`<Amazon-EFS-file-system-id>:/ /data/model9 efs defaults,_netdev 0 0`
	4. To define the path to the BMC AMI Cloud installation files and the target
		Cloud architect, Cloud administrator

Task	Description	Skills required

	installation location, run the following commands to export variables:
	`export MODEL9_HOME=/data/model9`
	`export M9INSTALL=/var/tmp`
	Note: We recommend that you add these EXPORT commands to your `.bashrc` script.
	5. To change the directory, run the `cd $MODEL9_HOME` command, and then create another subdirectory by running the `mkdir diag` command.
	6. To unzip the installation file, run the command:
	`unzip $M9INSTALL/model9-<v2.x.y>_build_<build-id>-server.zip`
	Note: Replace `x.y` (the version) and	

Task	Description	Skills required

	`build-id` with your values.
	
	7. To deploy the application, run the following commands:
	
	`docker load -i $MODEL9_HOME/model9-<v2.x.y>_build_<build-id>.docker`
	`docker load -i $MODEL9_HOME/postgres-12.10-x86.docker.gz`
	
	Note
	Replace `v2.x.y` (the version) and `build-id` with your values.
	
	8. In the `$MODEL9_HOME/conf` folder, update the `model9-local.yml` file.
	
	Note
	Some of the parameters have default values and others can be updated as necessary. For more	

Task	Description	Skills required

	information, see the instructions in the `model9-local.yml` file.
	9. Create a file called `$MODEL9_HOME/conf`, and then add the following parameters to the file:
	`TZ=America/New_York EXTRA_JVM_ARGS=-Xmx2048m`
	10. To create a Docker network bridge, run the command:
	`docker network create -d bridge model9network`
	11. To start the PostgreSQL database container for BMC AMI Cloud, run the following command:
	`docker run -p 127.0.0.1:5432:5432 \ -v $MODEL9_HOME/db/data:/var/lib/postgresql/data:z \ --name model9db --restart unless-stopped \ --network model9network \`	

Task	Description	Skills required

```
-e POSTGRES_PASSWORD=
model9 -e POSTGRES_
DB=model9 -d
 postgres:12.10
```

12. After the PostgreSQL container starts running, run the following command to start the application server:

```
docker run -d -p
 0.0.0.0:443:443 -p
 0.0.0.0:80:80 \
--sysctl net.ipv4.
tcp_keepalive_time
=600 \
--sysctl net.ipv4.
tcp_keepalive_intv
l=30 \
--sysctl net.ipv4.
tcp_keepalive_prob
es=10 \
-v $MODEL9_HOME:/mode
l9:z -h $(hostname)
 --restart unless-st
opped \
--env-file $MODEL9_H
OME/conf/model9.env
 \
--network model9net
work \
--name model9-v2.x.y
 model9:<v2.x.y>.<b
uild-id>
```

Note

Replace `v2.x.y` (the version) and

Task	Description	Skills required
	build-id with your values. 13. To check the health status of both containers, run the command: `docker ps -a` 14. To install a management server on the passive EC2 instances, repeat steps 1–4, 7, and 10–13. **Note** To troubleshoot issues, go to the logs stored in the `/data/model9/logs/` folder. For more information, see the [BMC documentation]().	

Add an agent and define a backup or archive policy on the BMC AMI Cloud management server

Task	Description	Skills required
Add a new agent.	Before you add a new agent, confirm the following: • A BMC AMI Cloud agent is running on the mainframe LPAR and has been	Mainframe storage administrator or developer

Task	Description	Skills required
	initialized completely. Identify the agent by looking for the `ZM91000I MODEL9 BACKUP AGENT INITIALIZED` initialization message in the spool. • A Docker container for the management server is fully initialized and running. You must create an agent on the management server before you define any backup and archive policies. To create the agent, do the following: 1. Use a web browser to access the management server that's deployed on your Amazon EC2 machine, and then log in with your mainframe credentials. 2. Choose the **AGENTS** tab, and then choose **ADD NEW AGENT**. 3. For **Name**, enter the agent name. 4. For **Hostname/IP Address**, enter the host name or IP address of your mainframe. 5. For **Port**, enter your port number. 6. Choose **TEST CONNECTION**. You can see	

Task	Description	Skills required
	a **success** message if the connectivity is successfully established. 7. Choose **CREATE**. After the agent is created, you'll see the **connected** status against the object storage and mainframe agent in a new window that appears in the table.	
Create a backup or archive policy.	1. Choose **POLICIES**. 2. Choose **CREATE POLICY**. 3. On the **CREATE A NEW POLICY** page, enter your policy specifications. **Note** For more information about the available specifications, see [Creating a new policy]() in the BMC documentation. 4. Choose **FINISH**. 5. The new policy is now listed as a table. To see this table, choose the **POLICIES** tab.	Mainframe storage administrator or developer

Run the backup or archive policy from the management server

Task	Description	Skills required
Run the backup or archive policy.	Run the data backup or archive policy that you created earlier from the management server either manually or automatically (based on a schedule). To run the policy manually: 1. Choose the **POLICIES** tab from the navigation menu. 2. On the right side of the table for the policy that you want to run, choose the three-dot menu. 3. Choose **Run Now**. 4. In the pop-up confirmation window, choose **YES, RUN POLICY NOW**. 5. After the policy runs, verify the run status in the policy activity section. 6. For the policy that ran, choose the three-dot menu, and then choose **View Run Log** to see the logs. 7. To verify that the backup was created, check the S3 bucket.	Mainframe storage administrator or developer
Restore the backup or archive policy.	1. On the navigation menu, choose the **POLICIES** tab.	Mainframe storage administrator or developer

Task	Description	Skills required
	2. Choose the policy to run your restore process on. This will list all the backup or archive activities that ran in the past for that specific policy. 3. To select the backups that you want to restore, choose the **Date-time** column. The `file/Volume/Storage` group name shows the run details of the policy. 4. On the right side of the table, choose the three-dot menu, and then choose **RESTORE**. 5. In the pop-up window, enter your target name, volume, and storage group, and then choose **RESTORE**. 6. Enter your mainframe credentials, and then choose **RESTORE** again. 7. To verify that the restore was successful, check the logs or the mainframe.	

Run the backup or archive policy from the mainframe

Task	Description	Skills required
Run the backup or archive policy by using M9CLI.	Use the M9CLI to perform backup and restore processes from TSO/E, REXX, or through JCLs without setting up rules on the BMC AMI Cloud management server. **Using TSO/E:** If you use TSO/E, make sure that `M9CLI REXX` is concatenated to `TSO`. To back up a dataset through TSO/E, use the `TSO M9CLI BACKDSN <DSNAME>` command. **Note** For more informati on about M9CLI commands, see CLI reference in the BMC documentation. **Using JCLs:** To run the backup and archive policy by using JCLs, run the `M9CLI` command. **Using batch operations:**	Mainframe storage administr ator or developer

Task	Description	Skills required

	The following example shows you how to archive a dataset by running the `M9CLI` command in batch:
	`//JOBNAME JOB …`
	`//M9CLI EXEC PGM=IKJEFT01`
	`//STEPLIB DD DISP=SHR, DSN=<MODEL9 LOADLIB>`
	`//SYSEXEC DD DISP=SHR, DSN=<MODEL9 EXEC LIB>`
	`//SYSTSPRT DD SYSOUT=*`
	`//SYSPRINT DD SYSOUT=*`
	`//SYSTSIN DD TSO M9CLI ARCHIVE`
	`M9CLI ARCHIVE <DSNNAME OR DSN PATTERN>`
	`/`	

Task	Description	Skills required
Run the backup or archive policy in JCL batch.	BMC AMI Cloud provides a sample JCL routine called **M9SAPIJ**. You can customize **M9SAPIJ** to run a specific policy created on the management server with a JCL. This job can also be part of a batch scheduler for running backup and restore processes automatically. The batch job expects the following mandatory values: • Management server IP address/host name • Port number • Policy ID or policy name (which is created on the management server) **Note** You can also change other values by following the instructions on the sample job.	Mainframe storage administrator or developer

Related resources

- Mainframe Modernization with AWS (AWS documentation)

- How Cloud Backup for Mainframes Cuts Costs with Model9 and AWS (AWS Partner Network Blog)
- How to Enable Mainframe Data Analytics on AWS Using Model9 (AWS Partner Network Blog)
- AWS Direct Connect Resiliency Recommendations (AWS documentation)
- BMC AMI Cloud documentation (BMC website)

Build COBOL Db2 programs by using AWS Mainframe Modernization and AWS CodeBuild

Luis Gustavo Dantas and Eduardo Zimelewicz, Amazon Web Services

Summary

This pattern explains how to create a simple AWS CodeBuild project to precompile and bind COBOL Db2 programs by using the AWS Mainframe Modernization Replatform tools. This enables the deployment and execution of these programs in the AWS Mainframe Modernization Replatform runtime environment.

COBOL, a business-oriented programming language, powers many critical applications due to its reliability and readability. IBM Db2, a relational database management system, manages large volumes of data efficiently and integrates with COBOL programs through SQL. Together, COBOL and Db2 form the backbone of mission-critical operations in industries such as finance and government, despite the emergence of newer technologies.

Migrating COBOL and Db2 components from the mainframe environment to other platforms leads to challenges such as platform compatibility, integration complexity, data migration, and performance optimization. Moving these critical components requires careful planning, technical expertise, and resources to ensure a smooth migration while maintaining reliability and functionality.

The AWS Mainframe Modernization service provides tools and resources to replatform mainframe applications and databases to run on AWS infrastructure, such as Amazon Elastic Compute Cloud (Amazon EC2) instances. This involves moving mainframe workloads to the cloud without major code changes.

The Db2 precompile and bind process is essential for optimizing the performance and reliability of database applications. Precompilation transforms embedded SQL statements into executable code, which reduces runtime overhead and enhances efficiency. The bind process links the precompiled code with database structures, facilitating access paths and query optimization. This process ensures data integrity, improves application responsiveness, and guards against security vulnerabilities. Properly precompiled and bound applications minimize resource consumption, enhance scalability, and mitigate the risks of SQL injection attacks.

Prerequisites and limitations

Prerequisites

- An AWS account and administrative-level console access.
- An IBM Db2 database system, such as IBM Db2 for z/OS or Db2 for Linux, Unix, and Windows (LUW).
- The IBM Data Server Client software, which is available for download from the IBM website. For more information, see IBM Data Server Client and Data Server Driver types.
- A COBOL Db2 program to be compiled and bound. Alternatively, this pattern provides a basic sample program that you can use.
- A virtual private cloud (VPC) on AWS with a private network. For information about creating a VPC, see the Amazon Virtual Private Cloud (Amazon VPC) documentation.
- A source control repository such as GitHub or GitLab.

Limitations

- For AWS CodeBuild quotas, see Quotas for AWS CodeBuild.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

Source technology stack

The source stack includes:

- COBOL programs that use a Db2 database to store data
- IBM COBOL compiler and Db2 for z/OS precompiler
- Other parts of the mainframe setup, such as the file system, transaction manager, and spool

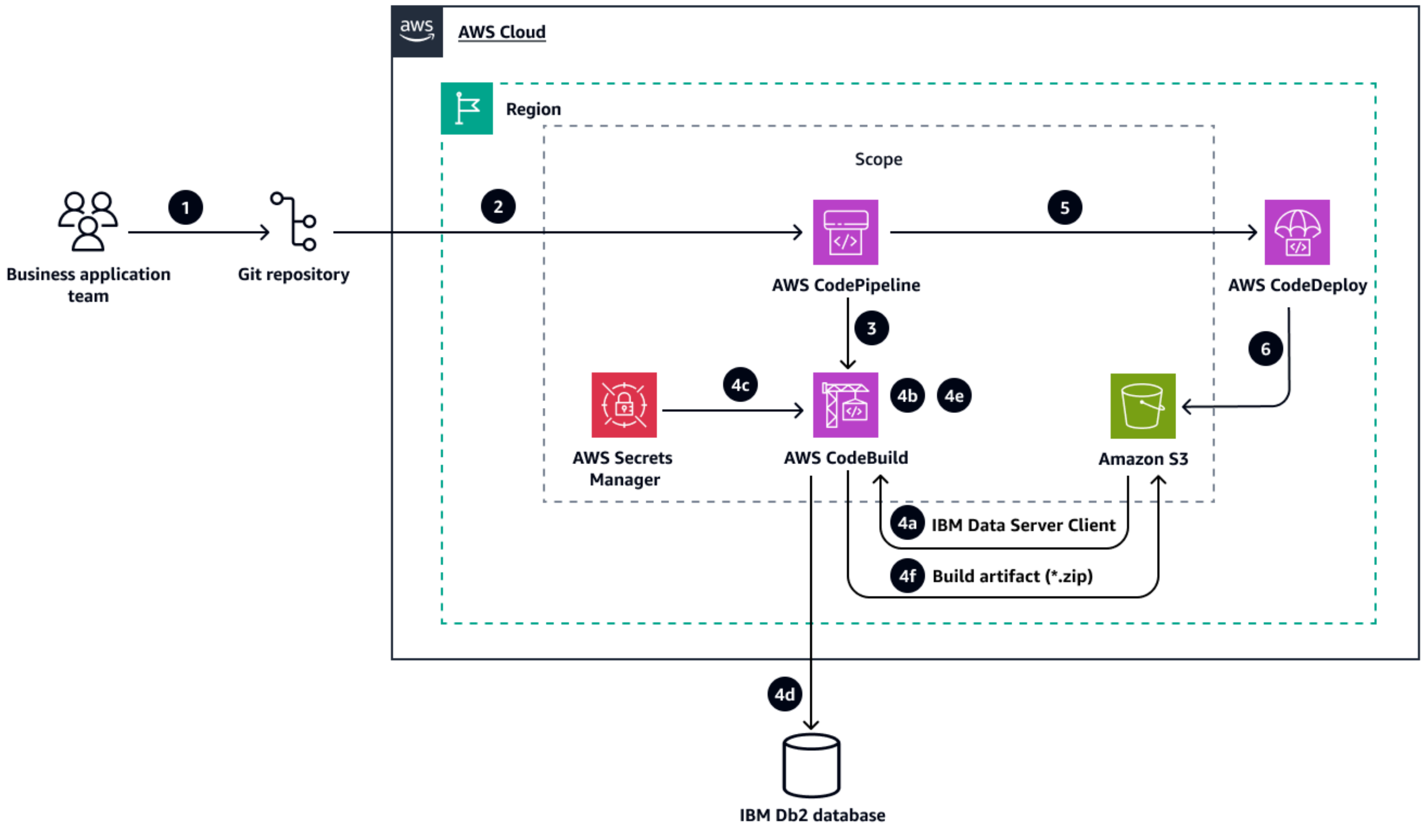
Target technology stack

This pattern's approach works for two options: moving data from Db2 for z/OS to Db2 for LUW, or staying on Db2 for z/OS. The target architecture includes:

- COBOL programs that use a Db2 database to store data
- AWS Mainframe Modernization Replatform compilation tools

- AWS CodeBuild as the infrastructure to build the application
- Other AWS Cloud resources such as Amazon Linux

Target architecture



The diagram illustrates the following:

1. The user uploads their code to a source control repository such as GitHub or GitLab.
2. AWS CodePipeline notices the change and gets the code from the repository.
3. CodePipeline starts AWS CodeBuild and sends the code.
4. CodeBuild follows the instructions in the `buildspec.yml` template (provided in the [Additional information](#) section) to:
   a. Get the IBM Data Server Client from an Amazon Simple Storage Service (Amazon S3) bucket.
   b. Install and set up the IBM Data Server Client.
   c. Retrieve Db2 credentials from AWS Secrets Manager.
   d. Connect to the Db2 server.
   e. Precompile, compile, and bind the COBOL program.

    f. Save the finished products in an S3 bucket for AWS CodeDeploy to use.

5. CodePipeline starts CodeDeploy.
6. CodeDeploy coordinates its agents, which are already installed in the runtime environments. The agents fetch the application from Amazon S3 and install it based on the instructions in `appspec.yml`.

To keep things simple and focused on the build, the instructions in this pattern cover steps 1 through 4 but don't include the deployment of the COBOL Db2 program.

Automation and scale

For simplicity, this pattern describes how to provision resources manually. However, there are numerous automation options available, such as AWS CloudFormation, AWS Cloud Development Kit (AWS CDK), and HashiCorp Terraform, which automate these tasks. For more information, see the AWS CloudFormation and AWS CDK documentation.

Tools

AWS services

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeDeploy automates deployments to Amazon EC2 or on-premises instances, AWS Lambda functions, or Amazon Elastic Container Service (Amazon ECS) services.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS Mainframe Modernization provides tools and resources to help you plan and implement migration and modernization from mainframes to AWS managed runtime environments.

Other tools

- **Amazon ECR image for AWS Mainframe Modernization Replatform tools**. To compile a COBOL application, you'll need to initiate CodeBuild by using an Amazon Elastic Container Registry (Amazon ECR) image that contains the AWS Mainframe Modernization Replatform tools:

```
673918848628.dkr.ecr.<your-region>.amazonaws.com/m2-enterprise-build-tools:9.0.7.R1
```

For more information about the ECR image available, see the tutorial in the *AWS Mainframe Modernization User Guide*.

- IBM Data Server Client software is essential for precompiling and binding COBOL Db2 programs in CodeBuild. It acts as a bridge between the COBOL compiler and Db2.

Best practices

- Not every COBOL program relies on Db2 as its data persistence layer. Make sure that compilation directives for accessing Db2 are applied only to COBOL programs that are specifically designed to interact with Db2. Implement a logic to distinguish between COBOL Db2 programs and COBOL programs that do not use Db2.
- We recommend that you avoid compiling programs that haven't been modified. Implement a process to identify which programs require compilation.

Epics

Create the cloud infrastructure

Task	Description	Skills required
Create an S3 bucket to host the IBM Data Server Client and pipeline artifacts.	You need to set up an S3 bucket to (a) upload the IBM Data Server Client, (b) store your code from the repository, and (c) store the results of the build process. 1. Sign in to the AWS Management Console, and open the Amazon S3 console. 2. Choose an existing S3 bucket or create a new bucket. Note the bucket's	General AWS

Task	Description	Skills required
	Amazon Resource Name (ARN) for future use. For ways to create an S3 bucket, see the [Amazon S3 documentation.]()	
Upload the IBM Data Server Client to the S3 bucket.	1. On the [Amazon S3 console](), choose the bucket to open it. 2. Choose **Create folder**, specify its name as **client**, and then choose **Create folder**. 3. Open the **client** folder, choose **Upload**, **Add files**. 4. Choose the **IBM Data Server Client** file that you previously downloaded from the [IBM website]() to your local file system. The file name should be similar to `v11.5.8_linuxx64_client.tar.gz` or `v11.5.9_linuxx64_client.tar.gz` . 5. Choose **Open**, **Upload** and wait for the upload to complete. 6. On the **Files and folders** tab, choose the Data Server Client and note its S3 URI.	General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create an AWS Secrets Manager secret for your Db2 credentials.</td><td>

To create a secret to securely store your DB2 credentials:

1. On the [Secrets Manager console](https://console.aws.amazon.com/secretsmanager/), choose **Store a new secret**.
2. In the **Choose secret type** pane, choose **Another type of secret** and **Plaintext**.
3. In the **Plaintext** box, type your Db2 credentials by using the following JSON structure.

```
{
    "username": "<your-db2-user-name>",
    "password": "<your-db2-password>",
    "db2node": "db2dev",
    "db2host": "<your-db2-hostname-or-IP>",
    "db2port": <your-db2-port>,
    "db2name": "<your-db2-location>",
    "qualifier": "<your-db2-qualifier>"
}
```

</td><td>General AWS</td></tr>
</table>

Task	Description	Skills required
	4. Choose **Next** and give the secret a name such as `dev-db2-cred` . 5. Choose **Next**, **Next**, and **Store**. For more information about creating secrets, see the [Secrets Manager documentation].	

Task	Description	Skills required
Verify that Db2 is accessible from the VPC subnet.	AWS CodeBuild needs a connection to the Db2 server so that the Data Server Client can perform precompilation and bind operations. Make sure that CodeBuild can reach the Db2 server through a secure connection. 1. Open the [Amazon VPC console](https://console.aws.amazon.com/vpc/). 2. On the navigation pane, choose **Subnets** and write down the IDs and **IPv4 CIDRs** of the **private subnets** where CodeBuild will work. 3. Update the current network access control settings for your Db2 system by introducing an inbound rule. This rule should enable custom TCP access to the Db2 port exclusively from the subnet CIDRs that are associate d with your CodeBuild project.	Network administrator, General AWS

Create the application artifacts

Task	Description	Skills required

Create the COBOL Db2 asset.	1. If you want to use a simple COBOL Db2 example, save the following source code as `CDB2SMP.cbl` . Or, you can replace this example with a program that you already own.
	``
	       IDENTIFICATION DIVISION.
	       PROGRAM-ID. CDB2SMP.
	       DATA DIVISION.
	       WORKING-STORAGE SECTION.
	       01 WS-NAME PIC X(100).
	       PROCEDURE DIVISION.
	           EXEC SQL
	             SELECT NAME INTO :WS-NAME
	             FROM SYSIBM.SYSTABLES
	           END-EXEC
	           GOBACK.
	``
	2. Commit the changes and push the file to your repository.
		App developer

Create the `buildspec.yml` file.	1. Generate the `buildspec.yml` file based on the example that's provided in	AWS DevOps

Task	Description	Skills required
	the [Additional information](#) section. 2. Commit the changes and push the file to your repository.	
Connect your repository to CodePipeline.	1. Open the [AWS Developer Tools console](#). 2. In the navigation pane, choose **Settings, Connections**. 3. Follow the [instructions](#) in the Developer Tools console documentation for the source provider of your choice. You will need the Amazon Resource Name (ARN) for the connection when you create the AWS Identity and Access Management (IAM) policy for CodePipeline in a later step.	AWS DevOps

Configure permissions

Task	Description	Skills required
Create an IAM policy for CodeBuild.	The CodeBuild project requires access to some resources, including Secrets Manager and Amazon S3.	General AWS

Task	Description	Skills required
	To set up the necessary permissions: 1. Open the IAM console. 2. In the navigation pane, choose **Policies**, **Create Policy**, and then select the CodeBuild service. 3. Switch the formatting from **Visual** to **JSON**, and copy the **CodeBuild policy** that's provided in the Additional information section to the **Policy editor** field. 4. Name and save this policy for future reference in the next step. For more information about creating IAM policies, see the IAM documentation.	

Task	Description	Skills required
Create an IAM role for CodeBuild.	To make the security policies available for CodeBuild, you need to configure an IAM role. To create this role: 1. On the [IAM console](), in the navigation pane, choose **Roles**, **Create Role**. 3. For **Trusted entity type**, keep the default **AWS service** setting. 4. For **Use case**, select the CodeBuild service, and then choose **Next**. 4. In the list of available IAM policies, locate the policy you created for CodeBuild, and then choose **Next** to attach it to the role. 5. Specify a name for the role, and choose **Create role** to save it for future reference in CodeBuild. For more information about creating an IAM role for an AWS service, see the [IAM documentation]().	General AWS

Task	Description	Skills required
Create an IAM policy for CodePipeline.	The AWS CodePipeline pipeline requires access to some resources, including your code repository and Amazon S3. Repeat the steps provided previously for CodeBuild to create an IAM policy for CodePipeline (in step 2, choose **CodePipeline** instead of **CodeBuild**).	AWS DevOps

Task	Description	Skills required

Create an IAM role for CodePipeline.	To make the security policies available for CodePipeline, you need to configure an IAM role.
	
	To create this role:
	
	1. On the [IAM console](https://console.aws.amazon.com/iam/), choose **Roles**, **Create Role**.
	2. For **Trusted entity type**, choose **Custom trust policy**.
	
	A policy with an empty `Principal` element will be displayed.
	3. On the `Principal` line, between the braces, add:
	
	`"Service": "codepipeline.amazonaws.com"`
	
	The trust policy will look like this:
	
	`{ "Version": "2012-10-17", "Statement": [ { "Sid": "Statement1", "Effect": "Allow", "Principal": {`
		AWS DevOps

Task	Description	Skills required

	``
	          "Service":
	  "codepipeline.amaz
	onaws.com"
	        },
	        "Action":
	  "sts:AssumeRole"
	     }
	    ]
	}
	``
	4. Choose **Next**.
	5. In the list of available IAM policies, locate the policy you created for CodePipeline, and then choose **Next** to attach it to the role.
	6. Specify a name for the role, and choose **Create role** to save it for future reference in CodePipeline.	

Compile and bind the COBOL Db2 program

Task	Description	Skills required
Create a CodePipeline pipeline and CodeBuild project.	To create a CodePipeline pipeline and the CodeBuild project that compiles and binds the COBOL Db2 program: 1. Open the [CodePipeline console](), and choose **Create Pipeline**, **Build custom pipeline**.	AWS DevOps

Task	Description	Skills required
	2. Specify a name for the pipeline. 3. For **Service role**, choose **Existing service role** and choose specify the ARN for the IAM role you created for CodePipeline. 4. Expand **Advanced settings**, choose **Custom location**, choose the S3 bucket you previously created, and then choose **Next**. 5. For **Source provider**, select your third-party source provider, and provide the relevant information for the provider:. a. For **Connection,** select the connection that was created for the source provider. b. For **Repository name**, select your repository. c. For **Default branch**, select the branch that stores your COBOL program and `buildspec.yml` . d. Choose **Next**. 6. For **Build provider**, choose **Other build providers**, **AWS CodeBuild**.	

Task	Description	Skills required
	7. For **Project name**, choose **Create project**. The console displays a CodeBuild window where you can create a build project. In this window: a. Enter a name for the project. b. For **Environment image**, choose **Custom image**. c. For **Environment type**, choose **Linux Container**. d. For **ECR account**, choose **Other ECR account**. e. For **Amazon ECR repository URI**, enter: `673918848628.dkr.ecr.<your-region>.amazonaws.com/m2-enterprise-build-tools:8.0.9.R1` . f. For **Service role**, choose **Existing service role** and select the role you created for CodeBuild. g. Expand the **Additional configuration** section, and then choose the VPC, private subnets,	

Task	Description	Skills required
	and security group for this project. h. In the **Buildspec** section, choose **Use a buildspec file**. i. At the end of the window, choose **Continue to** CodePipeline. The CodeBuild window will close so you can go back to the CodePipeline console. 8. Back in the CodePipeline console, choose **Next**. 9. In the **Add deploy stage** pane, choose **Skip deploy stage**, and confirm. 10Review the pipeline parameters and then choose **Create pipeline**.	
Review the output.	Verify the success of the build by reviewing the CodePipeline build logs.	AWS DevOps

Task	Description	Skills required

Check results in Db2.	Verify the package version on the SYSPLAN table.
	`select CAST(NAME AS VARCHAR(10)) as name, VALIDATE, LAST_BIND_TIME, LASTUSED, CAST(PKGVERSION AS VARCHAR(10)) as PKGVERSION from SYSIBM.SYSPLAN where NAME = 'CDB2SMP' order by LAST_BIND_TIME desc`
	The version must match the CodeBuild build ID, which is CDB2SMP in our example:
	`NAME       VALIDATE LAST_BIND_TIME             LASTUSED PKGVERSION`
	`---------- -------- -------------------------- -------- ----------`
	`CDB2SMP    B        2024-05-18-11.53.11.503738 01/01/0001 19`	

Troubleshooting

Issue	Solution
Occasionally, the AWS console switches Regions when you move between services.	Make sure to verify the selected AWS Region whenever you switch between services.

Issue	Solution
	The AWS Region selector is in the upper-right corner of the console window.
It can be difficult to identify Db2 connectivity issues from CodeBuild.	To troubleshoot connectivity problems, add the following DB2 connect command to the `buildspec.yml` file. This addition helps you debug and resolve connectivity issues. `db2 connect to $DB_NAME user $DB2USER using $DB2PASS`
Occasionally, the role pane in the IAM console doesn't immediately show the IAM policy you've created.	If you encounter a delay, refresh the screen to display the latest information.

Related resources

IBM documentation

- IBM Data Server Client and driver types
- Download IBM Data Server Client and driver types

AWS documentation

- Amazon S3 User Guide
- AWS CodeBuild User Guide
- AWS Mainframe Modernization User Guide
- AWS Secrets Manager User Guide
- AWS CodePipeline User Guide
- AWS CodeDeploy User Guide

Additional information

CodeBuild policy

Replace the placeholders `<RegionID>`, `<AccountID>`, `<SubnetARN>`, `<BucketARN>`, and `<DB2CredSecretARN>` with your values.

```
{"Version": "2012-10-17",
    "Statement": [
        {"Action": "ecr:GetAuthorizationToken", "Effect": "Allow", "Resource": "*" },
        {"Action": ["ecr:GetDownloadUrlForLayer", "ecr:BatchGetImage",
                    "ecr:BatchCheckLayerAvailability"],
         "Effect": "Allow",
         "Resource": "arn:aws:ecr:*:673918848628:repository/m2-enterprise-build-
tools"},
        {"Action": "s3:PutObject", "Effect": "Allow", "Resource": "arn:aws:s3:::aws-m2-
repo-*/*"},
        {"Action": ["logs:PutLogEvents", "logs:CreateLogStream",
 "logs:CreateLogGroup"],
         "Effect": "Allow", "Resource": "arn:aws:logs:<RegionId>:<AccountId>:*"},
        {"Action": ["ec2:DescribeVpcs", "ec2:DescribeSubnets",
                    "ec2:DescribeSecurityGroups", "ec2:DescribeNetworkInterfaces",
                    "ec2:DescribeDhcpOptions", "ec2:DeleteNetworkInterface",
                    "ec2:CreateNetworkInterface"],
         "Effect": "Allow", "Resource": "*"},
        {"Action": "ec2:CreateNetworkInterfacePermission",
         "Effect": "Allow", "Resource": ["<SubnetARN>"]},
        {"Action": "s3:*", "Effect": "Allow", "Resource": ["<BucketARN>/
*","<BucketARN>"]},
        {"Action": "secretsmanager:GetSecretValue",
         "Effect": "Allow", "Resource": "<DB2CredSecretARN>"}
    ]
}
```

CodePipeline policy

Replace the placeholders `<BucketARN>` and `<ConnectionARN>` with your values.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {"Action": ["s3:List*", "s3:GetObjectVersion", "s3:GetObject",
 "s3:GetBucketVersioning" ],
        "Effect": "Allow",
        "Resource": ["<BucketARN>/*", "<BucketARN>"]},
        {"Action": ["codebuild:StartBuild", "codebuild:BatchGetBuilds"],
         "Effect": "Allow", "Resource": "*"},
```

```
        {"Action": ["codestar-connections:UseConnection"],
         "Effect": "Allow", "Resource": "<ConnectionARN>"}
        ]
}
```

buildspec.yml

Replace the `<your-bucket-name>` placeholder with your actual S3 bucket name.

```
version: 0.2
phases:
  pre_build:
    commands:
      - /var/microfocuslicensing/bin/mfcesd -no > /var/microfocuslicensing/logs/
mfcesd_startup.log 2>&1 &
      - |
        mkdir $CODEBUILD_SRC_DIR/db2client
        aws s3 cp s3://<your-bucket-name>/v11.5.8_linuxx64_client.tar.gz
 $CODEBUILD_SRC_DIR/db2client/ >> /dev/null 2>&1
        tar -xf $CODEBUILD_SRC_DIR/db2client/v11.5.8_linuxx64_client.tar.gz -C
 $CODEBUILD_SRC_DIR/db2client/
        cd $CODEBUILD_SRC_DIR/db2client/
        ./client/db2_install -f sysreq -y -b /opt/ibm/db2/V11.5 >> /dev/null 2>&1

        useradd db2cli
        /opt/ibm/db2/V11.5/instance/db2icrt -s client -u db2cli db2cli
        DB2CRED=$(aws secretsmanager get-secret-value --secret-id dev-db2-cred | jq -r
 '.SecretString | fromjson')
        read -r DB2USER DB2PASS DB_NODE DB_HOST DB_PORT DB_NAME DB_QUAL <<<$(echo
 $DB2CRED | jq -r
 '.username, .password, .db2node, .db2host, .db2port, .db2name, .qualifier')
        . /home/db2cli/sqllib/db2profile
        db2 catalog tcpip node $DB_NODE remote $DB_HOST server $DB_PORT
        db2 catalog db $DB_NAME as $DB_NAME at node $DB_NODE authentication server
  build:
    commands:
      - |
        revision=$CODEBUILD_SRC_DIR/loadlib
        mkdir -p $revision; cd $revision
        . /opt/microfocus/EnterpriseDeveloper/bin/cobsetenv
        cob -zU $CODEBUILD_SRC_DIR/CDB2SMP.cbl -C "DB2(DB==${DB_NAME} PASS==${DB2USER}.
${DB2PASS} VERSION==${CODEBUILD_BUILD_NUMBER} COLLECTION==DB2AWSDB"
artifacts:
  files:
```

```
    - "**/*"
  base-directory: $revision
```

Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager

Kevin Yung and Krithika Palani Selvam, Amazon Web Services

Peter Woods, None

Abraham Rondon, Micro Focus

Summary

This pattern introduces a scalable architecture for mainframe applications using Micro Focus Enterprise Server in Scale-Out Performance and Availability Cluster (PAC) and an Amazon Elastic Compute Cloud (Amazon EC2) Auto Scaling group on Amazon Web Services (AWS). The solution is fully automated with AWS Systems Manager and Amazon EC2 Auto Scaling lifecycle hooks. By using this pattern, you can set up your mainframe online and batch applications to achieve high resiliency by automatically scaling in and out based on your capacity demands.

Note

This pattern was tested with Micro Focus Enterprise Server version 6.0. For version 8, see Set up Micro Focus Runtime (on Amazon EC2).

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Micro Focus Enterprise Server software and license. For details, contact Micro Focus sales.
- An understanding of the concept of rebuilding and delivering a mainframe application to run in Micro Focus Enterprise Server. For a high-level overview, see Micro Focus Enterprise Server Data Sheet.
- An understanding of the concepts in Micro Focus Enterprise Server scale-out Performance and Availability Cluster. For more information, see the Micro Focus Enterprise Server documentation.
- An understanding of the overall concept of mainframe application DevOps with continuous integration (CI). For an AWS Prescriptive Guidance pattern that was developed by AWSand Micro Focus, see Mainframe modernization: DevOps on AWS with Micro Focus.

> **Note**
>
> This pattern was tested with Micro Focus Enterprise Server version 6. For version 8, see Set up Micro Focus Runtime (on Amazon EC2).

Limitations

- For a list of platforms that are supported by Micro Focus Enterprise Server, see the Micro Focus Enterprise Server Data Sheet.
- The scripts and tests used in this pattern are based on Amazon EC2 Windows Server 2019; other Windows Server versions and operating systems were not tested for this pattern.
- The pattern is based on Micro Focus Enterprise Server 6.0 for Windows; earlier or later releases were not tested in the development of this pattern.

Product versions

- Micro Focus Enterprise Server 6.0
- Windows Server 2019

Architecture

In the conventional mainframe environment, you must provision hardware to host your applications and corporate data. To cater for and meet the spikes in seasonal, monthly, quarterly, or even unprecedented or unexpected demands, mainframe users must *scale out* by purchasing additional storage and compute capacity. Increasing the number of storage and compute capacity resources improves overall performance, but the scaling is not linear.

This is not the case when you start adopting an on-demand consumption model on AWS by using Amazon EC2 Auto Scaling and Micro Focus Enterprise Servers. The following sections explain detail how to build a fully automated, scalable mainframe application architecture using Micro Focus Enterprise Server Scale-Out Performance and Availability Cluster (PAC) with an Amazon EC2 Auto Scaling group.

Micro Focus Enterprise Server automatic scaling architecture

First, it is important to understand the basic concepts of Micro Focus Enterprise Server. This environment provides a mainframe-compatible, x86 deployment environment for applications that have traditionally run on the IBM mainframe. It delivers both online and batch runs and a transaction environment that supports the following:

- IBM COBOL
- IBM PL/I
- IBM JCL batch jobs
- IBM CICS and IMS TM transactions
- Web services
- Common batch utilities, including SORT

Micro Focus Enterprise Server enables mainframe applications to run with minimal changes. Existing mainframe workloads can be moved to x86 platforms and modernized to take advantage of AWS Cloud native extensions for rapid expansion to new markets or geographies.

The AWS Prescriptive Guidance pattern Mainframe modernization: DevOps on AWS with Micro Focus introduced the architecture to accelerate the development and testing of mainframe applications on AWS using Micro Focus Enterprise Developer and Enterprise Test Server with AWS CodePipeline and AWS CodeBuild. This pattern focuses on the deployment of mainframe applications to the AWS production environment to achieve high availability and resiliency.

In a mainframe production environment, you might have set up IBM Parallel Sysplex in the mainframe to achieve high performance and high availability. To create a scale-out architecture similar to Sysplex, Micro Focus introduced the Performance and Availability Cluster (PAC) to Enterprise Server. PACs support mainframe application deployment onto multiple Enterprise Server regions managed as a single image and scaled out in Amazon EC2 instances. PACs also support predictable application performance and system throughput on demand.

In a PAC, multiple Enterprise Server instances work together as a single logical entity. Failure of one Enterprise Server instance, therefore, will not interrupt business continuity because capacity is shared with other regions while new instances are automatically started using industry standard functionality such as an Amazon EC2 Auto Scaling group. This removes single points of failure, improving resilience to hardware, network, and application issues. Scaled-out Enterprise Server instances can be operated and managed by using the Enterprise Server Common Web Administration (ESCWA) APIs, simplifying the operational maintenance and serviceability of Enterprise Servers.

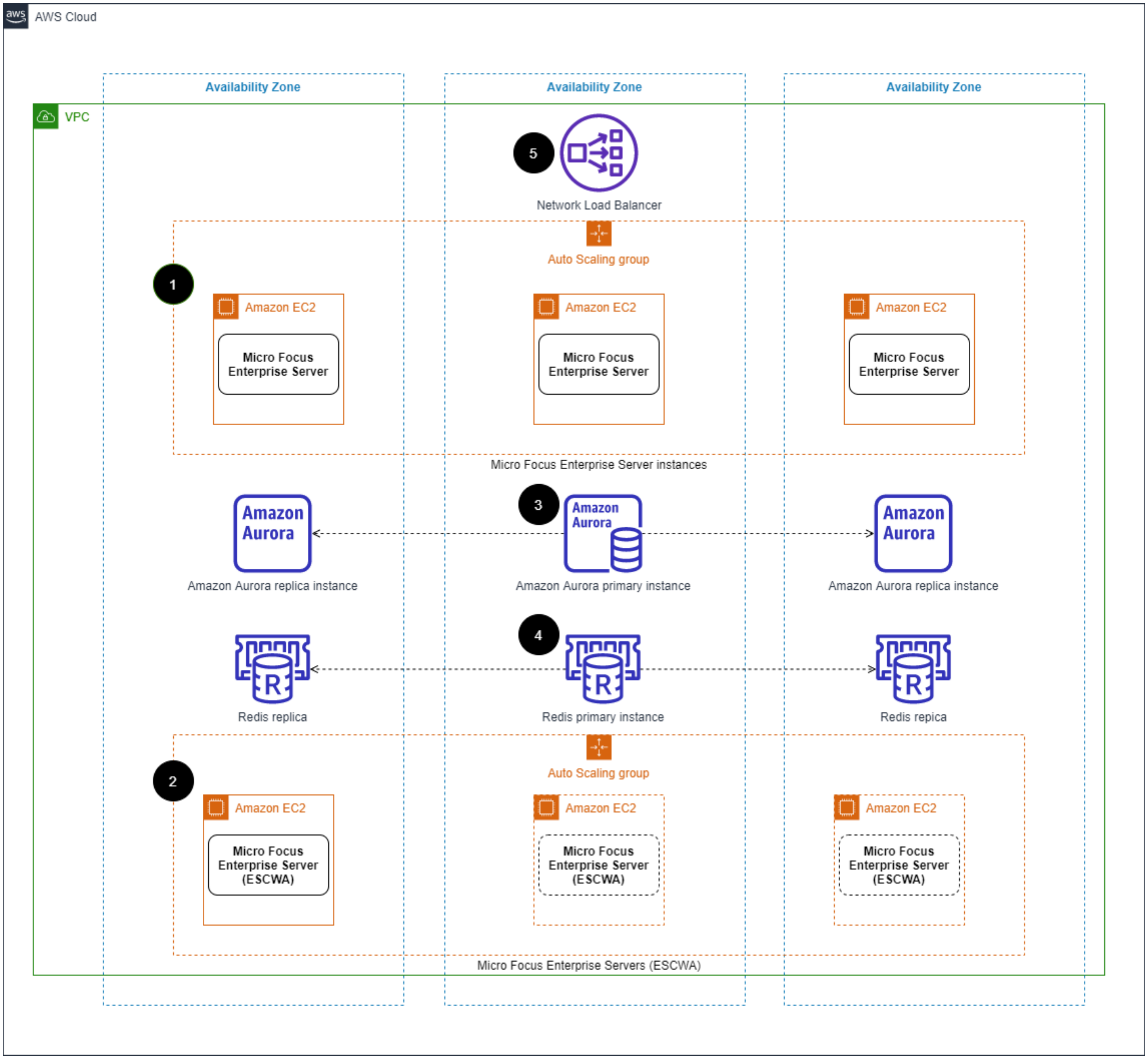
> **Note**
>
> Micro Focus recommends that the Performance and Availability Cluster (PAC) should consist of at least three Enterprise Server regions so that availability is not compromised in the event an Enterprise Server region fails or requires maintenance.

PAC configuration requires a supported relational database management service (RDBMS) to manage the region database, a cross-region database, and optional data store databases. A data store database should be used to managed Virtual Storage Access Method (VSAM) files using the Micro Focus Database File Handler support to improve availability and scalability. Supported RDBMSs include the following:

- Microsoft SQL Server 2009 R2 and later
- PostgreSQL 10.x, including Amazon Aurora PostgreSQL-Compatible Edition
- DB2 10.4 and later

For details of supported RDBMS and PAC requirements, see Micro Focus Enterprise Server - Prerequisites and Micro Focus Enterprise Server - Recommended PAC Configuration.

The following diagram shows a typical AWS architecture setup for a Micro Focus PAC.



	Component	Description
1	Enterprise Server instances automatic scaling group	Set up an automatic scaling group deployed with Enterprise Server instances in a PAC. The number of instances can be scaled out

		or in initiated by Amazon CloudWatch alarms using CloudWatch metrics.
2	Enterprise Server ESCWA instances automatic scaling group	Set up an automatic scaling group deployed with Enterprise Server Common Web Administration (ESCWA). ESCWA provides cluster management APIs. The ESCWA servers act as a control plane to add or remove Enterprise Servers and start or stop Enterpris e Server regions in the PAC during the Enterprise Server instance automatic scaling events. Because the ESCWA instance is used only for the PAC management, its traffic pattern is predictable, and its automatic scaling desired capacity requirement can be set to 1.
3	Amazon Aurora instance in a Multi-AZ setup	Set up a relational database management system (RDBMS) to host both user and system data files to be shared across the Enterprise Server instances.

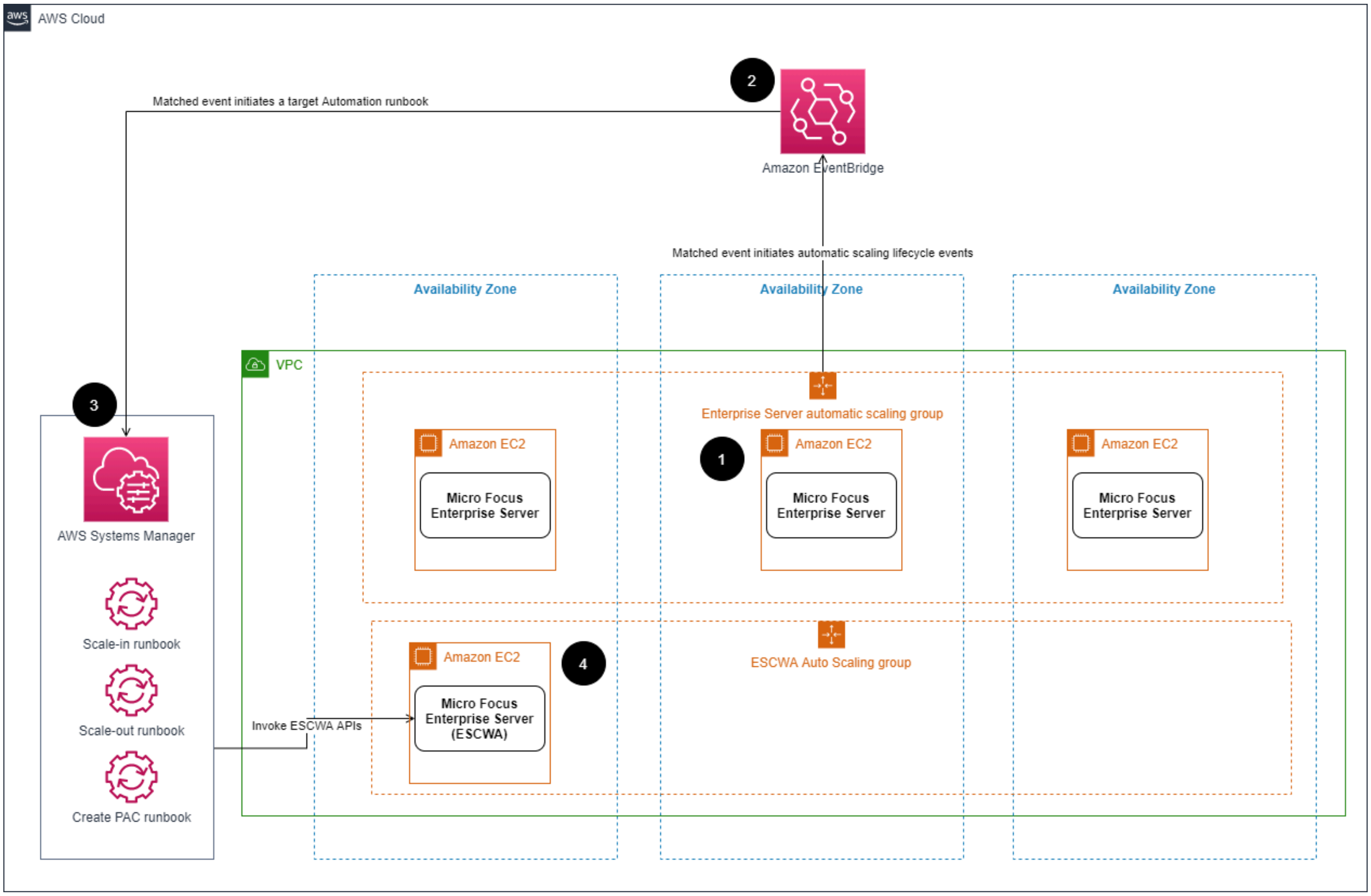
4	Amazon ElastiCache (Redis OSS) instance and replica	Set up an ElastiCache (Redis OSS) primary instance and at least one replica to host user data and act as a scale-out repository (SOR) for the Enterprise Server instances . You can configure one or more scale-out repository to store specific types of user data. Enterprise Server uses a Redis NoSQL database as an SOR, a requirement to maintain PAC integrity.
5	Network Load Balancer	Set up a load balancer, providing a hostname for applications to connect to the services provided by Enterprise Server instances (for example, accessing the application through a 3270 emulator).

These components form the minimum requirement for a Micro Focus Enterprise Server PAC cluster. The next section covers cluster management automation.

Using AWS Systems Manager Automation for scaling

After the PAC cluster is deployed on AWS, the PAC is managed through the Enterprise Server Common Web Administration (ESCWA) APIs.

To automate the cluster management tasks during automatic scaling events, you can use Systems Manager Automation runbooks and Amazon EC2 Auto Scaling with Amazon EventBridge. The architecture of these automations is shown in the following diagram.



	Component	Description
1	Automatic scaling lifecycle hook	Set up automatic scaling lifecycle hooks and send notifications to Amazon EventBridge when new instances are launched and existing instances are terminated in the automatic scaling group.
2	Amazon EventBridge	Set up an Amazon EventBrid ge rule to route automatic scaling events to Systems

		Manager Automation runbook targets.
3	Automation runbooks	Set up Systems Manager Automation runbooks to run Windows PowerShell scripts and invoke ESCWA APIs to manage the PAC. For examples, see the *Additional information* section.
4	Enterprise Server ESCWA instance in an automatic scaling group	Set up an Enterprise Server ESCWA instance in an automatic scaling group. The ESCWA instance provides APIs to manage the PAC.

Tools

- Micro Focus Enterprise Server – Micro Focus Enterprise Server provides the run environment for applications created with any integrated development environment (IDE) variant of Enterprise Developer.
- Amazon EC2 Auto Scaling – Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called Auto Scaling groups, and specify minimum and maximum numbers of instances.
- Amazon ElastiCache (Redis OSS) – Amazon ElastiCache is a web service for setting up, managing, and scaling a distributed in-memory data store or cache environment in the cloud. It provides a high-performance, scalable, and cost-effective caching solution.
- Amazon RDS – Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for a relational database and manages common database administration tasks.
- AWS Systems Manager – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS

resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting on (or taking corrective action on) any policy violations it detects.

Epics

Create an Amazon Aurora instance

Task	Description	Skills required
Create an AWS CloudFormation template for an Amazon Aurora instance.	Use the AWS example code snippet to make a CloudFormation template that will create an Amazon Aurora PostgreSQL-Compatible Edition instance.	Cloud architect
Deploy a CloudFormation stack to create the Amazon Aurora instance.	Use the CloudFormation template to create an Aurora PostgreSQL-Compatible instance that has Multi-AZ replication enabled for production workloads.	Cloud architect
Configure database connection settings for Enterprise Server.	Follow the instructions in the Micro Focus documentation to prepare the connection strings and database configuration for Micro Focus Enterprise Server.	Data engineer, DevOps engineer

Create an Amazon ElastiCache cluster for the Redis instance

Task	Description	Skills required
Create a CloudFormation template for the Amazon	Use the AWS example code snippet to make a CloudForm	Cloud architect

Task	Description	Skills required
ElastiCache cluster for the Redis instance.	ation template that will create an Amazon ElastiCache cluster for the Redis instance.	
Deploy the CloudFormation stack to create an Amazon ElastiCache cluster for the Redis instance.	Create the Amazon ElastiCache cluster for the Redis instance that has Multi-AZ replication enabled for production workloads.	Cloud architect
Configure Enterprise Server PSOR connection settings.	Follow the instructions in the [Micro Focus documentation](#) to prepare the PAC Scale-Out Repository (PSOR) connection configuration for Micro Focus Enterprise Server PAC.	DevOps engineer

Create a Micro Focus Enterprise Server ESCWA automatic scaling group

Task	Description	Skills required
Create a Micro Focus Enterprise Server AMI.	Create an Amazon EC2 Windows Server instance and install the Micro Focus Enterprise Server binary in the EC2 instance. Create an Amazon Machine Image (AMI) of the EC2 instance. For more information, see the [Enterprise Server installation documentation](#).	Cloud architect
Create a CloudFormation template for Enterprise Server ESCWA.	Use the [AWS example code snippet](#) to make a template for creating a custom stack of	Cloud architect

Task	Description	Skills required
	Enterprise Server ESCWA in an automatic scaling group.	
Deploy the CloudForm ation stack to create an Amazon EC2 scaling group for Enterprise Server ESCWA.	Use the CloudFormation template to deploy the automatic scaling group with the Micro Focus Enterprise Server ESCWA AMI created in the previous story.	Cloud architect

Create an AWS Systems Manager Automation runbook

Task	Description	Skills required
Create a CloudFormation template for a Systems Manager Automation runbook.	Use the example code snippets in the *Additional information* section to make a CloudFormation template that will create a Systems Manager Automation runbook for automating PAC creation, Enterprise Server scale in, and Enterprise Server scale out.	Cloud architect
Deploy the CloudFormation stack that contains the Systems Manager Automation runbook.	Use the CloudFormation template to deploy a stack that contains the Automatio n runbook for PAC creation, Enterprise Server scale in, and Enterprise Server scale out.	Cloud architect

Create an automatic scaling group for Micro Focus Enterprise Server

Task	Description	Skills required
Create a CloudFormation template for setting up an automatic scaling group for Micro Focus Enterprise Server.	Use the AWS example code snippet to make a CloudFormation template that will create an automatic scaling group. This template will reuse the same AMI that was created for the Micro Focus Enterprise Server ESCWA instance. Then use an AWS example code snippet to create the automatic scaling lifecycle event and set up Amazon EventBridge to filter for scale-out and scale-in events in the same CloudFormation template.	Cloud architect
Deploy the CloudFormation stack for the automatic scaling group for Micro Focus Enterprise Servers.	Deploy the CloudFormation stack that contains the automatic scaling group for Micro Focus Enterprise Servers.	Cloud architect

Related resources

- Micro Focus Enterprise Server Performance and Availability Cluster (PAC)
- Amazon EC2 Auto Scaling lifecycle hooks
- Running automations with triggers using EventBridge

Additional information

The following scenarios must be automated for scaling in or scaling out the PAC clusters.

Automation for starting or recreating a PAC

At the start of a PAC cluster, Enterprise Server requires ESCWA to invoke APIs to create a PAC configuration. This starts and adds Enterprise Server regions into the PAC. To create or recreate a PAC, use the following steps:

1. Configure a PAC Scale-Out Repository (PSOR) in ESCWA with a given name.

```
POST /server/v1/config/groups/sors
```

2. Create a PAC with a given name and attach the PSOR to it.

```
POST /server/v1/config/groups/pacs
```

3. Configure the region database and cross-region database if this is the first time you are setting up a PAC.

> **Note**
>
> This step uses SQL queries and the Micro Focus Enterprise Suite command line **dbhfhadmin** tool to create the database and import initial data.

4. Install the PAC definition into the Enterprise Server regions.

```
POST /server/v1/config/mfds
POST /native/v1/config/groups/pacs/${pac_uid}/install
```

5. Start Enterprise Server regions in the PAC.

```
POST /native/v1/regions/${host_ip}/${port}/${region_name}/start
```

The previous steps can be implemented by using a Windows PowerShell script.

The following steps explain how to build an automation for creating a PAC by reusing the Windows PowerShell script.

1. Create an Amazon EC2 launch template that downloads or creates the Windows PowerShell script as part of the bootstrap process. For example, you can use EC2 user data to download the script from an Amazon Simple Storage Service (Amazon S3) bucket.
2. Create an AWS Systems Manager Automation runbook to invoke the Windows PowerShell script.
3. Associate the runbook to the ESCWA instance by using the instance tag.
4. Create an ESCWA automatic scaling group by using the launch template.

You can use the following example AWS CloudFormation snippet to create the Automation runbook.

Example CloudFormation snippet for a Systems Manager Automation runbook used for PAC creation

```
PACInitDocument:
   Type: AWS::SSM::Document
   Properties:
     DocumentType: Command
     Content:
       schemaVersion: '2.2'
       description: Operation Runbook to create Enterprise Server PAC
       mainSteps:
       - action: aws:runPowerShellScript
         name: CreatePAC
         inputs:
           onFailure: Abort
           timeoutSeconds: "1200"
           runCommand:
           - |
             C:\Scripts\PAC-Init.ps1
PacInitAutomation:
   Type: AWS::SSM::Document
   Properties:
     DocumentType: Automation
     Content:
       description: Prepare Micro Focus PAC Cluster via ESCWA Server
       schemaVersion: '0.3'
       assumeRole: !GetAtt SsmAssumeRole.Arn
       mainSteps:
         - name: RunPACInitDocument
           action: aws:runCommand
           timeoutSeconds: 300
           onFailure: Abort
```

```
            inputs:
              DocumentName: !Ref PACInitDocument
              Targets:
                - Key: tag:Enterprise Server - ESCWA
                  Values:
                    - "true"
  PacInitDocumentAssociation:
    Type: AWS::SSM::Association
    Properties:
      DocumentVersion: "$LATEST"
      Name: !Ref PACInitDocument
      Targets:
        - Key: tag:Enterprise Server - ESCWA
          Values:
            - "true"
```

For more information, see [Micro Focus Enterprise Server - Configuring a PAC](#).

Automation for scaling out with a new Enterprise Server instance

When an Enterprise Server instance is scaled out, its Enterprise Server region must be added to the PAC. The following steps explain how to invoke ESCWA APIs and add the Enterprise Server region into the PAC.

1. Install the PAC definition into the Enterprise Server regions.

```
POST '/server/v1/config/mfds'
POST /native/v1/config/groups/pacs/${pac_uid}/install
```

2. Warm Start the region in the PAC.

```
POST /native/v1/regions/${host_ip}/${port}/${region_name}/start
```

3. Add the Enterprise Server instance to the load balancer by associating the automatic scaling group to the load balancer.

The previous steps can be implemented by using a Windows PowerShell script. For more information, see [Micro Focus Enterprise Server - Configuring a PAC](#).

The following steps can be used to build an event driven automation to add a newly launched Enterprise Server instance into a PAC by reusing the Windows PowerShell script.

1. Create an Amazon EC2 launch template for Enterprise Server instance that provisions an Enterprise Server Region during its bootstrap. For example, you can use the Micro Focus Enterprise Server command mfds to import a region configuration. For further details and options available for this command, see the Enterprise Server Reference.
2. Create an Enterprise Server automatic scaling group that uses the launch template created in the previous step.
3. Create a Systems Manager Automation runbook to invoke the Windows PowerShell script.
4. Associate the runbook to the ESCWA instance by using the instance tag.
5. Create an Amazon EventBridge rule to filter for the EC2 Instance Launch Successful event for the Enterprise Server automatic scaling group, and create the target to use the Automation runbook.

You can use the following example CloudFormation snippet to create the Automation runbook and the EventBridge rule.

Example CloudFormation snippet for Systems Manager used for scaling out Enterprise Server instances

```
ScaleOutDocument:
   Type: AWS::SSM::Document
   Properties:
     DocumentType: Command
     Content:
       schemaVersion: '2.2'
       description: Operation Runbook to Adding MFDS Server into an existing PAC
       parameters:
         MfdsPort:
           type: String
         InstanceIpAddress:
           type: String
           default: "Not-Available"
         InstanceId:
           type: String
           default: "Not-Available"
       mainSteps:
       - action: aws:runPowerShellScript
         name: Add_MFDS
         inputs:
           onFailure: Abort
           timeoutSeconds: "300"
```

```
            runCommand:
            - |
              $ip = "{{InstanceIpAddress}}"
              if ( ${ip} -eq "Not-Available" ) {
                $ip = aws ec2 describe-instances --instance-id {{InstanceId}} --output
 text --query "Reservations[0].Instances[0].PrivateIpAddress"
              }
              C:\Scripts\Scale-Out.ps1 -host_ip ${ip} -port {{MfdsPort}}

  PacScaleOutAutomation:
    Type: AWS::SSM::Document
    Properties:
      DocumentType: Automation
      Content:
        parameters:
          MfdsPort:
            type: String
          InstanceIpAddress:
            type: String
            default: "Not-Available"
          InstanceId:
            type: String
            default: "Not-Available"
        description: Scale Out 1 New Server in Micro Focus PAC Cluster via ESCWA
 Server
        schemaVersion: '0.3'
        assumeRole: !GetAtt SsmAssumeRole.Arn
        mainSteps:
          - name: RunScaleOutCommand
            action: aws:runCommand
            timeoutSeconds: 300
            onFailure: Abort
            inputs:
              DocumentName: !Ref ScaleOutDocument
              Parameters:
                InstanceIpAddress: "{{InstanceIpAddress}}"
                InstanceId: "{{InstanceId}}"
                MfdsPort: "{{MfdsPort}}"
              Targets:
                - Key: tag:Enterprise Server - ESCWA
                  Values:
                    - "true"
```

Automation for scaling in an Enterprise Server instance

Similar to scaling out, when an Enterprise Server instance is *scaled in*, the event EC2 Instance-terminate Lifecycle Action is initiated, and the following process and API calls are needed to remove a Micro Focus Enterprise Server instance from the PAC.

1. Stop the region in the terminating Enterprise Server instance.

```
POST "/native/v1/regions/${host_ip}/${port}/${region_name}/stop"
```

2. Remove the Enterprise Server Instance from the PAC.

```
DELETE "/server/v1/config/mfds/${uid}"
```

3. Send signal to continue terminating the Enterprise Server instance.

The previous steps can be implemented in a Windows PowerShell script. For additional details of this process, see Micro Focus Enterprise Server document - Administering a PAC.

The following steps explain how to build an event-driven automation to terminate an Enterprise Server instance from a PAC by reusing the Windows PowerShell script.

1. Create a Systems Manager Automation runbook to invoke the Windows PowerShell script.
2. Associate the runbook to the ESCWA instance by using the instance tag.
3. Create an automatic scaling group lifecycle hook for EC2 instance termination.
4. Create an Amazon EventBridge rule to filter EC2 Instance-terminate Lifecycle Action event for the Enterprise Server automatic scaling group, and create the target to use the Automation runbook.

You can use the following example CloudFormation template for creating a Systems Manager Automation runbook, lifecycle hook, and EventBridge rule.

Example CloudFormation snippet for a Systems Manager Automation runbook used for scaling in an Enterprise Server instance

```
ScaleInDocument:
   Type: AWS::SSM::Document
   Properties:
     DocumentType: Command
```

```
      Content:
        schemaVersion: '2.2'
        description: Operation Runbook to Remove MFDS Server from PAC
        parameters:
          MfdsPort:
            type: String
          InstanceIpAddress:
            type: String
            default: "Not-Available"
          InstanceId:
            type: String
            default: "Not-Available"
        mainSteps:
        - action: aws:runPowerShellScript
          name: Remove_MFDS
          inputs:
            onFailure: Abort
            runCommand:
            - |
              $ip = "{{InstanceIpAddress}}"
              if ( ${ip} -eq "Not-Available" ) {
                $ip = aws ec2 describe-instances --instance-id {{InstanceId}} --output
 text --query "Reservations[0].Instances[0].PrivateIpAddress"
              }
              C:\Scripts\Scale-In.ps1 -host_ip ${ip} -port {{MfdsPort}}

  PacScaleInAutomation:
    Type: AWS::SSM::Document
    Properties:
      DocumentType: Automation
      Content:
        parameters:
          MfdsPort:
            type: String
          InstanceIpAddress:
            type: String
            default: "Not-Available"
          InstanceId:
            type: String
            default: "Not-Available"
        description: Scale In 1 New Server in Micro Focus PAC Cluster via ESCWA Server
        schemaVersion: '0.3'
        assumeRole: !GetAtt SsmAssumeRole.Arn
        mainSteps:
```

```
            - name: RunScaleInCommand
              action: aws:runCommand
              timeoutSeconds: "600"
              onFailure: Abort
              inputs:
                DocumentName: !Ref ScaleInDocument
                Parameters:
                  InstanceIpAddress: "{{InstanceIpAddress}}"
                  MfdsPort: "{{MfdsPort}}"
                  InstanceId: "{{InstanceId}}"
                Targets:
                  - Key: tag:Enterprise Server - ESCWA
                    Values:
                      - "true"
            - name: TerminateTheInstance
              action: aws:executeAwsApi
              inputs:
                Service: autoscaling
                Api: CompleteLifecycleAction
                AutoScalingGroupName: !Ref AutoScalingGroup
                InstanceId: "{{ InstanceId }}"
                LifecycleActionResult: CONTINUE
                LifecycleHookName: !Ref ScaleInLifeCycleHook
```

Automation for an Amazon EC2 automatic scaling trigger

The process of setting up a scaling policy for Enterprise Server instances requires an understanding of the application behavior. In most cases, you can set up target tracking scaling policies. For example, you can use the average CPU utilization as the Amazon CloudWatch metric to set for the automatic scaling policy. For more information, see Target tracking scaling policies for Amazon EC2 Auto Scaling. For applications that have regular traffic patterns, consider using a predictive scaling policy. For more information, see Predictive scaling for Amazon EC2 Auto Scaling.

Build an advanced mainframe file viewer in the AWS Cloud

Boopathy GOPALSAMY and Jeremiah O'Connor, Amazon Web Services

Summary

This pattern provides code samples and steps to help you build an advanced tool for browsing and reviewing your mainframe fixed-format files by using AWS serverless services. The pattern provides an example of how to convert a mainframe input file to an Amazon OpenSearch Service document for browsing and searching. The file viewer tool can help you achieve the following:

- Retain the same mainframe file structure and layout for consistency in your AWS target migration environment (for example, you can maintain the same layout for files in a batch application that transmits files to external parties)
- Speed up development and testing during your mainframe migration
- Support maintenance activities after the migration

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC) with a subnet that's reachable by your legacy platform
- > **Note**
  >
  > An input file and its corresponding common business-oriented language (COBOL) copybook (: For input file and COBOL copybook examples, see gfs-mainframe-solutions on the GitHub repository. For more information about COBOL copybooks, see the Enterprise COBOL for z/OS 6.3 Programming Guide on the IBM website.)

Limitations

- Copybook parsing is limited to no more than two nested levels (OCCURS)

Architecture

Source technology stack

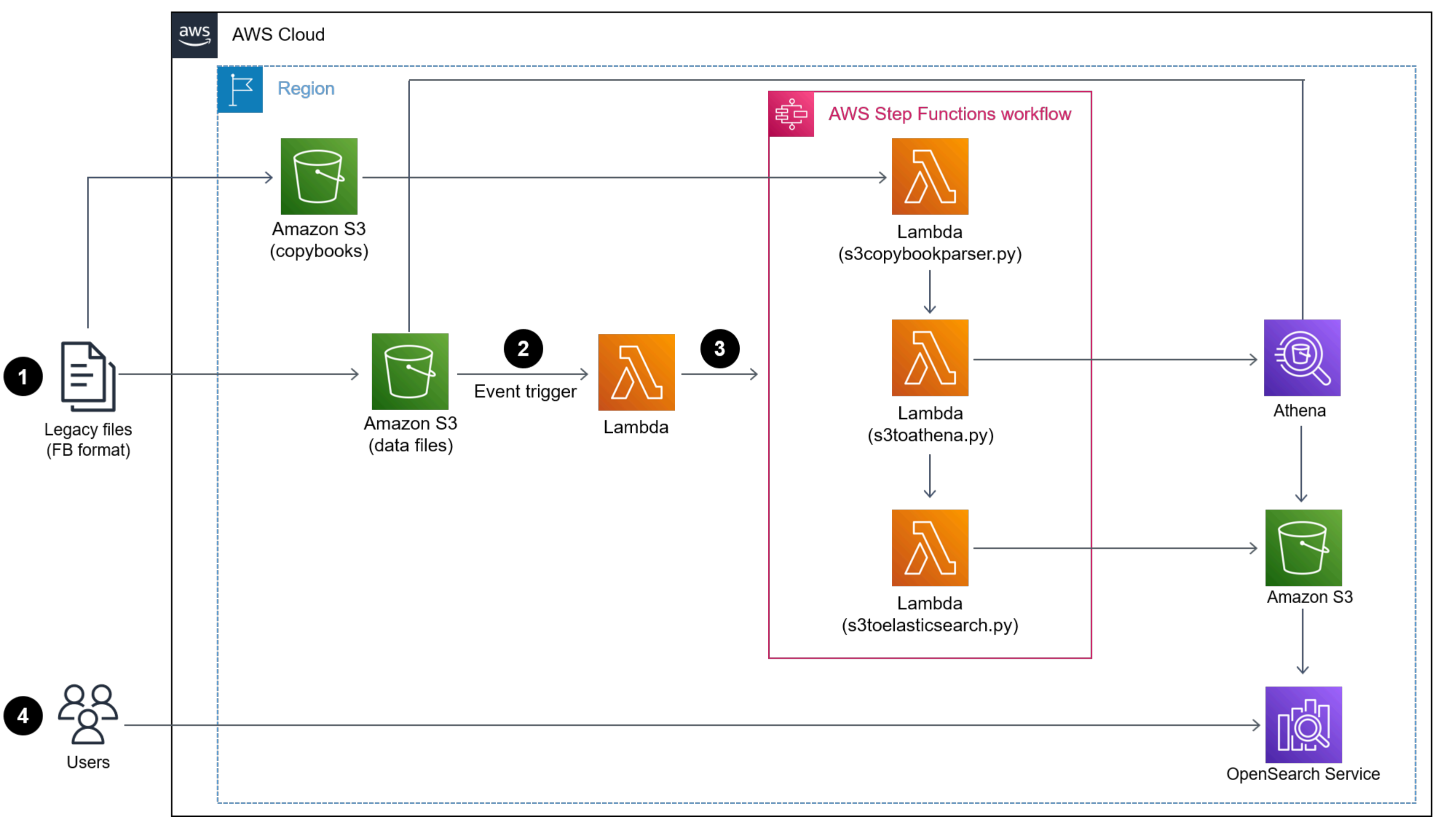
- Input files in FB (Fixed Blocked) format
- COBOL copybook layout

Target technology stack

- Amazon Athena
- Amazon OpenSearch Service
- Amazon Simple Storage Service (Amazon S3)
- AWS Lambda
- AWS Step Functions

Target architecture

The following diagram shows the process of parsing and converting a mainframe input file to an OpenSearch Service document for browsing and searching.



The diagram shows the following workflow:

1. An admin user or application pushes input files to one S3 bucket and COBOL copybooks to another S3 bucket.
2. > **Note**
   >
   > The S3 bucket with the input files invokes a Lambda function that kicks off a serverless Step Functions workflow. : The use of an S3 event trigger and Lambda function to drive the Step Functions workflow in this pattern is optional. The GitHub code samples in this pattern don't include the use of these services, but you can use these services based on your requirements.
3. The Step Functions workflow coordinates all the batch processes from the following Lambda functions:
   - The `s3copybookparser.py` function parses the copybook layout and extracts field attributes, data types, and offsets (required for input data processing).
   - The `s3toathena.py` function creates an Athena table layout. Athena parses the input data that's processed by the `s3toathena.py` function and converts the data to a CSV file.
   - The `s3toelasticsearch.py` function ingests the results file from the S3 bucket and pushes the file to OpenSearch Service.
4. Users access OpenSearch Dashboards with OpenSearch Service to retrieve the data in various table and column formats and then run queries against the indexed data.

Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use. In this pattern, you use Lambda to implement core logic, such as parsing files, converting data, and loading data into OpenSearch Service for interactive file access.
- Amazon OpenSearch Service is a managed service that helps you deploy, operate, and scale OpenSearch Service clusters in the AWS Cloud. In this pattern, you use OpenSearch Service to index the converted files and provide interactive search capabilities for users.

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Step Functions is a serverless orchestration service that helps you combine Lambda functions and other AWS services to build business-critical applications. In this pattern, you use Step Functions to orchestrate Lambda functions.

Other tools

- GitHub is a code-hosting service that provides collaboration tools and version control.
- Python is a high-level programming language.

Code

The code for this pattern is available in the GitHub gfs-mainframe-patterns repository.

Epics

Prepare the target environment

Task	Description	Skills required
Create the S3 bucket.	Create an S3 bucket for storing the copybooks, input files, and output files. We recommend the following folder structure for your S3 bucket: • `copybook/` • `input/` • `output/` • `query/`	General AWS

Task	Description	Skills required
	• `results/`	
Create the s3copybookparser function.	1. Create a Lambda function called `s3copybookparser` and upload the source code (`s3copybookparser.py` and `copybook.py`) from the GitHub repository. 2. Attach the IAM policy `S3ReadOnly` to the Lambda function.	General AWS
Create the s3toathena function.	1. Create a Lambda function called `s3toathena` and upload the source code (`s3toathena.py`) from the GitHub repository. Configure the Lambda timeout to > 60 seconds. 2. To provide access to the required resources, attach the IAM policies `AmazonAthenaFullAccess` and `S3FullAccess` to the Lambda function.	General AWS

Task	Description	Skills required
Create the s3toelasticsearch function.	1. **Important** [Add a Python dependency to your Lambda environment](). : To use the `s3toelasticsearch` function, you must add the Python dependency because the Lambda function uses Python Elasticsearch client dependencies (`Elasticsearch==7.9.0` and `requests_aws4auth`). 2. Create a Lambda function called `s3toelasticsearch` and upload the source code (`s3toelasticsearch.py`) from the [GitHub]() repository. 3. Import the Python dependency as a Lambda layer. 4. Attach the IAM policies `S3ReadOnly` and `AmazonOpenSearchSe`	General AWS

Task	Description	Skills required
	`rviceReadOnlyAccess` to the Lambda function.	

Task	Description	Skills required
Create the OpenSearch Service cluster.	**Create the cluster** 1. Create an OpenSearch Service cluster. When you create the cluster, do the following: • **Note** Create a master user and password for the cluster that you can use for signing in to OpenSearch Dashboards. : This step is not required if you use authentication through Amazon Cognito. • Choose fine-grained access control. This give you additional ways of controlling access to your data in OpenSearch Service. 2. Copy the domain URL and pass it as the environment variable **'HOST'** to the Lambda function `s3toelasticsearch` .	General AWS

Task	Description	Skills required
	Grant access to the IAM role To provide fine-grained access to the Lambda function's IAM role (`arn:aws:iam::**:role/service-role/s3toelasticsearch-role-**`), do the following: 1. Sign in to OpenSearch Dashboards as the master user. 2. Choose the **Security** tab, and then choose **Roles**, **all_access**, **Map user**, **Backend roles**. 3. Add the Amazon Resource Name (ARN) of the Lambda function's IAM role, and then choose **Save**. For more information, see [Mapping roles to users](#) in the OpenSearch Service documentation.	

Task	Description	Skills required
Create Step Functions for orchestration.	1. [Create a Step Functions state machine](#) with the standard flow. The definitio n is included in the [GitHub](#) repository. 2. In the JSON script, replace the Lambda function’s ARNs with the ARNs from the Lambda function in your environment.	General AWS

Deploy and run

Task	Description	Skills required
Upload the input files and copybooks to the S3 bucket.	Download sample files from the [GitHub](#) repository sample folder and upload the files to the S3 bucket that you created earlier. 1. Upload `Mockedcopy.cpy` and `acctix.cpy` to the `<S3_Bucket>/copybook` folder. 2. Upload the `Modedupdate.txt` and `acctindex.cpy` sample input files to the `<S3_Bucket>/input` folder.	General AWS
Invoke the Step Functions.	1. Sign in to the AWS Management Console and	General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

open the [Step Functions console](#).

2. In the navigation pane, choose **State machines**.
3. Choose your state machine, and then choose **Start execution**.
4. In the **Input** box, enter the following copybook/file path as a JSON variable to the S3 bucket, and then choose **Start execution**.

```
{
  "s3_copybook_bucket_name": "<BUCKET NAME>",
  "s3_copybook_bucket_key": "<COPYBOOK PATH>",
  "s3_source_bucket_name": "<BUCKET NAME",
  "s3_source_bucket_key": "INPUT FILE PATH"
}
```

For example:

```
{
  "s3_copybook_bucket_name": "fileaidtest",
  "s3_copybook_bucket_key": "copybook/acctix.cpy",
  "s3_source_bucket_name": "fileaidtest",
```

</td><td></td></tr>
</table>

Task	Description	Skills required
	``` "s3_source_bucket_key": "input/acctindex" } ```	
Validate the workflow execution in Step Functions.	In the [Step Functions console](), review the workflow execution in the **Graph inspector**. The execution run states are color coded to represent execution status. For example, blue indicates **In Progress**, green indicates **Succeeded**, and red indicates **Failed**. You can also review the table in the **Execution event history** section for more detailed information about the execution events. For an example of a graphical workflow execution, see *Step Functions graph* in the *Additional information* section of this pattern.	General AWS

Task	Description	Skills required
Validate the delivery logs in Amazon CloudWatch.	1. Sign in to the AWS Management Console and open the [CloudWatch console](). 2. In the navigation pane, expand **Logs**, and then choose **Log groups**. 3. In the search box, search for the `s3toelasticsearch` function's log group. For an example of successful delivery logs, see *CloudWatch delivery logs* in the *Additional information* section of this pattern.	General AWS

Task	Description	Skills required
Validate the formatted file in OpenSearch Dashboards and perform file operations.	1. Sign in to the AWS Management Console. Under **Analytics**, choose **Amazon OpenSearch Service**. 2. In the navigation pane, choose **Domains**. 3. In the search box, enter the URL for your domain in [OpenSearch Dashboards](#). 4. Choose your dashboard , and then [sign in as the master user](#). 5. Browse the indexed data in table format. 6. Compare the input file against the formatted output file (indexed document) in OpenSearc h Dashboards. The dashboard view shows the added column headers for your formatted files. Confirm that the source data from your unformatt ed input files matches the target data in the dashboard view. 7. Perform actions such as search (for example, by using field names, values, or expressions), filter, and [DQL](#) (Dashboard Query	General AWS

Task	Description	Skills required
	Language) operations against the indexed file.	

Related resources

References

- Example COBOL copybook (IBM documentation)
- BMC Compuware File-AID (BMC documentation)

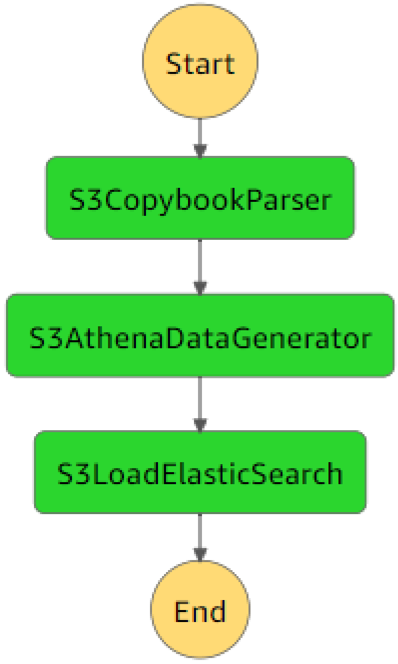
Tutorials

- Tutorial: Using an Amazon S3 trigger to invoke a Lambda function (AWS Lambda documentation)
- How do I create a serverless workflow with AWS Step Functions and AWS Lambda (AWS documentation)
- Using OpenSearch Dashboards with Amazon OpenSearch Service (AWS documentation)

Additional information

Step Functions graph

The following example shows a Step Functions graph. The graph shows the execution run status for the Lambda functions used in this pattern.



CloudWatch delivery logs

The following example shows successful delivery logs for the execution of the `s3toelasticsearch` execution.

2022-08-10T15:53:33.033-05:00	Number of processing documents: 100	
	2022-08-10T15:53:33.171-05:00	[INFO] 2022-08-10T20:53:33.171Z a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 POST https://search-essearch-3h4uqclifeqaj2vg4mphe7ffle.us-east-2.es.amazonaws.com:443/_bulk [status:200 request:0.100s]
	2022-08-10T15:53:33.172-05:00	Bulk write succeed: 100 documents

Containerize mainframe workloads that have been modernized by Blu Age

Richard Milner-Watts, Amazon Web Services

Summary

This pattern provides a sample container environment for running mainframe workloads that have been modernized by using the Blu Age tool. Blu Age converts legacy mainframe workloads into modern Java code. This pattern provides a wrapper around the Java application so you can run it by using container orchestration services such as Amazon Elastic Container Service (Amazon ECS) or Amazon Elastic Kubernetes Service (Amazon EKS).

For more information about modernizing your workloads by using Blu Age and AWS services, see these AWS Prescriptive Guidance publications:

- Running modernized Blu Age mainframe workloads on serverless AWS infrastructure
- Deploy an environment for containerized Blu Age applications by using Terraform

For assistance with using Blu Age to modernize your mainframe workloads, contact the Blu Age team by choosing **Contact our experts** on the Blu Age website. For assistance with migrating your modernized workloads to AWS, integrating them with AWS services, and moving them into production, contact your AWS account manager or fill out the AWS Professional Services form.

Prerequisites and limitations

Prerequisites

- A modernized Java application that was created by Blu Age. For testing purposes, this pattern provides a sample Java application that you can use as a proof of concept.
- A Docker environment that you can use to build the container.

Limitations

Depending on the container orchestration platform that you use, the resources that can be made available to the container (such as CPU, RAM, and storage) might be limited. For example, if you're using Amazon ECS with AWS Fargate, see the Amazon ECS documentation for limits and considerations.

Architecture

Source technology stack

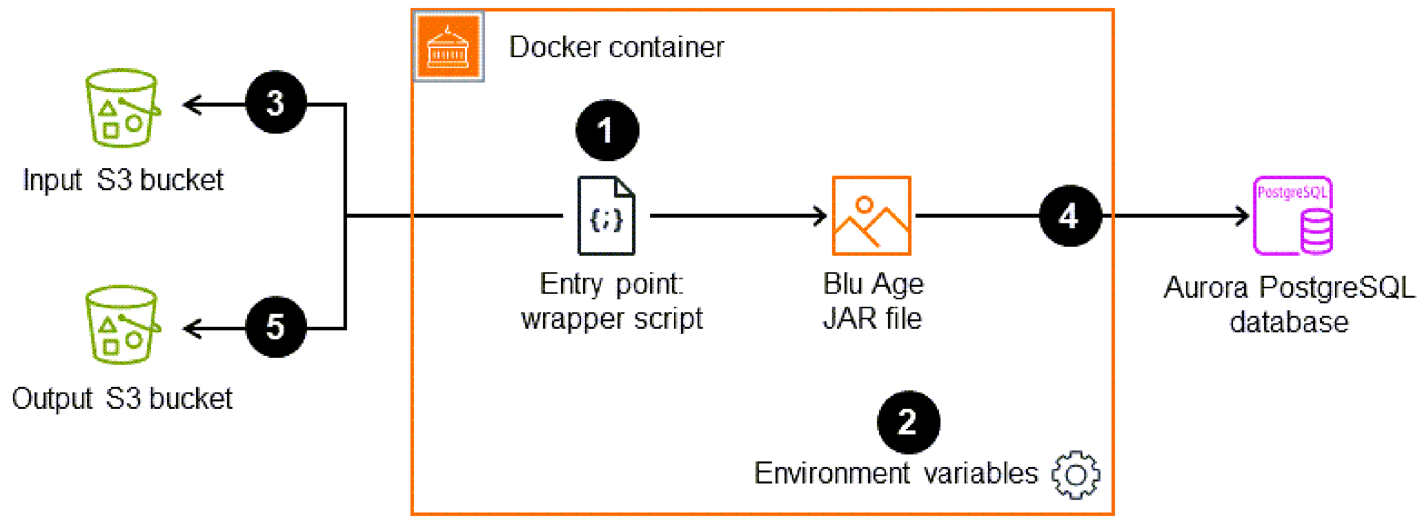
- Blu Age
- Java

Target technology stack

- Docker

Target architecture

The following diagram shows the architecture of the Blu Age application within a Docker container.



1. The entry point for the container is the wrapper script. This bash script is responsible for preparing the runtime environment for the Blu Age application and processing outputs.
2. Environment variables within the container are used to configure variables in the wrapper script, such as the Amazon Simple Storage Service (Amazon S3) bucket names and database credentials. Environment variables are supplied by either AWS Secrets Manager or Parameter Store, a capability of AWS Systems Manager. If you're using Amazon ECS as your container orchestration service, you can also hardcode the environment variables in the Amazon ECS task definition.
3. The wrapper script is responsible for pulling any input files from the S3 bucket into the container before you run the Blu Age application. The AWS Command Line Interface (AWS CLI) is

installed within the container. This provides a mechanism for accessing objects that are stored in Amazon S3 through the gateway virtual private cloud (VPC) endpoint.

4. The Java Archive (JAR) file for the Blu Age application might need to communicate with other data sources, such as Amazon Aurora.
5. After completion, the wrapper script delivers the resulting output files into an S3 bucket for further processing (for example, by Amazon CloudWatch logging services). The pattern also supports delivering zipped log files to Amazon S3, if you're using an alternative to standard CloudWatch logging.

Tools

AWS services

- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.

Tools

- Docker is a software platform for building, testing, and deploying applications. Docker packages software into standardized units called containers, which have everything the software needs to run, including libraries, system tools, code, and runtime. You can use Docker to deploy and scale applications into any environment.
- Bash is a command language interface (shell) for the GNU operating system.
- Java is the programming language and development environment used in this pattern.
- Blu Age is an AWS mainframe modernization tool that converts legacy mainframe workloads, including application code, dependencies, and infrastructure, into modern workloads for the cloud.

Code repository

The code for this pattern is available in the GitHub Blu Age sample container repository.

Best practices

- Externalize the variables for altering your application's behavior by using environment variables. These variables enable the container orchestration solution to alter the runtime environment without rebuilding the container. This pattern includes examples of environment variables that can be useful for Blu Age applications.
- Validate any application dependencies before you run your Blu Age application. For example, verify that the database is available and credentials are valid. Write tests in the wrapper script to verify dependencies, and fail early if they are not met.
- Use verbose logging within the wrapper script. Interacting directly with a running container can be challenging, depending on the orchestration platform and how long the job takes. Make sure that useful output is written to `STDOUT` to help diagnose any issues. For example, output might include the contents of the application's working directory both before and after you run the application.

Epics

Obtain a Blu Age application JAR file

Task	Description	Skills required
Option 1 - Work with Blu Age to obtain your application's JAR file.	The container in this pattern requires a Blu Age applicati on. Alternatively, you can use the sample Java applicati on that's provided with this pattern for a prototype. Work with the Blu Age team to obtain a JAR file for your application that can be baked into the container. If the JAR file isn't available, see the next task to use the sample application instead.	Cloud architect

Task	Description	Skills required
Option 2 - Build or use the supplied sample application JAR file.	This pattern provides a prebuilt sample JAR file. This file outputs the application's environment variables to `STDOUT` before sleeping for 30 seconds and exiting. This file is named `bluAgeSample.jar` and is located in the docker folder of the GitHub repository. If you want to alter the code and build your own version of the JAR file, use the source code located at ./java_sample/src/sample_java_app.java in the GitHub repository. You can use the build script at ./java_sample/build.sh to compile the Java source and build a new JAR fie.	App developer

Build the Blu Age container

Task	Description	Skills required
Clone the GitHub repository.	Clone the sample code repository by using the command: `git clone https://github.com/aws-samp`	AWS DevOps

Task	Description	Skills required
	`les/aws-blu-age-sa mple-container`	
Use Docker to build the container.	Use Docker to build the container before you push it to a Docker registry such as Amazon ECR: 1. From your chosen terminal, navigate to the `docker` folder within your local GitHub repository. 2. Use this command to build the container: `docker build -t <tag> .` where `<tag>` is the container name you want to use.	AWS DevOps
Test the Blu Age container.	(Optional) If necessary, test the container locally by using the command: `docker run -it <tag> /bin/bash`	AWS DevOps

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Authenticate to your Docker repository.</td><td>If you plan to use Amazon ECR, follow the instructions in the Amazon ECR documentation to install and configure the AWS CLI and authenticate the Docker CLI to your default registry.

We recommend that you use the get-login-password command for authentication.

Note
The Amazon ECR console provides a pre-populated version of this command if you use the View push commands button. For more information, see the Amazon ECR documentation.

<pre><code>aws ecr get-login
-password --region
 <region> | docker
 login --username
 AWS --password-stdin
 <account>.dkr.ecr.
<region>.amazonaws
.com</code></pre></td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	If you don't plan to use Amazon ECR, follow the instructions provided for your container registry system.	
Create a container repository.	Create a repository in Amazon ECR. For instructions, see the pattern [Deploy an environment for containerized Blu Age applications by using Terraform](). If you're using another container registry system, follow the instructions provided for that system.	AWS DevOps

Task	Description	Skills required

Tag and push your container to the target repository.	If you're using Amazon ECR:
	1. Tag the local Docker image with the Amazon ECR registry and repository, so you can push it to your remote repository:
	`docker tag <tag>:latest <account>.dkr.ecr.<region>.amazonaws.com/<repository>:<versionNumber>`
	2. Push the image to the remote repository:
	`docker push <account>.dkr.ecr.<region>.amazonaws.com/<repository>:<versionNumber>`
	For more information, see Pushing a Docker image in the *Amazon ECR User Guide*.
		AWS DevOps

Related resources

AWS resources

- AWS Blu Age sample container repository
- Running modernized Blu Age mainframe workloads on serverless AWS infrastructure
- Deploy an environment for containerized Blu Age applications by using Terraform

- Using Amazon ECR with the AWS CLI (*Amazon ECR User Guide*)
- Private registry authentication (*Amazon ECR User Guide*)
- Amazon ECS documentation
- Amazon EKS documentation

Additional resources

- Blu Age website
- Docker website

Convert and unpack EBCDIC data to ASCII on AWS by using Python

Luis Gustavo Dantas, Amazon Web Services

Summary

Because mainframes typically host critical business data, modernizing data is one of the most important tasks when migrating data to the Amazon Web Services (AWS) Cloud or other American Standard Code for Information Interchange (ASCII) environment. On mainframes, data is typically encoded in extended binary-coded decimal interchange code (EBCDIC) format. Exporting database, Virtual Storage Access Method (VSAM), or flat files generally produces packed, binary EBCDIC files, which are more complex to migrate. The most commonly used database migration solution is change data capture (CDC), which, in most cases, automatically converts data encoding. However, CDC mechanisms might not be available for these database, VSAM, or flat files. For these files, an alternative approach is required to modernize the data.

This pattern describes how to modernize EBCDIC data by converting it to ASCII format. After conversion, you can load the data into distributed databases or have applications in the cloud process the data directly. The pattern uses the conversion script and sample files in the mainframe-data-utilities GitHub repository.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An EBCDIC input file and its corresponding common business-oriented language (COBOL) copybook. A sample EBCDIC file and COBOL copybook are included in the mainframe-data-utilities GitHub repository. For more information about COBOL copybooks, see Enterprise COBOL for z/OS 6.4 Programming Guide on the IBM website.

Limitations

- File layouts defined inside COBOL programs are not supported. They must be made available separately.

Product versions

- Python version 3.8 or later

Architecture

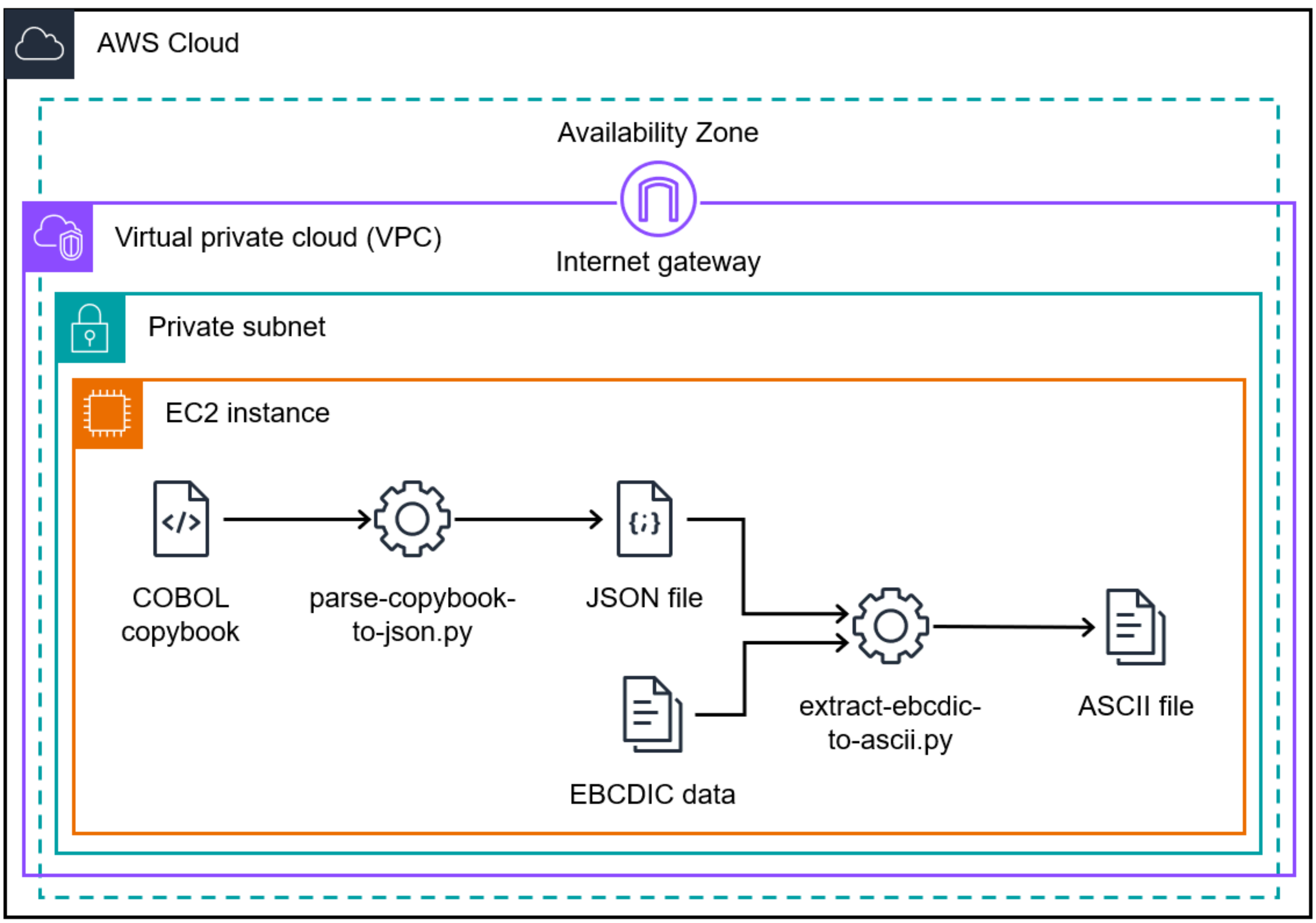
Source technology stack

- EBCDIC data on a mainframe
- COBOL copybook

Target technology stack

- Amazon Elastic Compute Cloud (Amazon EC2) instance in a virtual private cloud (VPC)
- Amazon Elastic Block Store (Amazon EBS)
- Python and its required packages, JavaScript Object Notation (JSON), sys, and datetime
- ASCII flat file ready to be read by a modern application or loaded in a relational database table

Target architecture



The architecture diagram shows the process of converting an EBCDIC file to an ASCII file on an EC2 instance:

1. Using the **parse_copybook_to_json.py** script, you convert the COBOL copybook to a JSON file.
2. Using the JSON file and the **extract_ebcdic_to_ascii.py** script, you convert the EBCDIC data to an ASCII file.

Automation and scale

After the resources needed for the first manual file conversions are in place, you can automate file conversion. This pattern doesn't include instructions for automation. There are multiple ways to automate the conversion. The following is an overview of one possible approach:

1. Encapsulate the AWS Command Line Interface (AWS CLI) and Python script commands into a shell script.
2. Create an AWS Lambda function that asynchronously submits the shell script job into an EC2 instance. For more information, see [Scheduling SSH jobs using AWS Lambda](https://aws.amazon.com/blogs/compute/scheduling-ssh-jobs-using-aws-lambda/).
3. Create an Amazon Simple Storage Service (Amazon S3) trigger that invokes the Lambda function every time a legacy file is uploaded. For more information, see [Using an Amazon S3 trigger to invoke a Lambda function](https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html).

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need, and quickly scale them up or down.
- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

Other tools

- GitHub is a code-hosting service that provides collaboration tools and version control.
- Python is a high-level programming language.

Code repository

The code for this pattern is available in the mainframe-data-utilities GitHub repository.

Epics

Prepare the EC2 instance

Task	Description	Skills required
Launch an EC2 instance.	The EC2 instance must have outbound internet access. This allows the instance to access the Python source code available on GitHub. To create the instance: 1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2. 2. Launch an EC2 Linux instance. Use a public IP address and allow inbound access through port 22. Ensure that the storage size of the instance is at least twice the size of the EBCDIC data file. For instructions, see the Amazon EC2 documentation.	General AWS
Install Git.	1. Using a secure shell (SSH) client, connect to the EC2	General AWS, Linux

Task	Description	Skills required
	instance you just launched. For more information, see [Connect to your Linux instance](). 2. In the Amazon EC2 console, run the following command. This installs Git on the EC2 instance. `sudo yum install git` 3. Run the following command and confirm that Git has been successfully installed. `git --version`	

Task	Description	Skills required

Install Python.	1. In the Amazon EC2 console, run the following command. This installs Python on the EC2 instance.
	`sudo yum install python3`
	2. In the Amazon EC2 console, run the following command. This installs Pip3 on the EC2 instance.
	`sudo yum install python3-pip`
	3. In the Amazon EC2 console, run the following command. This installs AWS SDK for Python (Boto3) on the EC2 instance.
	`sudo pip3 install boto3`
	4. In the Amazon EC2 console, run the following command, where `<us-east-1>` is the code for your AWS Region. For a complete list of Region codes, see Available Regions in the Amazon EC2 documentation.
		General AWS, Linux

Task	Description	Skills required
	`export AWS_DEFAULT_REGION=<us-east-1>`	
Clone the GitHub repository.	1. In the Amazon EC2 console, run the following command. This clones the **mainframe-data-utilities** repository from GitHub and opens the default copy location, the home folder. `git clone https://github.com/aws-samples/mainframe-data-utilities.git` 2. In the home folder, confirm that the `mainframe-data-utilities` folder is present.	General AWS, GitHub

Create the ASCII file from the EBCDIC data

Task	Description	Skills required
Parse the COBOL copybook into the JSON layout file.	Inside the `mainframe-data-utilities` folder, run the **parse_copybook_to_json.py** script. This automation module reads the file layout from a COBOL copybook and creates a JSON file. The JSON file contains the informati	General AWS, Linux

Task	Description	Skills required

	on required to interpret and extract the data from the source file. This creates the JSON metadata from the COBOL copybook.
	
	The following command converts the COBOL copybook to a JSON file.
	
	`python3 parse_copybook_to_json.py \ -copybook LegacyReference/COBPACK2.cpy \ -output sample-data/cobpack2-list.json \ -dict sample-data/cobpack2-dict.json \ -ebcdic sample-data/COBPACK.OUTFILE.txt \ -ascii sample-data/COBPACK.ASCII.txt \ -print 10000`
	
	The script prints the received arguments.
	
	`------------------------------------------------------------------------------------ Copybook file...............| LegacyReference/COBPACK2.cpy Parsed copybook (JSON List).| sample-data/cobpack2-list.json`	

Task	Description	Skills required
	``` JSON Dict (document ation)...| sample-da ta/cobpack2-dict.json ASCII file..... ....| sample- data/COBPACK.ASCII.t xt EBCDIC file..... ....| sample- data/COBPACK.OUTFILE .txt Print each..... ....| 10000 --------------------- ------------------ ------------------ -------------- ``` For more information about the arguments, see the README file in the GitHub repository.	

Task	Description	Skills required

Inspect the JSON layout file.	1. Navigate to the output path defined in the **parse_copybook_to_json.py** script.
	2. Check the creation time of the **sample-data/cobpack2-list.json** file to confirm that you have selected the appropriate JSON layout file.
	3. Examine the JSON file and confirm that the contents are similar to the following.
	
	`"input": "extract-ebcdic-to-ascii/COBPACK.OUTFILE.txt",`
	`"output": "extract-ebcdic-to-ascii/COBPACK.ASCII.txt",`
	`"max": 0,`
	`"skip": 0,`
	`"print": 10000,`
	`"lrecl": 150,`
	`"rem-low-values": true,`
	`"separator": "|",`
	`"transf": [`
	`{`
	`"type": "ch",`
	`"bytes": 19,`
	`"name": "OUTFILE-TEXT"`
	`}`
	
	The most important attributes of the JSON layout file are:
		General AWS, JSON

Task	Description	Skills required
	• `input` – Contains the path of the EBCDIC file to be converted • `output` – Defines the path where the ASCII file will be generated • `lrecl` – Specifies the size in bytes of the logical record length • `transf` – Lists all fields and their size in bytes For more information about the JSON layout file, see the README file in the GitHub repository.	

Task	Description	Skills required

Create the ASCII file.	Run the **extract_ebcdic_to_ascii.py** script, which is included in the cloned GitHub repository. This script reads the EBCDIC file and writes a converted and readable ASCII file.
	`python3 extract_ebcdic_to_ascii.py -local-json sample-data/cobpack2-list.json`
	As the script processes the EBCDIC data, it prints a message for every batch of 10,000 records. See the following example.
	`------------------------------------------------------------------------------------------------`
	`2023-05-15 21:21:46.322253 | Local Json file   | -local-json | sample-data/cobpack2-list.json`
	`2023-05-15 21:21:47.034556 | Records processed | 10000`
	`2023-05-15 21:21:47.736434 | Records processed | 20000`
	`2023-05-15 21:21:48.441696 | Records processed | 30000`
		General AWS

Task	Description	Skills required
	``` 2023-05-15 21:21:49. 173781 | Records processed | 40000 2023-05-15 21:21:49. 874779 | Records processed | 50000 2023-05-15 21:21:50. 705873 | Records processed | 60000 2023-05-15 21:21:51. 609335 | Records processed | 70000 2023-05-15 21:21:52. 292989 | Records processed | 80000 2023-05-15 21:21:52. 938366 | Records processed | 89280 2023-05-15 21:21:52. 938448 Seconds 6.616232 ``` For information about how to change the print frequency, see the README file in the GitHub repository.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Examine the ASCII file.</td><td>

Check the creation time of the extract-ebcdic-to-ascii/COBPACK.ASCII.txt file to verify that it was recently created.
In the Amazon EC2 console, enter the following command. This opens the first record of the ASCII file.
<pre><code>head sample-data/
COBPACK.ASCII.txt -n
 1| xxd</code></pre>
Examine the contents of the first record. Because EBCDIC files are usually binary, they don't have carriage return and line feed (CRLF) special characters. The extract_ebcdic_to_ascii.py script adds a pipe character as a column separator, which is defined in the script parameters.

If you used the sample EBCDIC file provided, the following is the first record in the ASCII file.
</td><td>General AWS, Linux</td></tr>
</table>

Task	Description	Skills required

	<pre>00000000: 2d30 3030 3030 3030 3030 3130 3030 3030 -000000000100000
	00000010: 3030 307c 3030 3030 3030 3030 3031 3030 000|000000000100
	00000020: 3030 3030 3030 7c2d 3030 3030 3030 3030 000000|-00000000
	00000030: 3031 3030 3030 3030 3030 7c30 7c30 7c31 0100000000|0|0|1
	00000040: 3030 3030 3030 3030 7c2d 3130 3030 3030 00000000|-100000
	00000050: 3030 307c 3130 3030 3030 3030 307c 2d31 000|100000000|-1
	00000060: 3030 3030 3030 3030 7c30 3030 3030 7c30 00000000|00000|0
	00000070: 3030 3030 7c31 3030 3030 3030 3030 7c2d 0000|100000000|-
	00000080: 3130 3030 3030 3030 307c 3030 3030 3030 100000000|000000
	00000090: 3030 3030 3130 3030 3030 3030 307c 2d30 000010000000|-0
	000000a0: 3030 3030 3030 3030 3031 3030</pre>	

Task	Description	Skills required
	``` 3030 3030 000000000 1000000 000000b0: 3030 7c41 7c41 7c0a 00|A|A|. ```	

Task	Description	Skills required

Evaluate the EBCDIC file.	In the Amazon EC2 console, enter the following command. This opens the first record of the EBCDIC file.
	```head sample-data/COBPACK.OUTFILE.txt -c 150 | xxd```
	If you used the sample EBCDIC file, the following is the result.
	```00000000: 60f0 f0f0 f0f0 f0f0 f0f0 f1f0 f0f0 f0f0 `....... .... 00000010: f0f0 f0f0 f0f0 f0f0 f0f0 f0f0 f1f0 f0f0 .... 00000020: f0f0 f0f0 f0f0 f0f0 f0f0 f0f0 f0f0 f1f0 .... 00000030: f0f0 f0f0 f0f0 d000 0000 0005 f5e1 00fa .... 00000040: 0a1f 0000 0000 0005 f5e1 00ff ffff fffa .... 00000050: 0a1f 0000 000f 0000 0c10 0000 000f 1000 .... 00000060: 0000 0d00 0000 0000 1000 0000```
		General AWS, Linux, EBCDIC

Task	Description	Skills required
	``` 0f00 0000 .... 00000070: 0000 1000 0000 0dc1 c100 0000 0000 0000 .... 00000080: 0000 0000 0000 0000 0000 0000 0000 0000 .... 00000090: 0000 0000 0000 ...... ``` To evaluate the equivalence between the source and target files, comprehensive knowledge of EBCDIC is required. For example, the first character of the sample EBCDIC file is a hyphen (-). In hexadecimal notation of the EBCDIC file, this character is represented by 60, and in hexadecimal notation of the ASCII file, this character is represented by 2D. For an EBCDIC-to-ASCII conversion table, see EBCDIC to ASCII on the IBM website.	

Related resources

References

- The EBCDIC character set (IBM documentation)
- EBCDIC to ASCII (IBM documentation)

- COBOL (IBM documentation)
- Basic JCL concepts (IBM documentation)
- Connect to your Linux instance (Amazon EC2 documentation)

Tutorials

- Scheduling SSH jobs using AWS Lambda (AWS blog post)
- Using an Amazon S3 trigger to invoke a Lambda function (AWS Lambda documentation)

Convert mainframe files from EBCDIC format to character-delimited ASCII format in Amazon S3 using AWS Lambda

Luis Gustavo Dantas, Amazon Web Services

Summary

This pattern shows you how to launch an AWS Lambda function that automatically converts mainframe EBCDIC (Extended Binary Coded Decimal Interchange Code) files to character-delimited ASCII (American Standard Code for Information Interchange) files. The Lambda function runs after the ASCII files are uploaded to an Amazon Simple Storage Service (Amazon S3) bucket. After the file conversion, you can read the ASCII files on x86-based workloads or load the files into modern databases.

The file conversion approach demonstrated in this pattern can help you overcome the challenges of working with EBCDIC files on modern environments. Files encoded in EBCDIC often contain data represented in a binary or packed decimal format, and fields are fixed-length. These characteristics create obstacles because modern x86-based workloads or distributed environments generally work with ASCII-encoded data and can't process EBCDIC files.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An S3 bucket
- An AWS Identity and Access Management (IAM) user with administrative permissions
- AWS CloudShell
- Python 3.8.0 or later
- A flat file encoded in EBCDIC and its corresponding data structure in a common business-oriented language (COBOL) copybook

> **Note**
>
> This pattern uses a sample EBCDIC file (CLIENT.EBCDIC.txt) and its corresponding COBOL copybook (COBKS05.cpy). Both files are available in the GitHub mainframe-data-utilities repository.

Limitations

- COBOL copybooks usually hold multiple layout definitions. The [mainframe-data-utilities] project can parse this kind of copybook but can't infer which layout to consider on data conversion. This is because copybooks don't hold this logic (which remains on COBOL programs instead). Consequently, you must manually configure the rules for selecting layouts after you parse the copybook.
- This pattern is subject to [Lambda quotas].

Architecture

Source technology stack

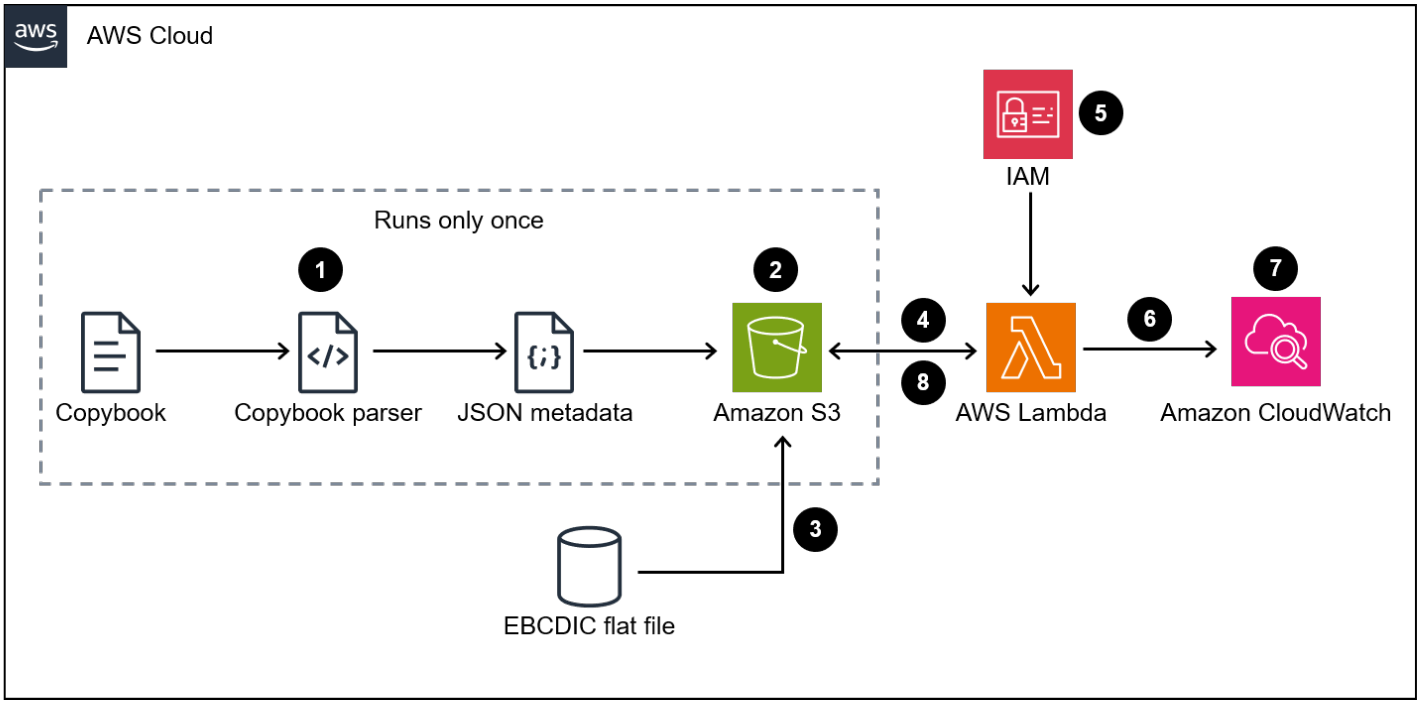
- IBM z/OS, IBM i, and other EBCDIC systems
- Sequential files with data encoded in EBCDIC (such as IBM Db2 unloads)
- COBOL copybook

Target technology stack

- Amazon S3
- Amazon S3 event notification
- IAM
- Lambda function
- Python 3.8 or later
- Mainframe Data Utilities
- JSON metadata
- Character-delimited ASCII files

Target architecture

The following diagram shows an architecture for converting mainframe EBCDIC files to ASCII files.



The diagram shows the following workflow:

1. The user runs the copybook parser script to convert the COBOL copybook into a JSON file.
2. The user uploads the JSON metadata to an S3 bucket. This makes the metadata readable by the data conversion Lambda function.
3. The user or an automated process uploads the EBCDIC file to the S3 bucket.
4. The S3 notification event triggers the data conversion Lambda function.
5. AWS verifies the S3 bucket read-write permissions for the Lambda function.
6. Lambda reads the file from the S3 bucket and locally converts the file from EBCDIC to ASCII.
7. Lambda logs the process status in Amazon CloudWatch.
8. Lambda writes the ASCII file back to Amazon S3.

Note

The copybook parser script runs only once, after it converts the metadata to JSON and then uploads that data to an S3 bucket. After the initial conversion, any EBCDIC file that uses the same JSON file that's uploaded to the S3 bucket will use the same metadata.

Tools

AWS tools

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications that you run on AWS in real time.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS CloudShell is a browser-based shell that you can use to manage AWS services by using the AWS Command Line Interface (AWS CLI) and a range of preinstalled development tools.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. Lambda runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Other tools

- GitHub is a code-hosting service that provides collaboration tools and version control.
- Python is a high-level programming language.

Code

The code for this pattern is available in the GitHub mainframe-data-utilities repository.

Best practices

Consider the following best practices:

- Set the required permissions at the Amazon Resource Name (ARN) level.
- Always grant least-privilege permissions for IAM policies. For more information, see Security best practices in IAM in the IAM documentation.

Epics

Create environment variables and a working folder

Task	Description	Skills required

Create the environment variables.	Copy the following environment variables to a text editor, and then replace the <placeholder> values in the following example with your resource values:
	`bucket=<your_bucket_name>`
	`account=<your_account_number>`
	`region=<your_region_code>`
	Note
	You will create references to your S3 bucket, AWS account, and AWS Region later.
	To define environment variables, open the [CloudShell console](), and then copy and paste your updated environment variables onto the command line.
		General AWS

Task	Description	Skills required
	Note You must repeat this step every time the CloudShell session restarts.	
Create a working folder.	To simplify the resource clean-up process later on, create a working folder in CloudShell by running the following command: `mkdir workdir; cd workdir` **Note** You must change the directory to the working directory (`workdir`) every time you lose a connectio n to your CloudShell session.	General AWS

Define an IAM role and policy

Task	Description	Skills required
Create a trust policy for the Lambda function.	The EBCDIC converter runs in a Lambda function. The function must have an IAM	General AWS

Task	Description	Skills required

	role. Before you create the IAM role, you must define a trust policy document that enables resources to assume that policy.
	
	From the CloudShell working folder, create a policy document by running the following command:
	
	`E2ATrustPol=$(cat <<EOF`
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Effect": "Allow",`
	`            "Principal": {`
	`                "Service": "lambda.amazonaws.com"`
	`            },`
	`            "Action": "sts:AssumeRole"`
	`        }`
	`    ]`
	`}`
	`EOF`
	`)`
	`printf "$E2ATrustPol" > E2ATrustPol.json`	

Task	Description	Skills required
Create the IAM role for Lambda conversion.	To create an IAM role, run the following AWS CLI command from the CloudShell working folder: `aws iam create-role --role-name E2AConvLambdaRole --assume-role-policy-document file://E2ATrustPol.json`	General AWS

Task	Description	Skills required

Create the IAM policy document for the Lambda function.	The Lambda function must have read-write access to the S3 bucket and write permissions for Amazon CloudWatch Logs.
	
	To create an IAM policy, run the following command from the CloudShell working folder:
	
	`E2APolicy=$(cat <<EOF`
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Sid": "Logs",`
	`            "Effect": "Allow",`
	`            "Action": [`
	`                "logs:PutLogEvents",`
	`                "logs:CreateLogStream",`
	`                "logs:CreateLogGroup"`
	`            ],`
	`            "Resource": [`
	`                "arn:aws:logs:*:*:log-group:*",`
	`                "arn:aws:logs:*:*:log-group:*:log-stream:*"`
	`            ]`
		General AWS

Task	Description	Skills required

	``
	        },
	        {
	            "Sid": "S3",
	            "Effect":
	 "Allow",
	            "Action": [
	
	 "s3:GetObject",
	
	 "s3:PutObject",
	
	 "s3:GetObjectVersion"
	            ],
	            "Resource":
	 [
	
	 "arn:aws:s3:::%s/*",
	
	 "arn:aws:s3:::%s"
	            ]
	        }
	    ]
	}
	EOF
	)
	printf "$E2APolicy"
	 "$bucket" "$bucket"
	 > E2AConvLambdaPolic
	y.json
	``	

Task	Description	Skills required
Attach the IAM policy document to the IAM role.	To attach the IAM policy to the IAM role, run the following command from your CloudShell working folder: `aws iam put-role-policy --role-name E2AConvLambdaRole --policy-name E2AConvLambdaPolicy --policy-document file://E2AConvLambdaPolicy.json`	General AWS

Create the Lambda function for EBCDIC conversion

Task	Description	Skills required
Download the EBCDIC conversion source code.	From the CloudShell working folder, run the following command to download the mainframe-data-utilities source code from GitHub: `git clone https://github.com/aws-samples/mainframe-data-utilities.git mdu`	General AWS
Create the ZIP package.	From the CloudShell working folder, run the following command to create the ZIP package that creates the Lambda function for EBCDIC conversion:	General AWS

Task	Description	Skills required
	`cd mdu; zip ../mdu.zip *.py; cd ..`	
Create the Lambda function.	From the CloudShell working folder, run the following command to create the Lambda function for EBCDIC conversion: `aws lambda create-function \ --function-name E2A \ --runtime python3.9 \ --zip-file fileb://mdu.zip \ --handler extract_ebcdic_to_ascii.lambda_handler \ --role arn:aws:iam::$account:role/E2AConvLambdaRole \ --timeout 10 \ --environment "Variables={layout=$bucket/layout/}"` **Note** The environment variable layout tells the Lambda function where the JSON metadata resides.	General AWS

Task	Description	Skills required
Create the resource-based policy for the Lambda function.	From the CloudShell working folder, run the following command to allow your Amazon S3 event notificat ion to trigger the Lambda function for EBCDIC conversio n: ```aws lambda add-permi ssion \ --function-name E2A \ --action lambda:In vokeFunction \ --principal s3.amazon aws.com \ --source-arn arn:aws:s 3:::$bucket \ --source-account $account \ --statement-id 1```	General AWS

Create the Amazon S3 event notification

Task	Description	Skills required
Create the configuration document for the Amazon S3 event notification.	The Amazon S3 event notification initiates the EBCDIC conversion Lambda function when files are placed in the input folder. From the CloudShell working folder, run the following command to create the JSON document for the Amazon S3 event notification:	General AWS

Task	Description	Skills required

	``
	{
	"LambdaFunctionC
	onfigurations": [
	    {
	      "Id": "E2A",
	      "LambdaFu
	nctionArn": "arn:aws:
	lambda:%s:%s:funct
	ion:E2A",
	      "Events":
	 ["s3:ObjectCreated:
	Put"],
	      "Filter": {
	        "Key": {
	          "FilterRu
	les": [
	            {
	              "Name":
	 "prefix",
	              "Value":
	 "input/"
	            }
	          ]
	        }
	      }
	    }
	  ]
	}
	EOF
	)
	printf "$S3E2AEvent"
	 "$region" "$account" >
	 S3E2AEvent.json
	``	

Task	Description	Skills required
Create the Amazon S3 event notification.	From the CloudShell working folder, run the following command to create the Amazon S3 event notification: ```aws s3api put-bucket-notification-configuration --bucket $bucket --notification-configuration file://S3E2AEvent.json```	General AWS

Create and upload the JSON metadata

Task	Description	Skills required
Parse the COBOL copybook.	From the CloudShell working folder, run the following command to parse a sample COBOL copybook into a JSON file (which defines how to read and slice the data file properly): ```python3 mdu/parse_copybook_to_json.py \ -copybook mdu/LegacyReference/COBKS05.cpy \ -output CLIENT.json \ -output-s3key CLIENT.ASCII.txt \ -output-s3bkt $bucket \ -output-type s3 \```	General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td><pre><code>-print   25</code></pre></td><td></td></tr>
<tr><td>Add the transformation rule.</td><td>The sample data file and its corresponding COBOL copybook is a multi-layout file. This means that the conversion must slice data based on certain rules. In this case, bytes on position 3 and 4 in each row define the layout.

From the CloudShell working folder, edit the <code>CLIENT.json</code> file and change the contents from <code>"transf-rule": [],</code> to the following:
<pre><code>"transf-rule": [
{
"offset": 4,
"size": 2,
"hex": "0002",
"transf": "transf1"
},
{
"offset": 4,
"size": 2,
"hex": "0000",
"transf": "transf2"
}
],</code></pre></td><td>General AWS, IBM Mainframe, Cobol</td></tr>
</table>

Task	Description	Skills required
Upload the JSON metadata to the S3 bucket.	From the CloudShell working folder, run the following AWS CLI command to upload the JSON metadata to your S3 bucket: `aws s3 cp CLIENT.json s3://$bucket/layout/CLIENT.json`	General AWS

Convert the EBCDIC file

Task	Description	Skills required
Send the EBCDIC file to the S3 bucket.	From the CloudShell working folder, run the following command to send the EBCDIC file to the S3 bucket: `aws s3 cp mdu/sample-data/CLIENT.EBCDIC.txt s3://$bucket/input/` **Note** We recommend that you set different folders for input (EBCDIC) and output (ASCII) files to avoid calling the Lambda conversion function again when the ASCII	General AWS

Task	Description	Skills required
	file is uploaded to the S3 bucket.	

Check the output.	From the CloudShell working folder, run the following command to check if the ASCII file is generated in your S3 bucket:
	`awss3 ls s3://$bucket/`
	Note
	The data conversion can take several seconds to happen. We recommend that you check for the ASCII file a few times.
	After the ASCII file is available, run the following command to download the file from the S3 bucket to the current folder:
	`aws s3 cp s3://$bucket/ CLIENT.ASCII.txt .`
	Check the ASCII file content:
	`head CLIENT.ASCII.txt`
		General AWS

Clean the environment

Task	Description	Skills required

(Optional) Prepare the variables and folder.	If you lose connection with CloudShell, reconnect and then run the following command to change the directory to the working folder:
	`cd workdir`
	Ensure that the environment variables are defined:
	`bucket=<your_bucket_name>`
	`account=<your_account_number>`
	`region=<your_region_code>`
		General AWS

Remove the notification configuration for the bucket.	From the CloudShell working folder, run the following command to remove the Amazon S3 event notification configuration:
	`aws s3api put-bucket-notification-configuration \`
	`--bucket=$bucket \`
	`--notification-configuration="{}"`
		General AWS

Delete the Lambda function.	From the CloudShell working folder, run the following	General AWS

Task	Description	Skills required
	command to delete the Lambda function for the EBCDIC converter: `awslambdadelete-function--function-nameE2A`	

Delete the IAM role and policy.	From the CloudShell working folder, run the following command to remove the EBCDIC converter role and policy:
	`aws iam delete-role-policy --role-name E2AConvLambdaRole --policy-name E2AConvLambdaPolicy`
	
	`aws iam delete-role --role-name E2AConvLambdaRole`
		General AWS

Delete the files generated in the S3 bucket.	From the CloudShell working folder, run the following command to delete the files generated in the S3 bucket:
	`aws s3 rm s3://$bucket/layout --recursive`
	`aws s3 rm s3://$bucket/input --recursive`
	`aws s3 rm s3://$bucket/CLIENT.ASCII.txt`
		General AWS

Task	Description	Skills required
Delete the working folder.	From the CloudShell working folder, run the following command to remove `workdir` and its contents: `cd ..; rm -Rf workdir`	General AWS

Related resources

- Mainframe Data Utilities README (GitHub)
- The EBCDIC character set (IBM documentation)
- EBCDIC to ASCII (IBM documentation)
- COBOL (IBM documentation)
- Using an Amazon S3 trigger to invoke a Lambda function (AWS Lambda documentation)

Convert mainframe data files with complex record layouts using Micro Focus

Peter West, Amazon Web Services

Summary

This pattern shows you how to convert mainframe data files with non-text data and complex record layouts from EBCDIC (Extended Binary Coded Decimal Interchange Code) character encoding to ASCII (American Standard Code for Information Interchange) character encoding by using a Micro Focus structure file. To complete the file conversion, you must do the following:

1. Prepare a single source file that describes all the data items and record layouts in your mainframe environment.
2. Create a structure file that contains the record layout of the data by using the Micro Focus Data File Editor as part of the Micro Focus Classic Data File Tools or Data File Tools. The structure file identifies the non-text data so that you can correctly convert your mainframe files from EBCDIC to ASCII.
3. Test the structure file by using the Classic Data File Tools or Data File Tools.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Micro Focus Enterprise Developer for Windows, available through AWS Mainframe Modernization

Product versions

- Micro Focus Enterprise Server 7.0 and later

Tools

- Micro Focus Enterprise Developer provides the run environment for applications created with any integrated development environment (IDE) variant of Enterprise Developer.
- Micro Focus Classic Data File Tools help you convert, navigate, edit, and create data files. The Classic Data File Tools include Data File Converter, Record Layout Editor, and Data File Editor.

- Micro Focus Data File Tools help you create, edit, and move data files. The Data File Tools include Data File Editor, File Conversion Utilities, and the Data File Structure Command Line Utility.

Epics

Prepare the source file

Task	Description	Skills required
Identify source components.	Identify all possible record layouts for the file, including any redefinitions that contain non-text data. If you have layouts that contain redefinitions, you must factor these layouts down to unique layouts that describe each possible permutation of the data structure. Typically, a data file's record layouts can be described by the following archetypes: • Record layout with only text data • Record layout with non-text data • Record layout with non-text data subordinate to a REDEFINES clause For more information on creating flattened record layouts for files that contain complex record layouts, see	App developer

Task	Description	Skills required
	Rehosting EBCDIC applications on ASCII environments for mainframe migrations.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Identify record layout conditions.</td><td>For files with multiple record layouts or files that contain complex layouts with a REDEFINES clause, identify the data and conditions within a record that you can use to define which layout to use during conversion. We recommend that you discuss this task with a subject matter expert (SME) who understan ds the programs that process these files.

For example, a file might contain two record types that contain non-text data. You can inspect the source and possibly find code similar to the following:

<pre>MOVE "M" TO PART-TYPE
 MOVE "MAIN ASSEMBLY" TO
 PART-NAME
MOVE "S" TO PART-TYPE
 MOVE "SUB ASSEMBLY 1"
 TO PART-NAME</pre>
The code helps you identify the following:

• The "PART-TYPE" field is used to determine the record type
• The value "M" is used for the "M-PART-RECORD"</td><td>App developer</td></tr>
</table>

Task	Description	Skills required
	• The value "S" is used for the "S-PART-RECORD" You can document the values that are used by this field to associate the record layouts with the correct data records in the file.	
Build the source file.	If the file is described over multiple source files or if the record layout contains non-text data that's subordina te to a REDEFINES clause, then create a new source file that contains the record layouts. The new program doesn't need to describe the file using SELECT and FD statements. The program can simply contain the record descriptions as 01 levels within Working-Storage. **Note** You can create a source file for each data file or create a master source file that describes all the data files.	App developer

Task	Description	Skills required
Compile the source file.	Compile the source file to build the data dictionary. We recommend that you compile the source file by using the EBCDIC character set. If the IBMCOMP directive or ODOSLIDE directives are being used, then you must use these directives in the source file too. **Note** IBMCOMP affects the byte storage of COMP fields and ODOSLIDE affects padding on OCCURS VARYING structures. If these directives are set incorrectly, then the conversion tool won't read the data record correctly. This results in bad data in the converted file.	App developer

(Option A) Create the structure file using Classic Data File Tools

Task	Description	Skills required
Start the tool and load the dictionary.	1. Choose the Windows Start menu icon, search for and choose **Micro Focus**	App developer

Task	Description	Skills required
	Enterprise Developer, and then choose **Classic Data File Tools**. 2. Choose **File**, and then choose **Record Layout**. 3. In the **Select a file to construct the layouts from** dialog box, for **File name**, select the IDY (`.idy`) file that was created when you compiled the source file earlier. Then, choose **Open**. 4. To confirm that Classic Data File Tools is using EBCDIC, in the **Data File Tools** dialog box, choose **YES** if the IDY file is set to **EBCDIC** and **Datatools** is set to **ANSI**.	

Task	Description	Skills required
Create the default record layout.	Use the default record layout for all records that don't match any conditional layouts. 1. In the **Layout** window, expand the data structure , and then locate the 01 level used for the default layout. 2. Right-click the 01 item, and then choose **New Layout**. 3. In the **New Record Layout Wizard** dialog box, choose **Default Layout**, and then choose **Next**. 4. Choose **Finish**. The default layout appears in the **Layouts** pane and can be identified by the red folder icon.	App developer

Task	Description	Skills required
Create a conditional record layout.	Use the conditional record layout when there is more than one record layout in a file. 1. In the **Layouts** pane, expand the data structure , and then locate the 01 level used for the condition al layout. 2. Right-click the 01 item, and then choose **New Layout**. 3. In the **New Record Layout Wizard** dialog box, choose **Conditional Layout**, and then choose **Next**. 4. Choose **Finish**. The conditional layout appears in the **Layouts** pane and can be identified by the yellow folder icon. 5. Expand the conditional layout, right-click the field where you must place a condition, and then choose **Properties**. 6. In the **Field Propertie s** dialog box, enter the condition. Confirm that the character set is set to **EBCDIC**, and then choose **OK**. A checkmark appears	App developer

Task	Description	Skills required
	next to the field that has a condition set. 7. Repeat steps 5–6 for any other fields that require conditions for this layout. 8. Repeat steps 1–6 for any other conditional layouts that must be added. 9. Choose **File**, choose **Save As**, and then save the structure file to disk.	

(Option B) Create the structure file using Data File Tools

Task	Description	Skills required
Start the tool and load the dictionary.	1. Choose the Windows Start menu icon, search for and choose **Micro Focus Enterprise Developer**, and then choose **Data File Tools**. 2. Choose **File**, **New**, **Structure File**. 3. In the **Open** dialog box, for **File name**, select the IDY (`.idy`) file that was created when you compiled the source file earlier. Then, choose **Open**. 4. To confirm that Data File Tools is using EBCDIC, confirm that the drop-	App developer

Task	Description	Skills required
	down menu in the **Debug File** section is set to **EBCDIC**.	
Create the default record layout.	Use the default record layout for all records that do not match any conditional layouts. 1. In the **Available Layouts** section on the left pane, expand the data structure , and then locate the 01 level used for the default layout. 2. Right-click the 01 item, and then choose **Create Default Layout**. The default layout appears in the **Layouts** pane and can be identified by the blue "D" icon.	App developer

Task	Description	Skills required
Create a conditional record layout.	Use the conditional record layout when there is more than one record layout in a file. 1. In the **Selected Layouts** section on the right pane, expand the data structure , and then locate the 01 level used for the condition al layout. 2. Right-click the 01 item, and then choose **Create Conditional Layout**. The conditional layout appears in the **Layouts** pane on the right side and can be identified by the green "C" icon. 3. Expand the conditional layout, right-click the field where you must place a condition, and then choose **Properties**. 4. In the **Field Propertie s** dialog box, enter the condition. Confirm that the character set is set to **EBCDIC**, and then choose **OK**. A red "IF" icon appears next to the field that has a condition set.	App developer

Task	Description	Skills required
	5. Repeat steps 3–4 for any other fields that require conditions for this layout. 6. Repeat steps 1–4 for any other conditional layouts that must be added. 7. Choose **File**, choose **Save As**, and then save the structure file to disk.	

(Option A) Test the structure file using Classic Data File Tools

Task	Description	Skills required
Test an EBCDIC data file.	Confirm that you can use your structure file to view an EBCDIC test data file correctly . 1. Choose the Windows Start menu icon, find and choose **Micro Focus Enterpris e Developer**, and then choose **Classic Data Tools**. 2. Choose **File**, and then choose **Open**. 3. In the **Open** dialog box, for **File name**, select the EBCDIC dataset, and then choose **Open**. 4. Choose **File**, **Data File Editor**, **Load Record Layouts**.	App developer

Task	Description	Skills required
	5. In the **Open** dialog box, for **File name**, select the structure file, and then choose **Open**. 6. To confirm that the character set mode is set to EBCDIC, confirm that the drop-down menu is set to **EBCDIC**. You can see the raw record data in the left pane and the formatted data in the right pane. 7. Choose various records to ensure that all formats are rendered with the correct layout.	

(Option B) Test the structure file using Data File Tools

Task	Description	Skills required
Test an EBCDIC data file.	Confirm that you can use your structure file to view an EBCDIC test data file correctly . 1. Choose the Windows Start menu icon, find and select **Micro Focus Enterprise Developer**, and then choose **Data File Tools**. 2. Choose **File**, **Open**, **Data File**.	App developer

Task	Description	Skills required
	3. In the **Open Data File** dialog box, on the **Local** tab, for **Filename**, choose **Browse** to find the location of the EBCDIC test file. 4. For **Structure File (optional)**, choose **Browse** to find the location of the structure file. 5. In the **File Details** section, enter the details of the file and confirm that **Encoding** is set to **EBCDIC**. 6. Choose either **Open Shared** or **Open Exclusive** mode depending on your requirements. 7. Confirm that the drop-down menu in the **Appearance** section of the toolbar is set to **EBCDIC**. You will see the raw record data in the left pane and the formatted data in the right pane. 8. Choose various records to ensure that all formats are rendered with the correct layout.	

Test data file conversion

Task	Description	Skills required
Test the conversion of an EBCDIC file.	1. Choose the Windows Start menu icon, find and select **Micro Focus Enterpris e Developer**, and then choose **Classic Data Tools**. 2. Choose **Tools**, and then choose **Convert**. 3. In the **Data File Convert** dialog box, in the **Input file** section, for **Filename**, choose **Browse** to find and select the EBCDIC input file. Confirm that **Character set** is set to **EBCDIC**. 4. In the **Character Set Conversion** section, select the **Convert character set** and **Records contain non-text data items** check boxes. Choose **Select layout for conversion**, and then choose **Browse** to find and select the structure file. 5. In the **New file** section, for **Filename**, enter the path and file name of the ASCII output file that you want to create. By default, the conversion tool defaults to the same format as the	App developer

Task	Description	Skills required
	input file. For testing, leave the options set to their default values. 6. Choose **Convert**. 7. Follow the steps in either the *(Option A) Test the structure file using Classic Data File Tools* or *(Option B) Test the structure file using Data File Tools* section, but load the ASCII output file instead of the EBCDIC file. 8. Load both the EBCDIC and ASCII files into the Data File Editor, and then compare the files side by side to check the accuracy of the conversion.	

Related resources

- Micro Focus (Micro Focus documentation)
- Mainframe and legacy code (AWS Blog posts)
- AWS Prescriptive Guidance (AWS documentation)
- AWS Documentation (AWS documentation)
- AWS General Reference (AWS documentation)
- AWS glossary (AWS documentation)

Deploy an environment for containerized Blu Age applications by using Terraform

Richard Milner-Watts, Amazon Web Services

Summary

Migrating legacy mainframe workloads into modern cloud architectures can eliminate the costs of maintaining a mainframe—costs that only increase as the environment ages. However, migrating jobs from a mainframe can pose unique challenges. Internal resources might not be familiar with the job logic, and the high performance of mainframes at these specialized tasks can be difficult to replicate when compared to commodity, generalized CPUs. Rewriting these jobs can be a large undertaking and require significant effort.

Blu Age converts legacy mainframe workloads into modern Java code, which you can then run as a container.

This pattern provides a sample serverless architecture for running a containerized application that has been modernized with the Blu Age tool. The included HashiCorp Terraform files will build a secure architecture for the orchestration of Blu Age containers, supporting both batch tasks and real-time services.

For more information about modernizing your workloads by using Blu Age and AWS services, see these AWS Prescriptive Guidance publications:

- Running mainframe workloads that have been modernized with Blu Age on AWS serverless infrastructure
- Containerize mainframe workloads that have been modernized by Blu Age

For assistance with using Blu Age to modernize your mainframe workloads, contact the Blu Age team by choosing **Contact our experts** on the Blu Age website. For assistance with migrating your modernized workloads to AWS, integrating them with AWS services, and moving them into production, contact your AWS account manager or fill out the AWS Professional Services form.

Prerequisites and limitations

Prerequisites

- The sample containerized Blu Age application provided by the [Containerize mainframe workloads that have been modernized by Blu Age] pattern. The sample application provides the logic to handle the processing of input and output for the modernized application, and it can integrate with this architecture.
- Terraform is required to deploy these resources.

Limitations

- Amazon Elastic Container Service (Amazon ECS) places limits on the task resources that can be made available to the container. These resources include CPU, RAM, and storage. For example, when using Amazon ECS with AWS Fargate, the [task resource limits apply].

Product versions

This solution was tested with the following versions:

- Terraform 1.3.6
- Terraform AWS Provider 4.46.0

Architecture

Source technology stack

- Blu Age
- Terraform

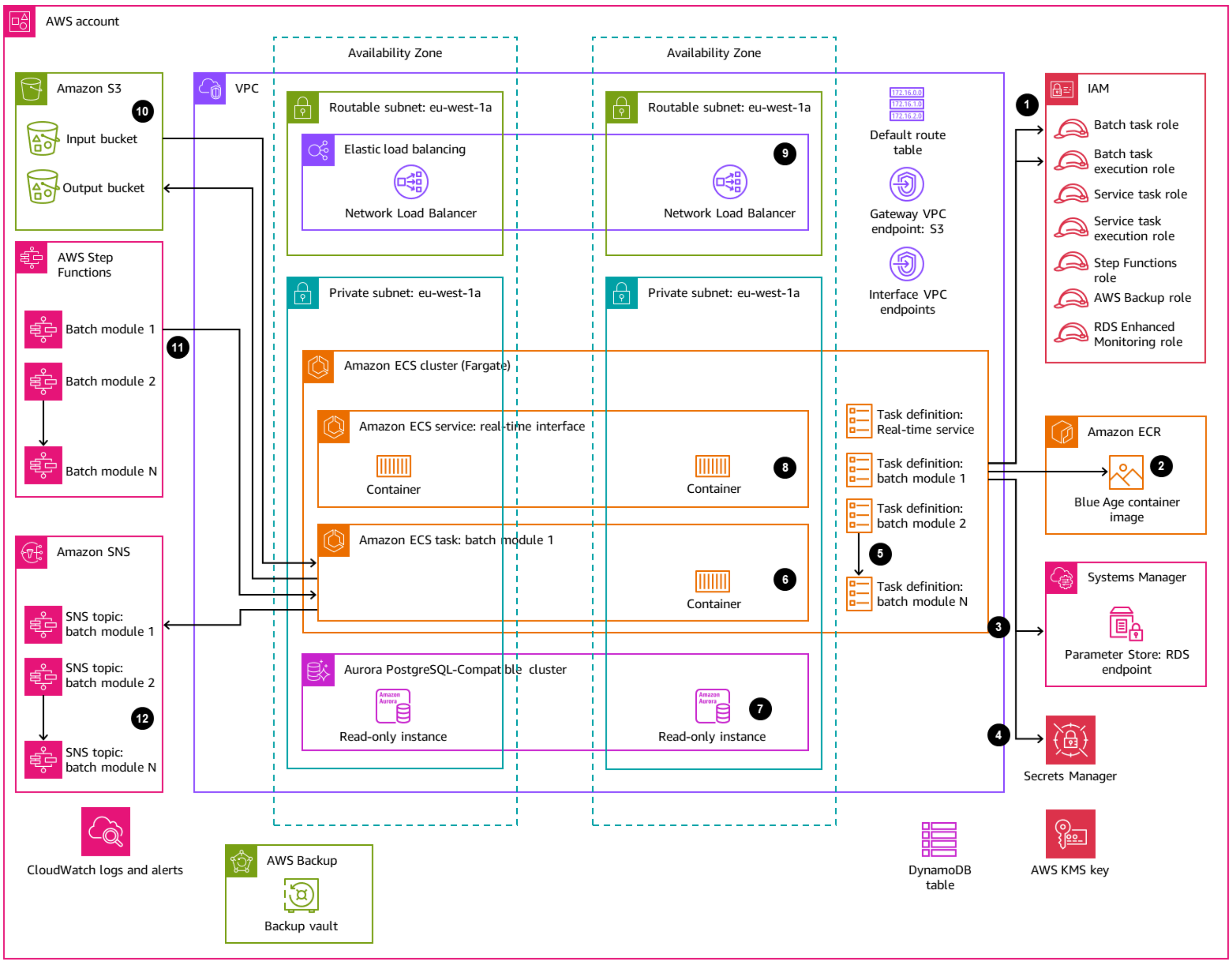
Target technology stack

- Amazon Aurora PostgreSQL-Compatible Edition
- AWS Backup
- Amazon Elastic Container Registry (Amazon ECR)
- Amazon ECS
- AWS Identity and Access Management Service (IAM)
- AWS Key Management Server (AWS KMS)
- AWS Secrets Manager

- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Storage Service (Amazon S3)
- AWS Step Functions
- AWS Systems Manager

Target architecture

The following diagram shows the solution architecture.



1. The solution deploys the following IAM roles:
   - Batch task role
   - Batch task execution role

- Service task role
- Service task execution role
- Step Functions role
- AWS Backup role
- RDS Enhanced Monitoring role.

The roles conform to least-privileged access principles.

2. Amazon ECR is used to store the container image that is orchestrated by this pattern.
3. AWS Systems Manager Parameter Store provides configuration data about each environment to the Amazon ECS task definition at runtime.
4. AWS Secrets Manager provides sensitive configuration data about the environment to the Amazon ECS task definition at runtime. The data has been encrypted by AWS KMS.
5. The Terraform modules create Amazon ECS task definitions for all real-time and batch tasks.
6. Amazon ECS runs a batch task by using AWS Fargate as the compute engine. This is a short-lived task, initiated as required by AWS Step Functions.
7. Amazon Aurora PostgreSQL-Compatible provides a database to support the modernized application. This replaces mainframe databases such as IBM Db2 or IBM IMS DB.
8. Amazon ECS runs a long-lived service to deliver a modernized real-time workload. These stateless applications run permanently with containers spread across Availability Zones.
9. A Network Load Balancer is used to grant access to the real-time workload. The Network Load Balancer supports earlier protocols, such as IBM CICS. Alternatively, you can use an Application Load Balancer with HTTP-based workloads.
10. Amazon S3 provides object storage for job inputs and outputs. The container should handle pull and push operations into Amazon S3 to prepare the working directory for the Blu Age application.
11. The AWS Step Functions service is used to orchestrate running the Amazon ECS tasks to process batch workloads.
12. SNS topics for each batch workload are used to integrate the modernized application with other systems, such as email, or to initiate additional actions, such as delivering output objects from Amazon S3 into FTP.

Note

By default, the solution has no access to the internet. This pattern assumes that the virtual private cloud (VPC) will be connected to other networks using a service such as AWS Transit Gateway. As such, multiple interface VPC endpoints are deployed to grant access to the AWS services used by the solution. To turn on direct internet access, you can use the toggle in the Terraform module to replace the VPC endpoints with an internet gateway and the associated resources.

Automation and scale

The use of serverless resources throughout this pattern helps to ensure that, by scaling out, there are few limits on the scale of this design. This reduces *noisy neighbor concerns*, such as the competition for compute resources that might be experienced on the original mainframe. Batch tasks can be scheduled to run simultaneously as needed.

Individual containers are limited by the maximum sizes supported by Fargate. For more information, see the *Task CPU and memory* section in the Amazon ECS documentation.

To scale real-time workloads horizontally, you can add containers.

Tools

AWS services

- Amazon Aurora PostgreSQL-Compatible Edition is a fully managed, ACID-compliant relational database engine that helps you set up, operate, and scale PostgreSQL deployments.
- AWS Backup is a fully managed service that helps you centralize and automate data protection across AWS services, in the cloud, and on premises.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.

- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management.

Other services

- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources. This pattern uses Terraform to create the sample architecture.

Code repository

The source code for this pattern is available in the GitHub Blu Age Sample ECS Infrastructure (Terraform) repository.

Best practices

- For test environments, use features such as the `forceDate` option to configure the modernized application to generate consistent test results by always running for a known time period.
- Tune each task individually to consume the optimal amount of resources. You can use Amazon CloudWatch Container Insights to obtain guidance on potential bottlenecks.

Epics

Prepare the environment for deployment

Task	Description	Skills required
Clone the solution source code.	Clone the solution code from the [GitHub project]().	DevOps engineer
Bootstrap the environment by deploying resources to store the Terraform state.	1. Open a terminal window, and confirm that Terraform is installed and that AWS credentials are available. 2. Navigate to the `bootstrap-terraform` folder. 3. Edit the file `main.tf` if you want to change the names of the S3 bucket (`<accountId>-terraform-backend`) and Amazon DynamoDB table (`terraform-lock`). 4. Run the `terraform apply` command to deploy the resources. Make a note of the S3 bucket and DynamoDB table names.	DevOps engineer

Deploy the solution infrastructure

Task	Description	Skills required
Review and update the Terraform configuration.	In the root directory, open the file `main.tf`, review the contents, and consider	DevOps engineer

Task	Description	Skills required
	making the following updates: 1. Update the AWS Region by searching for and replacing the string `eu-west-1` with the desired Region that you want to use. 2. Update the bucket name in the `Terraform Backend` block if the default was altered in the previous epic. 3. Update the `dynamodb_table` value if the default was altered in the previous epic. 4. Update the value of the `stack_prefix` variable to the string that you want. This string will be prepended to the names of all resources created by this pattern. 5. Update the value of the `vpc_cidr` This should be at least a `/24` address range. 6. Review the `Locals` section. This is used to define the Blu Age tasks that will be deployed. The solution will iterate over the list object `bluage_ba`	

Task	Description	Skills required
	`tch_modules` , creating the associated resources (Step Functions state machine, task definition, and SNS topic) for each element of the list. In some cases, you might want to adjust variables for different environments. For example, to force the runtime in test environments, you can change value of the `force_execution_time` variable. 7. To turn on internet access, change the value for `direct_internet_access_required` from `false` to `true`. This will deploy an internet gateway, along with the NAT gateways and route tables that turn on public internet access for the infrastructure. By default, the solution will deploy interface VPC endpoints into a VPC without direct internet access. 8. To grant access to any client-server workloads that are served through Elastic Load Balancing , update the values of	

Task	Description	Skills required
	`additional_nlb_ingress_cidrs` with the CIDR networks that should be allowed.	
Deploy the Terraform file.	From your terminal, run the `terraform apply` command to deploy all resources. Review the changes generated by Terraform, and enter **yes** to initiate the build. Note that it can take over 15 minutes to deploy this infrastructure.	DevOps engineer

(Optional) Deploy a valid Blu Age containerized application

Task	Description	Skills required
Push the Blu Age container image to Amazon ECR.	Push the container into the Amazon ECR repository that you created in the previous epic. For instructions, see the [Amazon ECR documentation](https://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-push-ecr-image.html). Make a note of the container image URI.	DevOps engineer
Update the Terraform to reference the Blu Age container image.	Update the file `main.tf` to reference the container image that you uploaded.	DevOps engineer
Redeploy the Terraform file.	From your terminal, run `terraform apply` to deploy all resources. Review the	DevOps engineer

Task	Description	Skills required
	suggested updates from Terraform, and then enter **yes** to proceed with the deployment.	

Related resources

- Blu Age
- Running mainframe workloads that have been modernized with Blu Age on AWS serverless infrastructure
- Containerize mainframe workloads that have been modernized by Blu Age

Generate Db2 z/OS data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight

Shubham Roy, Roshna Razack, and Santosh Kumar Singh, Amazon Web Services

Summary

If your organization is hosting business-critical data in an IBM Db2 mainframe environment, gaining insights from that data is crucial for driving growth and innovation. By unlocking mainframe data, you can build faster, secure, and scalable business intelligence to accelerate data-driven decision-making, growth, and innovation in the Amazon Web Services (AWS) Cloud.

This pattern presents a solution for generating business insights and creating sharable narratives from mainframe data in IBM Db2 for z/OS tables. Mainframe data changes are streamed to Amazon Managed Streaming for Apache Kafka (Amazon MSK) topic using AWS Mainframe Modernization Data Replication with Precisely. Using Amazon Redshift streaming ingestion, Amazon MSK topic data is stored in Amazon Redshift Serverless data warehouse tables for analytics in Amazon QuickSight.

After the data is available in QuickSight, you can use natural language prompts with Amazon Q in QuickSight to create summaries of the data, ask questions, and generate data stories. You don't have to write SQL queries or learn a business intelligence (BI) tool.

Business context

This pattern presents a solution for mainframe data analytics and data insights use cases. Using the pattern, you build a visual dashboard for your company's data. To demonstrate the solution, this pattern uses a health care company that provides medical, dental, and vision plans to its members in the US. In this example, member demographics and plan information are stored in the IBM Db2 for z/OS data tables. The visual dashboard shows the following:

- Member distribution by region
- Member distribution by gender
- Member distribution by age
- Member distribution by plan type
- Members who have not completed preventive immunization

For examples of member distribution by region and members who have not completed preventive immunization, see the Additional information section.

After you create the dashboard, you generate a data story that explains the insights from the previous analysis. The data story provides recommendations for increasing the number of members who have completed preventive immunizations.

Prerequisites and limitations

Prerequisites

- An active AWS account. This solution was built and tested on Amazon Linux 2 on Amazon Elastic Compute Cloud (Amazon EC2).
- An virtual private cloud (VPC) with a subnet that can be accessed by your mainframe system.
- A mainframe database with business data. For the example data used to build and test this solution, see the *Attachments* section.
- Change data capture (CDC) enabled on the Db2 z/OS tables. To enable CDC on Db2 z/OS, see the IBM documentation.
- Precisely Connect CDC for z/OS installed on the z/OS system that's hosting the source databases. The Precisely Connect CDC for z/OS image is provided as a zip file within the AWS Mainframe Modernization - Data Replication for IBM z/OS Amazon Machine Image (AMI). To install Precisely Connect CDC for z/OS on the mainframe, see the Precisely installation documentation.

Limitations

- Your mainframe Db2 data should be in a data type that's supported by Precisely Connect CDC. For a list of supported data types, see the Precisely Connect CDC documentation.
- Your data at Amazon MSK should be in a data type that's supported by Amazon Redshift. For a list of supported data types, see the Amazon Redshift documentation.
- Amazon Redshift has different behaviors and size limits for different data types. For more information, see the Amazon Redshift documentation.
- The near real-time data in QuickSight depends on the refresh interval set for the Amazon Redshift database.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. Amazon Q in QuickSight is currently not available in every Region that supports

QuickSight. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

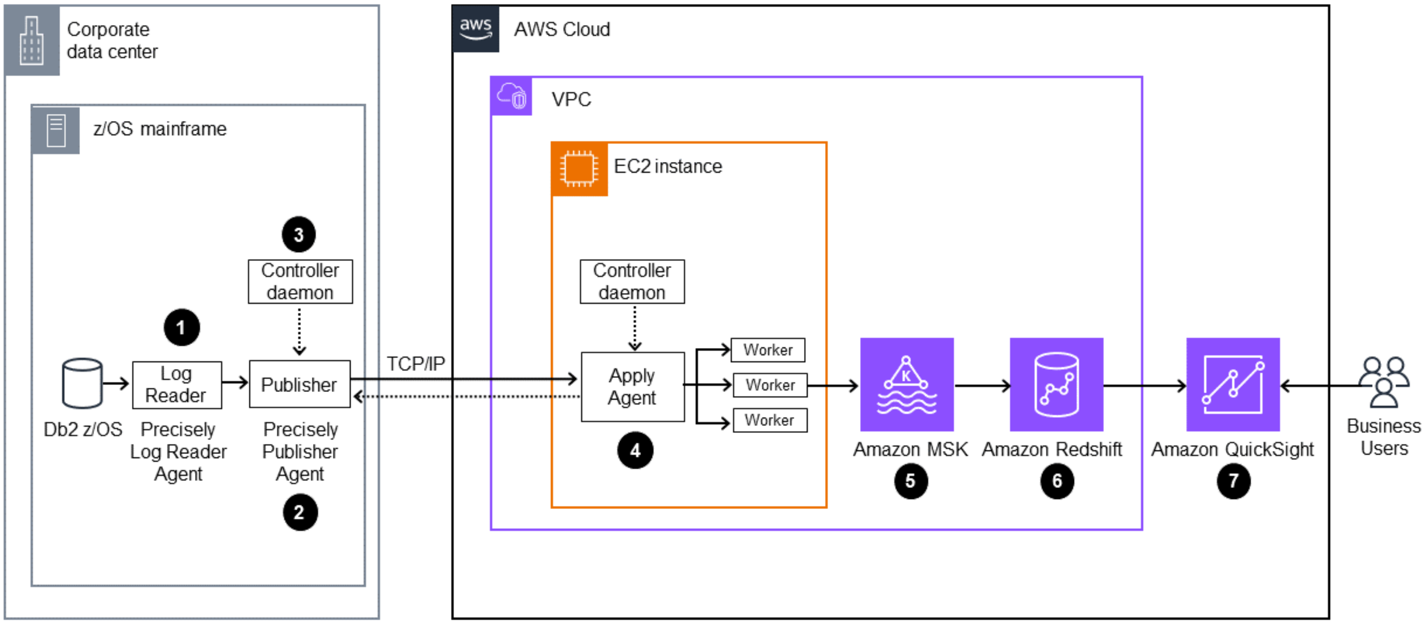
Product versions

- AWS Mainframe Modernization Data Replication with Precisely version 4.1.44
- Python version 3.6 or later
- Apache Kafka version 3.5.1

Architecture

Target architecture

The following diagram shows an architecture for generating business insights from mainframe data by using AWS Mainframe Modernization Data Replication with Precisely and Amazon Q in QuickSight.



The diagram shows the following workflow:

1. The Precisely Log Reader Agent reads data from Db2 logs and writes the data into transient storage on an OMVS file system on the mainframe.
2. The Publisher Agent reads the raw Db2 logs from transient storage.

3. The on-premises controller daemon authenticates, authorizes, monitors, and manages operations.
4. The Apply Agent is deployed on Amazon EC2 by using the preconfigured AMI. It connects with the Publisher Agent through the controller daemon by using TCP/IP. The Apply Agent pushes data to Amazon MSK using multiple workers for high-throughput.
5. The workers write the data to the Amazon MSK topic in JSON format. As the intermediate target for the replicated messages, Amazon MSK provides the highly available and automated failover capabilities.
6. Amazon Redshift streaming ingestion provides low-latency, high-speed data ingestion from Amazon MSK to an Amazon Redshift Serverless database. A stored procedure in Amazon Redshift performs the mainframe change data (insert/update/deletes) reconciliation into Amazon Redshift tables. These Amazon Redshift tables serves as the data analytics source for QuickSight.
7. Users access the data in QuickSight for analytics and insights. You can use Amazon Q in QuickSight to interact with the data by using natural language prompts.

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them out or in.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that helps you build and run applications that use Apache Kafka to process streaming data.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data in a single dashboard. This pattern uses the generative BI capabilities of Amazon Q in QuickSight.
- Amazon Redshift Serverless is a serverless option of Amazon Redshift that makes it more efficient to run and scale analytics in seconds without the need to set up and manage data warehouse infrastructure.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.

Other tools

- Precisely Connect CDC collects and integrates data from legacy systems into cloud and data platforms.

Code repository

The code for this pattern is available in the GitHub Mainframe_DataInsights_change_data_reconciliation repository. The code is a stored procedure in Amazon Redshift. This stored procedure reconciles mainframe data changes (inserts, updates, and deletes) from Amazon MSK into the Amazon Redshift tables. These Amazon Redshift tables serve as the data analytics source for QuickSight.

Best practices

- Follow best practices while setting up your Amazon MSK cluster.
- Follow Amazon Redshift data parsing best practices for improving performance.
- When you create the AWS Identity and Access Management (IAM) roles for the Precisely setup, follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.

Epics

Set up AWS Mainframe Modernization Data Replication with Precisely on Amazon EC2

Task	Description	Skills required
Set up a security group.	To connect to the controlle r daemon and the Amazon MSK cluster, create a security group for the EC2 instance. Add the following inbound and outbound rules: • **Inbound rule 1:**	DevOps engineer, AWS DevOps

Task	Description	Skills required
	- For **Type**,choose **Custom TCP**. - For **Protocol**, choose **TCP**. - For **Port range**, choose **2626** (default port for the Precisely controlle r daemon) or the port number of the controlle r daemon running on the mainframe. - For **Source**, choose CIDR block. **Inbound rule 2:** - For **Type**, choose **Custom TCP**. - For **Protocol**, choose **SSH**. - For **Port range**, choose **22**. - For **Source**, choose **IP address** or **prefix list**. **Inbound rule 3:** - For **Type**, choose **Custom TCP**. - For **Protocol**, choose **TCP**. - For **Port range**, choose **9092-9098**. - For **Source**, choose **CIDR block**. **Outbound rule 1:**	

Task	Description	Skills required
	- For **Type**, choose **Custom TCP**. - For **Protocol**, choose **TCP**. - For **Port range**, choose **9092-9098**. - For **Source**, choose **CIDR block**. • **Outbound rule 2:** - For **Type**, choose **Custom TCP**. - For **Protocol**, choose **TCP**. - For **Port range**, choose **2626** (default port for the Precisely controlle r daemon) or the port number of the controlle r daemon running on the mainframe. - For **Source**, choose **CIDR block**. Note the name of the security group. You will need to reference the name when you launch the EC2 instance and configure the Amazon MSK cluster.	

Task	Description	Skills required
Create an IAM policy and an IAM role.	1. To create an IAM policy and an IAM role, follow the instructions in the [AWS documentation](). The IAM policy grants access to create topics on the Amazon MSK cluster and to send data to those topics. 2. After you create the IAM role, associate the policy with it. Note the IAM role name. This role will be used as the IAM instance profile when you launch the EC2 instance.	DevOps engineer, AWS systems administrator

Task	Description	Skills required
Provision an EC2 instance.	To provision an EC2 instance to run Precisely CDC and connect to Amazon MSK, do the following: 1. Sign in to AWS Marketplace, and subscribe to AWS Mainframe Modernization – Data Replication for IBM z/OS. 2. Select the AMI from the managed subscriptions, and choose on **Launch new instance.** 3. Provide other configuration details, such as the instance name, instance type, key pair, VPC, and subnets. For more information, see the Amazon EC2 documentation. 4. In the dropdown list, choose the security group that you created earlier. 5. Under **Advanced details**, **IAM instance profile**, you must select the role that you created earlier. 6. Choose **Launch instance**.	AWS administrator, DevOps engineer

Set up Amazon MSK

Task	Description	Skills required
Create the Amazon MSK cluster.	To create an Amazon MSK cluster, do the following : 1. Sign in to the AWS Management Console, and open the Amazon MSK console at https://console.aws.amazon.com/msk/. 2. Choose **Create cluster**. 3. For the cluster creation method, choose **Custom create**, and for the cluster type, choose **Provisioned**. 4. Provide a name for the cluster. 5. Update **Cluster settings** as needed, and keep the defaults for the other settings. 6. Note the **<Kafka version>**. You will need it during Kafka client setup. 7. Choose **Next**. 8. Choose the same VPC and subnets that you used for the Precisely EC2 instance, and choose the security group that you created earlier.	AWS DevOps, Cloud administrator

Task	Description	Skills required
	9. In the **Security settings** section, enable both **SASL/ SCRAM** and **IAM role-base d authentication**. Precisely Connect CDC uses SASL/ SCRAM (Simple Authentic ation and Security Layer/ Salted Challenge Response Mechanism), and IAM is needed for connecting to Amazon Redshift. 10. Choose **Next**. 11. For reviewing, choose **Monitoring** and **Broker log delivery method.** 12. Choose **Next**, and then choose **Create the cluster**. A typical provisioned cluster takes up to 15 minutes to create. After the cluster is created, its status changes from **Creating** to **Active**.	

Task	Description	Skills required
Set up SASL/SCRAM authentication.	To set up SASL/SCRAM authentication for an Amazon MSK cluster, do the following: 1. To set up a secret in Secrets Manager, follow the instructions in the AWS documentation. 2. Open the Amazon MSK console, and select the Amazon MSK cluster that you created previously. 3. Choose the **Properties** tab. 4. Choose **Associate secrets**, **Choose the secrets**, select the secrets key that you created, and then choose **Associate secrets**. You will see a success message similar to the following: `Successfully associated 1 secret for cluster <chosen cluster name>` 5. Choose the cluster name. 6. In the cluster summary, choose **View client information**. 7. Note the private endpoint connection string for the	Cloud architect

Task	Description	Skills required
	authentication type SASL/ SCRAM.	

Task	Description	Skills required

Create the Amazon MSK topic.	To create the Amazon MSK topic, do the following:
	
	1. Connect to the EC2 instance that you created earlier, and install the latest updates by running the following command:
	`sudo yum update -y`
	2. Install the Java and Kafka library by running the following command:
	`sudo yum install -y java-11 librdkafka librdkafka-devel`
	3. To create a folder named `kafka` in `/home/ec2-user`, navigate to that folder, and run the following command:
	`mkdir kafka;cd kafka`
	4. Download the `kafka` client library to the `kafka` folder, replacing `<YOUR MSK VERSION>` with the Kafka version that you noted during the Amazon MSK cluster creation:
	`wget https://archive.apache.org/`
		Cloud administrator

Task	Description	Skills required
	``` dist/kafka//kafka_ 2.13-<YOUR MSK VERSION>.tgz ``` 5. To extract the downloaded file, run the following command, replacing `YOUR MSK VERSION>`: ``` tar -xzf kafka_2.13- <YOUR MSK VERSION>. tgz ``` 6. To navigate to the `kafka libs` directory and download the Java IAM authentication Java Archive (JAR) file, run the following commands, replacing `<YOUR MSK VERSION>`: ``` cd kafka_2.13-<YOUR MSK VERSION>/libs wget https://g ithub.com/aws/aws- msk-iam-auth/relea ses/download/v1.1. 1/aws-msk-iam-auth -1.1.1-all.jarkafka ``` 7. To navigate to the Kafka `bin` directory and create the `client.properties` file, run the following commands:	

Task	Description	Skills required

	`cd /home/ec2-user/kafka/kafka_2.13-<YOUR MSK VERSION>/bin`
	`cat >client.properties`
	
	8. Update the `client.properties` file with the following contents:
	
	`security.protocol=SASL_SSL`
	`sasl.mechanism=AWS_MSK_IAM`
	`sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;`
	`sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler`
	
	9. To create a Kafka topic, navigate to the Kafka bin and run the following command, replacing `<kafka broker>` with the IAM bootstrap server private endpoint that you noted during Amazon MSK cluster creation:
	
	`./kafka-topics.sh --bootstrap-server <kafka broker> --command-config`	

Task	Description	Skills required
	`client.properties --create --replication-factor 3 —partitions 6 --topic <topic name>` 10. When the message `Created topic <topic name>` appears, note the topic name.	

Configure the Precisely Apply Engine on Amazon EC2

Task	Description	Skills required
Set up the Precisely scripts to replicate data changes.	To set up the Precisely Connect CDC scripts to replicate changed data from the mainframe to the Amazon MSK topic, do the following: 1. To create a folder name precisely and change to that folder, run the following command: `mkdir /home/ec2-user/precisely;cd /home/ec2-user/precisely` 2. To create two folders inside precisely called `scripts` and `ddls`, and then change to the `scripts` folder, run the following command:	App developer, Cloud architect

Task	Description	Skills required
	``` mkdir scripts;mkdir ddls;cd scripts ``` 3. To create a file named `sqdata_kafka_producer.conf` in the `scripts` folder, run the following command: ``` cat >sqdata_k afka_producer.conf ``` 4. Update `sqdata_kafka_producer.conf` file with the following content: ``` builtin.features=S ASL_SCRAM security.protocol =SASL_SSL sasl.mechanism=SC RAM-SHA-512 sasl.username=<User Name> sasl.passwor d=<Password> metadata.broke r.list=<SASL/SCRAM Bootstrap servers> ``` **Note** Update `<SASL/SCRAM Bootstrap servers>` with the Amazon MSK	

Task	Description	Skills required

	SASL/SCRAM broker list that you previously configured. Update `<User Name>` and `<Password>` with the username and password that you set up previously in Secrets Manager.
	
	5. Create a `script.sqd` file in the `scripts` folder.
	
	`cat >script.sqd`
	
	The Apply Engine uses `script.sqd` to process source data and replicate the source data to the target. For an example Apply Engine script, see the Additional information section.
	
	6. To change to the `ddls` folder and create a .ddl file for each Db2 table, run the following commands:
	
	`cd /home/ec2-user/precisely/ddls`
	`cat >mem_details.ddl`
	`cat >mem_plans.ddl`	

Task	Description	Skills required
	For example .ddl files, see the Additional information section.	

Task	Description	Skills required

Generate the network ACL key.	To generate the network access control list (network ACL) key, do the following:
	1. To export the `sqdata` installation path, run the following command:
	`export PATH=$PATH:/usr/sbin:/opt/precisely/di/sqdata/bin`
	2. To change to the `/home/ec2-user` directory and generate the network ACL key, run the following commands:
	`cd /home/ec2-user`
	`sqdutil keygen --force`
	After the public and private keys are generated, the following message is displayed:
	`SQDUT04I Generating a private key in file /home/ec2-user/.nacl/id_nacl`
	`SQDC017I sqdutil(pid=27344) terminated successfully`
		Cloud architect, AWS DevOps

Task	Description	Skills required
	3. Note the generated public key that is stored in the `.nacl` folder.	

Prepare the mainframe source environment

Task	Description	Skills required
Configure defaults in the ISPF screen.	To configure default settings in the Interactive System Productivity Facility (ISPF), follow the instructions in the [Precisely documentation]().	Mainframe system administr ator
Configure the controller daemon.	To configure the controller daemon, do the following: 1. On the **SQData z/OS Main Menu** screen, choose option 2. 2. On the **Add Daemon to List** screen, in the **Daemon Name** field, enter a name for the daemon, and then press **Enter**.	Mainframe system administr ator
Configure the publisher.	To configure the publisher, do the following: 1. On the **SQData z/OS Main Menu** screen, choose option 3. This takes you to the **Capture/Publisher Summary** screen.	Mainframe system administr ator

Task	Description	Skills required
	2. Choose the option to add a CAB file. This will take you to the **Add CAB File to List** screen. 3. In the **Name** field, enter a name for the CAB file. For **Db2**, enter the **Type** as D. 4. Press **Enter**. This takes you to the **Create New Db2 Capture CAB File** screen. 5. In the **zFS Dir** field, specify the storage mount point. 6. Press **Enter** to save and continue.	

Task	Description	Skills required

Update the daemon configuration file.	To update the publisher details in the controller daemon configuration file, do the following:
	
	1. On the **SQData z/OS Main Menu** screen, choose option 2.
	2. Enter S near the daemon that you created to see the daemon details.
	3. Enter 1, and then press **Enter** to edit the agents file.
	4. Add your CAB file details. The following example shows the details for a CAB file named `DB2ZTOMSK` . Use your mainframe user ID instead of `<userid>`.
	
	`ÝDB2ZTOMSK¨`
	`type=capture`
	`cab=/u/<userid>/sqdata/DB2ZTOMSK.cab`
	
	5. Press **F3**.
	6. Enter 2 to edit the ACL file. Add your `userid` to the `acl` configuration file as shown in the following example:
	
	`Ýacls¨`
	`prod=admin,<userid>`
		Mainframe system administrator

Task	Description	Skills required
	7. Press **F3** to save and exit.	
Create the job to start the controller daemon.	To create the job, do the following: 1. In **Options**, enter `G`. 2. Enter the JOB Card, job and proc libraries, and `Db2 load` library details. 3. Enter the network ACL file details, and enter option 2 to generate the job control language (JCL) file in the specified job library.	Mainframe system administr ator

Task	Description	Skills required
Generate the capture publisher JCL file.	To generation the capture publisher JCL file, do the following: 1. On the **SQData z/OS Main Menu** screen, choose option 3. This takes you to the **Capture/Publisher Summary** screen. 2. Enter S next to the CAB file to select it. This takes you to the **Db2 Capture/Publisher Detail** screen. 3. In **Options**, enter G in options to generate the `capture/publisher` job. 4. Enter the JOB card, job, and procedure libraries , and Db2 load library details. 5. To create the job, choose option 4. The job is created in the job library specified in job library.	Mainframe system administr ator

Task	Description	Skills required

Check and update CDC.	1. Check the `DATACAPTURE` flag for the Db2 table by running the following query, changing `<table name>` to your Db2 table name:
	`SELECT DATACAPTURE FROM SYSIBM.SYSTABLES WHERE NAME='<table name>';`
	Confirm that the result shows `DATACAPTURE` as Y.
	2. If `DATACAPTURE` is not Y, run the following query to enable CDC on the Db2 table, changing `<table name>` to your Db2 table name:
	`ALTER TABLE <table name> DATA CAPTURE CHANGES;`
		Mainframe system administrator

Task	Description	Skills required
Submit the JCL files.	Submit the following JCL files that you configured in the previous steps: • JCL file to start the controller daemon • JCL file to start capture and publish After you submit the JCL files, you can start the Apply Engine in Precisely on the EC2 instance.	Mainframe system administr ator

Run and validate CDC

Task	Description	Skills required
Start the Apply Engine and validate the CDC.	To start the Apply Engine on the EC2 instance and validate the CDC, do the following: 1. To connect to the EC2 instance, follow the instructions in the AWS documentation. 2. Change to the directory that contains the `script.sqd` file: `cd /home/ec2-user/pre cisely/scripts`	Cloud architect, App developer

Task	Description	Skills required
	3. To start the Apply Engine, run the following `sqdeng` start command: `sqdeng -s script.sqd --identity=/home/ec2-user/.nacl/id_nacl` The Apply Engine will start waiting for updates from the mainframe source. 4. To test CDC, make some record inserts or updates in the Db2 table. 5. Validate that the Apply Engine log shows the number of records captured and written to the target.	

Task	Description	Skills required

Validate the records on the Amazon MSK topic.	To read the message from the Kafka topic, do the following:
	
	1. To change to the `bin` directory of the Kafka client installation path on the EC2 instance, run the following command, replacing `<Kafka version>` with your version:
	`cd /home/ec2-user/kafka/kafka_2.13-<Kafka version>/bin`
	2. To validate the Db2 CDC written as messages in the Kafka topic, run the following command, replacing `<kafka broker>` and `<Topic Name>` with the topic that you created earlier:
	`./kafka-console-consumer.sh --bootstrap-server <kafka broker>:9098 --topic <Topic Name> --from-beginning --consumer.config client.properties`
	3. Validate that the messages match the number of
		App developer, Cloud architect

Task	Description	Skills required
	records updated in the Db2 table.	

Store mainframe change data in an Amazon Redshift Serverless data warehouse

Task	Description	Skills required
Set up Amazon Redshift Serverless.	To create an Amazon Redshift Serverless data warehouse, follow the instructions in the [AWS documentation](). On the Amazon Redshift Serverless dashboard, validate that the namespace and workgroup were created and are available. For this example pattern, the process might take 2‒5 minutes.	Data engineer
Set up the IAM role and trust policy required for streaming ingestion.	To set up Amazon Redshift Serverless streaming ingestion from Amazon MSK, do following: 1. Create an IAM policy for Amazon Redshift to access Amazon MSK. Replacing `[region]` with the AWS Region for Amazon MSK, `[account-id]` with your AWS account ID, and `[msk-cluster-name]` with	Data engineer

Task	Description	Skills required
	Amazon MSK cluster name, run the following code: `{"Version": "2012-10-17","Statement": [{"Sid": "MSKIAMpolicy","Effect": "Allow","Action": ["kafka-cluster:ReadData","kafka-cluster:DescribeTopic","kafka-cluster:Connect"], "Resource": ["arn:aws:kafka:[region]:[account-id]:cluster/[msk-cluster-name]/*","arn:aws:kafka:[region]:[account-id]:topic/[msk-cluster-name]/*"]},{"Effect": "Allow","Action": ["kafka-cluster:AlterGroup","kafka-cluster:DescribeGroup"],"Resource": ["arn:aws:kafka:[region]:[account-id]:group/[msk-cluster-name]/*"]}]}` You can find the cluster name and the Amazon	

Task	Description	Skills required
	Resource Name (ARN) on the Amazon MSK console. On the console, choose **Cluster summary**, and then choose **ARN**. 2. To create an IAM role and attach the policy, follow the instructions in the AWS documentation. 3. To attach the IAM role to the Amazon Redshift Serverless namespace, do following: a. Sign in to the console, and open the Amazon Redshift console at https://console.aws.amazon.com/redshiftv2/. b. Choose the **Serverless** dashboard. c. Choose **Namespace**. d. Choose the **Security and encryption** tab. e. Choose **Permission**, and attach the IAM role that you created. 4. In your Amazon Redshift Serverless security group, create an inbound rule with the following details: • For **Type**, choose **Custom TCP**.	

Task	Description	Skills required
	• For **Protocol**, choose **TCP**. • For **Port range**, choose **9098, 9198**. • For **Source**, choose **Amazon MSK security group**. 5. In your Amazon MSK security group, create an inbound rule with following details: • For **Type**, choose **Custom TCP**. • For **Protocol**, choose **TCP**. • For **Port range**, choose **9098, 9198**. • For **Source**, choose **Amazon Redshift security group**. This pattern uses the port for IAM authentication for both Amazon Redshift and Amazon MSK configura tion. For more information, see the AWS documenta tion (step 2). 6. Turn on enhanced VPC routing for the Amazon Redshift Serverless workgroup. For more	

Task	Description	Skills required
	information, see the [AWS documentation]().	

Connect Amazon Redshift Serverless to Amazon MSK.	To connect to the Amazon MSK topic, create an external schema in Amazon Redshift Serverless. In Amazon Redshift query editor v2, run the following SQL command, replacing `'iam_role_arn'` with the role that you created previously and replacing `'MSK_cluster_arn '`with the ARN for your cluster.
	`CREATE EXTERNAL SCHEMA member_schema`
	`FROM MSK`
	`IAM_ROLE 'iam_role_arn'`
	`AUTHENTICATION iam`
	`URI 'MSK_cluster_arn';`
		Migration engineer

Task	Description	Skills required

Create a materialized view.	To consume the data from the Amazon MSK topic in Amazon Redshift Serverless, create a materialized view. In Amazon Redshift query editor v2, run the following SQL commands, replacing `<MSK_Topic_name>` with the name of your Amazon MSK topic.
	`CREATE MATERIALIZED VIEW member_view`
	`AUTO REFRESH YES`
	`AS SELECT`
	`kafka_partition,`
	`kafka_offset,`
	`refresh_time,`
	`json_parse(kafka_value) AS Data`
	`FROM member_schema.<MSK_Topic_name>`
	`WHERE CAN_JSON_PARSE(kafka_value);`
		Migration engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create target tables in Amazon Redshift.</td><td>Amazon Redshift tables provide the input for QuickSight. This pattern uses the tables <code>member_dtls</code> and <code>member_plans</code> , which match the source Db2 tables on the mainframe.

To create the two tables in Amazon Redshift, run the following SQL commands in Amazon Redshift query editor v2:

<pre>-- Table 1: members_d
tls
CREATE TABLE members_d
tls (
 memberid INT ENCODE
 AZ64,
 member_name VARCHAR(1
00) ENCODE ZSTD,
 member_type VARCHAR(5
0) ENCODE ZSTD,
 age INT ENCODE AZ64,
 gender CHAR(1) ENCODE
 BYTEDICT,
 email VARCHAR(100)
 ENCODE ZSTD,
 region VARCHAR(50)
 ENCODE ZSTD
) DISTSTYLE AUTO;

-- Table 2: member_pl
ans
CREATE TABLE member_pl
ans (
 memberid INT ENCODE
 AZ64,</pre></td><td>Migration engineer</td></tr>
</table>

Task	Description	Skills required
	``` medical_plan CHAR(1) ENCODE BYTEDICT, dental_plan CHAR(1) ENCODE BYTEDICT, vision_plan CHAR(1) ENCODE BYTEDICT, preventive_immuniz ation VARCHAR(50) ENCODE ZSTD) DISTSTYLE AUTO; ```	
Create a stored procedure in Amazon Redshift.	This pattern uses a stored procedure to sync-up change data (`INSERT`, `UPDATE`, `DELETE`) from the source mainframe to the target Amazon Redshift data warehouse table for analytics in QuickSight. To create the stored procedure in Amazon Redshift, use query editor v2 to run the stored procedure code that's in the GitHub repository.	Migration engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Read from the streaming materialized view and load to the target tables.</td><td>The stored procedure reads data change from the streaming materialized view and loads the data changes to the target tables. To run the stored procedure, use the following command:
<pre><code>call SP_Members_Load();</code></pre>
You can use Amazon EventBridge to schedule the jobs in your Amazon Redshift data warehouse to call this stored procedure based on your data latency requireme nts. EventBridge runs jobs at fixed intervals. To monitor whether the previous call to the procedure completed , you might need to use a mechanism such as an AWS Step Functions state machine. For more information, see the following resources:
Creating an Amazon EventBridge rule that runs on a schedule
Accelerate orchestration of an ELT process using AWS Step Functions and Amazon Redshift Data API.</td><td>Migration engineer</td></tr>
</table>

Task	Description	Skills required
	Another option is to use Amazon Redshift query editor v2 to schedule the refresh. For more information, see [Scheduling a query with query editor v2]().	

Connect QuickSight to data in Amazon Redshift

Task	Description	Skills required
Set up QuickSight.	To set up QuickSight, follow the instructions in the [AWS documentation]().	Migration engineer
Set up a secure connectio n between QuickSight and Amazon Redshift.	To set up secure a connectio n between QuickSight and Amazon Redshift, do the following 1. To authorize connectio ns from QuickSight to Amazon Redshift, open the Amazon Redshift console, and add an inbound rule in the Amazon Redshift security group. The rule should allow traffic to port 5439 (the default Redshift port) from the CIDR range where you set up QuickSight. For a list of AWS Regions and their IP addresses, see [Supported]()	Migration engineer

Task	Description	Skills required
	[AWS Regions for QuickSight](#). 2. On the Amazon Redshift console, choose **Workgroup**, **Data access**, **Network and security**, and enable **Publicly accessible**.	

Task	Description	Skills required
Create a dataset for QuickSight.	To create a dataset for QuickSight from Amazon Redshift, do following: 1. On the QuickSight console, in the navigation pane, choose **Datasets**. 2. On the **Datasets** page, choose **New data set**. 3. Choose **Redshift Manual connect**. 4. In the **New Redshift data source** window, enter the connection information: • For **Data source name**, enter a name for the Amazon Redshift data source. • For **Database server**, enter the endpoint of the Amazon Redshift cluster. You can get the endpoint value from the **Endpoint** field in the **General information** section for the cluster workgroup on the Amazon Redshift Serverless dashboard. The server address is the first part of the endpoint before the colon, as shown in the following example:	Migration engineer

Task	Description	Skills required
	`mfdata-insights.NNNNNNNNN.us-east-1.redshift-serverless.amazonaws.com:5439/dev` • For the port, enter 5439 (the default port for Amazon Redshift). • Enter the name of the database (after the slash in the endpoint). In this case, the database name is dev. • For **Username** and **Password**, enter the username and password for the Amazon Redshift database. 5. Choose **Validate Connection**. If successful, you should see a green check mark, which indicates validation. If validation fails, see the Troubleshooting section. 6. Choose **Create data source**.	

Task	Description	Skills required
Join the dataset.	To create analytics in QuickSight, join the two tables by following the instructions in the [AWS documentation](). In the **Join Configuration** pane, choose **Left** for **Join type**. Under **Join clauses**, use `memberid from member_plans = memberid from members_details .`	Migration engineer

Get business insights from the mainframe data by using Amazon Q in QuickSight

Task	Description	Skills required
Set up Amazon Q in QuickSight.	To set up the Amazon Q in QuickSight Generative BI capability, follow the instructions in the [AWS documentation]().	Migration engineer
Analyze mainframe data and build a visual dashboard.	To analyze and visualize your data in QuickSight, do the following: 1. To create the mainframe data analysis, follow the instructions in the [AWS documentation](). For the dataset, choose the dataset that you created.	Migration engineer

Task	Description	Skills required
	2. On the analysis page choose **Build visual**. 3. In the **Create topic for analysis** window, choose **Update existing topic**. 4. In the **Select a topic** dropdown list, choose the topic that you created previously. 5. Choose **Topic linking**. 6. After you link the topic, choose **Build visual** to open the Amazon Q **Build a Visual** window. 7. In the prompt bar. write your analysis questions . The example questions used for this pattern are the following: • Show member distribution by region • Show member distribution by age • Show member distribution by gender • Show member distribution by plan type • Show members not completed preventive immunization After you enter your questions, choose **Build**.	

Task	Description	Skills required
	Amazon Q in QuickSight creates the visuals. 8. To add the visuals to your visual dashboard, choose **ADD TO ANALYSIS**. When you're finished, you can publish your dashboard to share with others in your organization. For examples, see *Mainframe visual dashboard* in the [Additional information](#) section.	

Create a data story with Amazon Q in QuickSight from mainframe data

Task	Description	Skills required
Create a data story.	Create a data story to explain insights from the previous analysis, and generate a recommendation to increase preventive immunization for members: 1. To create the data story, follow the instructions in the [AWS documentation](#). 2. For the data story prompt, use the following: `Build a data story about Region with`	Migration engineer

Task	Description	Skills required
	`most numbers of members. Also show the member distribution by medical plan, vision plan, dental plan. Recommend how to motivate members to complete immunizat ion. Include 4 points of supportin g data for this pattern.` You can also build your own prompt to generate data stories for other business insights. 3. Choose **Add visuals**, and add the visuals that are relevant to the data story. For this pattern, use the visuals that you created previously. 4. Choose **Build**. 5. For example data story output, see *Data story output* in the [Additional information](#) section.	
View the generated data story.	To view the generated data story, choose that story on the **Data stories** page.	Migration engineer

Task	Description	Skills required
Edit a generated data story.	To change the formatting, layout, or visuals in a data story, follow the instructions in the [AWS documentation]().	Migration engineer
Share a data story.	To share a data story, follow the instructions in the [AWS documentation]().	Migration engineer

Troubleshooting

Issue	Solution
For QuickSight to Amazon Redshift dataset creation, `Validate Connection` has faled.	1. Confirm that the security group attached to the Amazon Redshift Serverless instance allows inbound traffic from the IP address range associated with the Region where you set up QuickSight. 2. Confirm that the VPC where Amazon Redshift Serverless is deployed is publicly available. 3. Confirm that you are using the correct username and password for Amazon Redshift. You can reset the username and password on the Amazon Redshift console.
Trying to start the Apply engine on the EC2 instance returns the following error: `-bash: sqdeng: command not found`	Export the `sqdata` installation path by running following command: `export PATH=$PATH:/usr/sbin:/opt/precisely/di/sqdata/bin`
Trying to start the Apply Engine returns one of the following connection errors:	Check the mainframe spool to make sure that the controller daemon jobs are running.

Issue	Solution
• `SQDD018E Cannot connect to transfer socket(rc==0x18468). Agent:<Agent Name > Socket:/u/./sqdata/.DB2ZTOMSK.cab.data` • `SQDUR06E Error opening url cdc://<VPC end point name>:2626/DB2ZTOMSK/DB2ZTOMSK : errno:1128 (Unknown error 1128)`	

Related resources

- Generate insights by using AWS Mainframe Modernization and Amazon Q in QuickSight (pattern)
- Generate data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight (demo)
- AWS Mainframe Modernization - Data Replication for IBM z/OS
- Amazon Redshift streaming ingestion to a materialized view

Additional information

Example .ddl files

members_details.ddl

```
CREATE TABLE MEMBER_DTLS (
memberid INTEGER NOT NULL,
member_name VARCHAR(50),
member_type VARCHAR(20),
age INTEGER,
gender CHAR(1),
email VARCHAR(100),
region VARCHAR(20)
);
```

member_plans.ddl

```
CREATE TABLE MEMBER_PLANS (
```

```
memberid INTEGER NOT NULL,
medical_plan CHAR(1),
dental_plan CHAR(1),
vision_plan CHAR(1),
preventive_immunization VARCHAR(20)
);
```

Example .sqd file

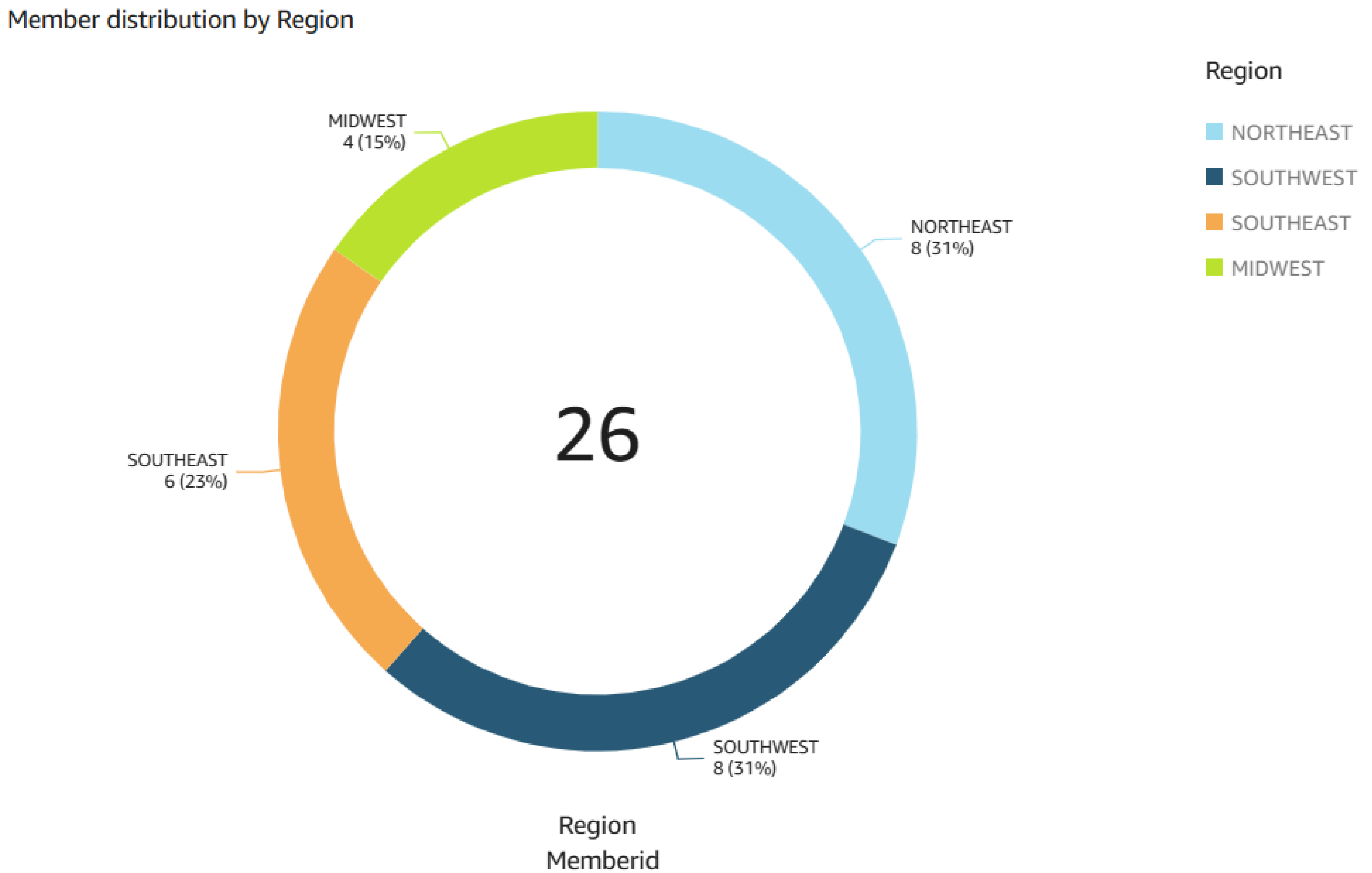
Replace `<kafka topic name>` with your Amazon MSK topic name.

script.sqd

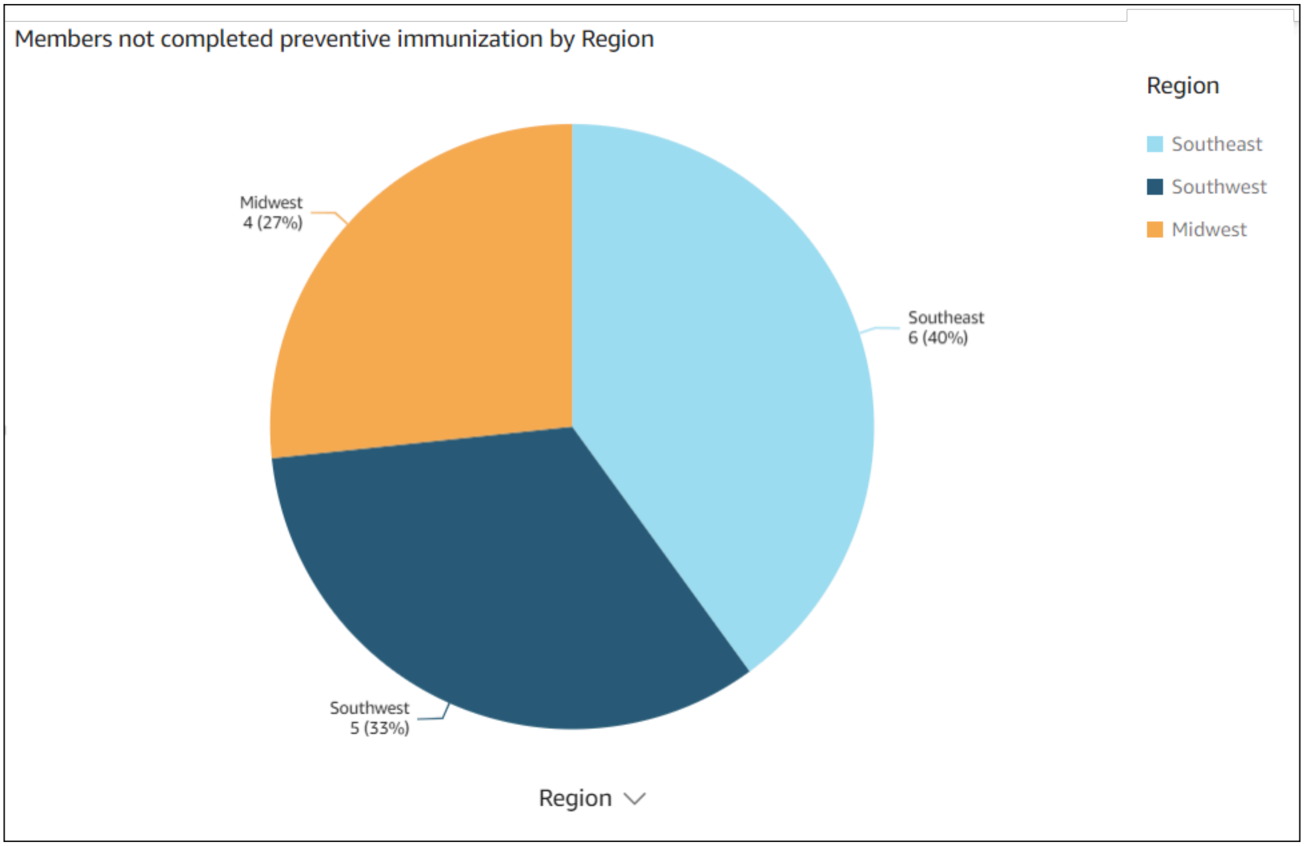
```
-- Name: DB2ZTOMSK: DB2z To MSK JOBNAME DB2ZTOMSK;REPORT EVERY 1;OPTIONS
 CDCOP('I','U','D');-- Source Descriptions
JOBNAME DB2ZTOMSK;
REPORT EVERY 1;
OPTIONS CDCOP('I','U','D');

-- Source Descriptions
BEGIN GROUP DB2_SOURCE;
DESCRIPTION DB2SQL /var/precisely/di/sqdata/apply/DB2ZTOMSK/ddl/mem_details.ddl AS
 MEMBER_DTLS;
DESCRIPTION DB2SQL /var/precisely/di/sqdata/apply/DB2ZTOMSK/ddl/mem_plans.ddl AS
 MEMBER_PLANS;
END GROUP;
-- Source Datastore
DATASTORE cdc://<zos_host_name>/DB2ZTOMSK/DB2ZTOMSK
OF UTSCDC
AS CDCIN
DESCRIBED BY GROUP DB2_SOURCE ;
-- Target Datastore(s)
DATASTORE 'kafka:///<kafka topic name>/key'
OF JSON
AS TARGET
DESCRIBED BY GROUP DB2_SOURCE;
PROCESS INTO TARGET
SELECT
{
REPLICATE(TARGET)
}
FROM CDCIN;
```

Mainframe visual dashboard

The following data visual was created by Amazon Q in QuickSight for the analysis question `show member distribution by region`.



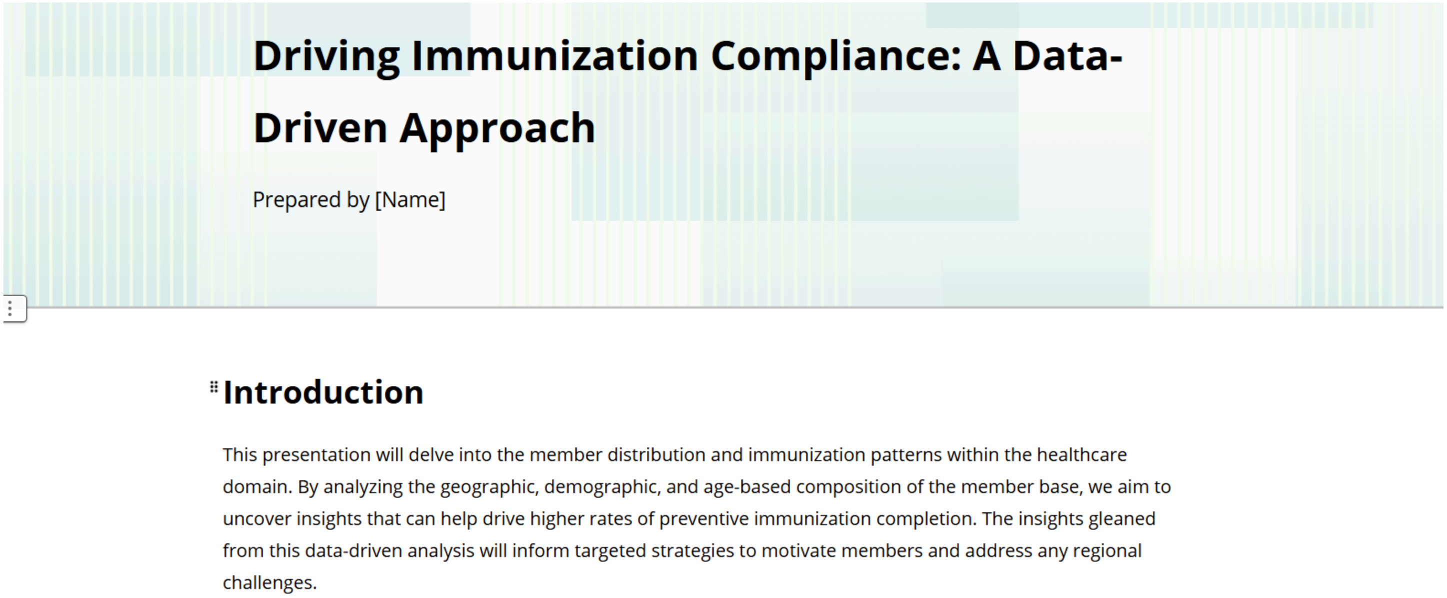
The following data visual was created by Amazon Q in QuickSight for the question `show member distribution by Region who have not completed preventive immunization, in pie chart`.



Data story output

The following screenshots show sections of the data story created by Amazon Q in QuickSight for the prompt `Build a data story about Region with most numbers of members. Also show the member distribution by age, member distribution by gender. Recommend how to motivate members to complete immunization. Include 4 points of supporting data for this pattern.`

In the introduction, the data story recommends choosing the region with the most members to gain the greatest impact from immunization efforts.



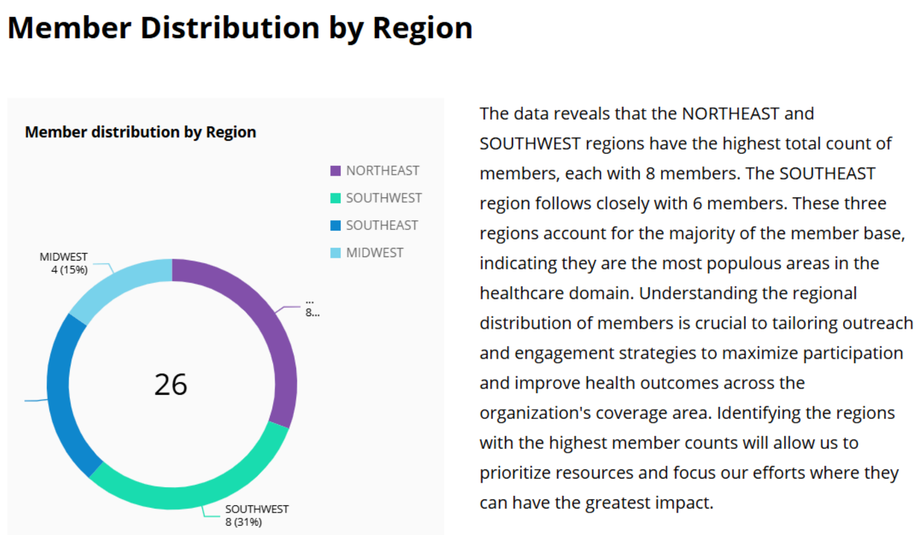
Driving Immunization Compliance: A Data-Driven Approach

Prepared by [Name]

Introduction

This presentation will delve into the member distribution and immunization patterns within the healthcare domain. By analyzing the geographic, demographic, and age-based composition of the member base, we aim to uncover insights that can help drive higher rates of preventive immunization completion. The insights gleaned from this data-driven analysis will inform targeted strategies to motivate members and address any regional challenges.

The data story provides an analysis of member numbers for the four regions. The Northeast, Southwest, and Southeast regions have the most members.

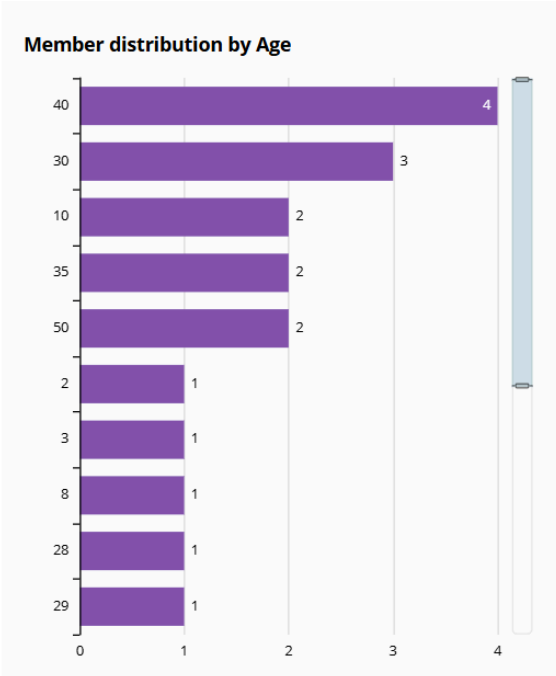


Member Distribution by Region

The data reveals that the NORTHEAST and SOUTHWEST regions have the highest total count of members, each with 8 members. The SOUTHEAST region follows closely with 6 members. These three regions account for the majority of the member base, indicating they are the most populous areas in the healthcare domain. Understanding the regional distribution of members is crucial to tailoring outreach and engagement strategies to maximize participation and improve health outcomes across the organization's coverage area. Identifying the regions with the highest member counts will allow us to prioritize resources and focus our efforts where they can have the greatest impact.

The data story presents an analysis of members by age.

Member Distribution by Age



The age distribution of the member base reveals some interesting trends. According to the graph data, the top three ages with the highest total distinct count of member IDs are 40 with 4 members, 30 with 3 members, and 10 with 2 members. Conversely, the bottom three ages with the lowest distinct member counts are 55, 45, and 28, each with only 1 member. This suggests that the healthcare organization's member base skews towards younger and middle-aged adults, with fewer senior and adolescent members. Understanding this age distribution can help the organization tailor its outreach and engagement strategies to better meet the needs of its target demographics.

The data story focuses on immunization efforts in the Midwest.

Personalized Outreach Campaigns

By leveraging the detailed member distribution insights, we can craft personalized outreach campaigns that resonate with our target audience. Tailoring the messaging and resources to address the unique needs and concerns of members in different age groups and genders will enhance the impact of our immunization promotion efforts. Deploying these targeted outreach initiatives across the regions identified as having the lowest immunization rates will help drive increased member engagement and completion of preventive care.

Addressing Regional Challenges

The Midwest region presents unique challenges in driving higher immunization rates among members. This area is characterized by a diverse population with varying socioeconomic backgrounds and access to healthcare resources. To address these regional nuances, we will develop targeted educational initiatives that resonate with the local community. Our outreach efforts will leverage partnerships with trusted community organizations to improve awareness and accessibility of preventive care services. By tailoring our approach to the specific needs of the Midwest, we can empower members to prioritize their health and complete recommended immunizations.

Anticipated Outcomes

By leveraging the proposed gamification strategies, personalized outreach campaigns, and targeted regional interventions, we expect to see a significant improvement in the overall immunization rates across the healthcare organization. This will enable us to enhance the health and well-being of our members, leading to better healthcare outcomes and potential cost savings. The implementation of these strategies will serve as a crucial step in ensuring our members receive the necessary preventive care and protection against potential health risks.

Conclusion

The insights gleaned from this comprehensive analysis of member distribution and immunization patterns within the healthcare domain provide a solid foundation for strategic interventions. By understanding the demographic composition of the member base and identifying regional variations in preventive care completion, we can develop tailored outreach campaigns and leverage gamification techniques to effectively motivate members to prioritize their health. These targeted efforts will not only improve individual wellness but also strengthen the overall member engagement and loyalty within the organization.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Generate data insights by using AWS Mainframe Modernization and Amazon Q in QuickSight

Shubham Roy, Roshna Razack, and Santosh Kumar Singh, Amazon Web Services

Summary

If your organization is hosting business-critical data in a mainframe environment, gaining insights from that data is crucial for driving growth and innovation. By unlocking mainframe data, you can build faster, secure, and scalable business intelligence to accelerate data-driven decision-making, growth, and innovation in the Amazon Web Services (AWS) Cloud.

This pattern presents a solution for generating business insights and creating sharable narratives from mainframe data by using AWS Mainframe Modernization File Transfer with BMC and Amazon Q in QuickSight. Mainframe datasets are transferred to Amazon Simple Storage Service (Amazon S3) by using AWS Mainframe Modernization File Transfer with BMC. An AWS Lambda function formats and prepares mainframe data file for loading into Amazon QuickSight.

After the data is available in QuickSight, you can use natural language prompts with Amazon Q in QuickSight to create summaries of the data, ask questions, and generate data stories. You don't have to write SQL queries or learn a business intelligence (BI) tool.

Business context

This pattern presents a solution for mainframe data analytics and data insights use cases. Using the pattern, you build a visual dashboard for your company's data. To demonstrate the solution, this pattern uses a health care company that provides medical, dental, and vision plans to its members in the US. In this example, member demographics and plan information are stored in the mainframe datasets. The visual dashboard shows the following:

- Member distribution by region
- Member distribution by gender
- Member distribution by age
- Member distribution by plan type
- Members who have not completed preventive immunization

After you create the dashboard, you generate a data story that explains the insights from the previous analysis. The data story provides recommendations for increasing the number of members who have completed preventive immunizations.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Mainframe datasets with business data
- Access to install a file transfer agent on the mainframe

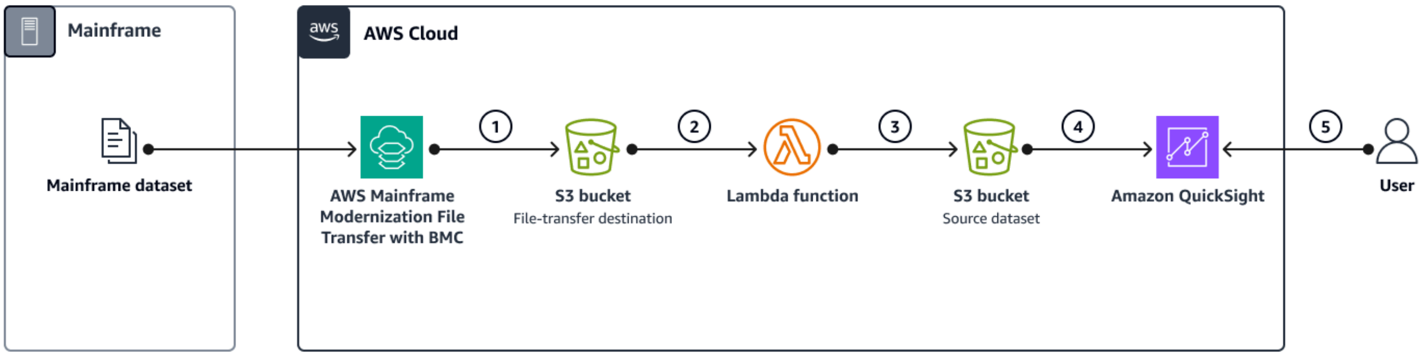
Limitations

- Your mainframe data file should be in one of the file formats supported by QuickSight. For a list supported file formats, see the Amazon QuickSight documentation.

  This pattern uses a Lambda function to convert the mainframe file into a format supported by QuickSight.

Architecture

The following diagram shows an architecture for generating business insights from mainframe data by using AWS Mainframe Modernization File Transfer with BMC and Amazon Q in QuickSight.



The diagram shows the following workflow:

1. A mainframe dataset containing business data is transferred to Amazon S3 by using AWS Mainframe Modernization File Transfer with BMC.

2. The Lambda function converts the file that's in the file-transfer destination S3 bucket to comma-separated values (CSV) format.
3. The Lambda function sends the converted file to the source dataset S3 bucket.
4. The data in the file is ingested by QuickSight.
5. Users access the data in QuickSight. You can use Amazon Q in QuickSight to interact with the data by using natural language prompts.

Tools

AWS services

- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Mainframe Modernization File Transfer with BMC converts and transfers mainframe datasets to Amazon S3 for mainframe modernization, migration, and augmentation use cases.
- Amazon QuickSight is a cloud-scale BI service that helps you visualize, analyze, and report your data in a single dashboard. This pattern uses the generative BI capabilities of Amazon Q in QuickSight.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Best practices

- When you create the AWS Identity and Access Management (IAM) roles for AWS Mainframe Modernization File Transfer with BMC and the Lambda function, follow the principle of least privilege.
- Ensure that your source dataset has supported data types for QuickSight. If your source dataset contains unsupported data types, convert them to supported data types. For information about unsupported mainframe data types and how to convert them to data types supported by Amazon Q in QuickSight, see the Related resources section.

Epics

Set up AWS Mainframe Modernization File Transfer with BMC

Task	Description	Skills required
Install the file transfer agent.	To install AWS Mainframe Modernization File Transfer Agent on your mainframe, follow the instructions in the AWS documentation.	Mainframe system administr ator
Create an S3 bucket for mainframe file transfer.	Create an S3 bucket to store the output file from AWS Mainframe Modernization File Transfer with BMC. In the architecture diagram, this is the file-transfer destination bucket.	Migration engineer
Create the data transfer endpoint.	1. Create an S3 bucket to stage the input mainframe file for AWS Mainframe Modernization File Transfer with BMC. 2. To create the mainframe data transfer endpoint, follow the instructions in the AWS documentation.	AWS Mainframe Moderniza tion specialist

Convert the mainframe file name extension for QuickSight integration

Task	Description	Skills required
Create an S3 bucket.	Create an S3 bucket for the Lambda function to copy the converted mainframe file	Migration engineer

Task	Description	Skills required
	from the source to the final destination bucket.	

Task	Description	Skills required
Create a Lambda function.	To create a Lambda function that changes the file extension and copies the mainframe file to the destination bucket, do the following: 1. Sign in to the AWS Management Console, and navigate to the AWS Lambda console. 2. Choose **Create function**, and then choose **Author from scratch**. 3. For **Function name**, enter a name for your function. 4. In the **Runtime** dropdown list, choose **Python.3.X**. 5. Expand **Change default execution role**, and then choose **Create a new role with basic Lambda permissions**. 6. Choose **Create function**. 7. Choose the **Code** tab, and then paste the `S3CopyLambda.py` Python code that's provided in the [Additional information](#) section. The Python code was generated by using [Amazon Q Developer](#) in the Microsoft Visual Studio	Migration engineer

Task	Description	Skills required
	integrated development environment (IDE). 8. Edit the `destination_bucket_name` to the name of the S3 bucket that you created previously, and change `destination_file_key` to the mainframe file name. 9. Deploy the Lambda function.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create an Amazon S3 trigger to invoke the Lambda function.</td><td>To configure a trigger that invokes the Lambda function, do the following:

1. On the Lambda console, open the Functions page.
2. Choose the Lambda function.
3. In Function overview, choose Add trigger.
4. In the Trigger configura tion dropdown list, choose S3.
5. In the Bucket field, enter the name of your source bucket.
6. In the Event type dropdown list, choose All object create events.
7. Select the I acknowledge that using the same S3 bucket for both input and output is not recommend ed check box, and then choose Add.

For more information, see Tutorial: Using an Amazon S3 trigger to invoke a Lambda function.</td><td>Migration lead</td></tr>
</table>

Task	Description	Skills required
Provide IAM permissions for the Lambda function.	IAM permissions are required for the Lambda function to access the file-transfer destination and source dataset S3 buckets. Update the policy associated with the Lambda function execution role by allowing `s3:GetObject` and `s3:DeleteObject` permissions for the file-transfer destination S3 bucket and `s3:PutObject` access for the source dataset S3 bucket. For more information, see the [Create a permissions policy](#) section in *Tutorial: Using an Amazon S3 trigger to invoke a Lambda function*.	Migration lead

Define a mainframe data transfer task

Task	Description	Skills required
Create a transfer task to copy the mainframe file to the S3 bucket.	To create a mainframe file transfer task, follow the instructions in the [AWS Mainframe Modernization documentation.](#) **Note** Specify **Source code page** encoding as	Migration engineer

Task	Description	Skills required
	IBM1047 and **Target code page** encoding as **UTF-8**.	
Verify the transfer task.	To verify that the data transfer is successful, follow the instructions in the [AWS Mainframe Modernization documentation](). Confirm that the mainframe file is in the file-transfer destination S3 bucket.	Migration lead
Verify the Lambda copy function.	Verify that the Lambda function is initiated and that the file is copied with a .csv extension to the source dataset S3 bucket. The .csv file created by the Lambda function is the input data file for QuickSight. For example data, see the `Sample-data-member-healthcare-APG` file in the [Attachments]() section.	Migration lead

Connect QuickSight to the mainframe data

Task	Description	Skills required
Set up QuickSight.	To set up QuickSight, follow the instructions in the [AWS documentation]().	Migration lead

Task	Description	Skills required
Create a dataset for QuickSight.	To create a dataset for QuickSight, follow the instructions in the AWS documentation. The input data file is the converted mainframe file that was created when you defined the mainframe data transfer task.	Migration lead

Get business insights from the mainframe data by using Amazon Q in QuickSight

Task	Description	Skills required
Set up Amazon Q in QuickSight.	This capability requires Enterprise Edition. To set up Amazon Q in QuickSight, do the following: 1. To get the Amazon Q add-on, follow the instructions Step 1: Get the Q add-on in the AWS documentation. 2. To use the generative BI capabilities in Amazon Q, upgrade your users' accounts. Follow the instructions in the AWS documentation. 3. Create an Amazon Q topic by using the dataset that you created previously. Follow the instructions in the AWS documentation.	Migration lead

Task	Description	Skills required
	4. To configure the topic metadata so that it's natural language–friendly, follow the instructions in the [AWS documentation]().	

Task	Description	Skills required
Analyze mainframe data and build a visual dashboard.	To analyze and visualize your data in QuickSight, do the following: 1. To create the mainframe data analysis, follow the instructions in the [AWS documentation](). For the dataset, choose the dataset created in the previous step. 2. On the analysis page choose **Build visual**. 3. In the **Create topic for analysis** window, choose **Update existing topic**. 4. In the **Select a topic** dropdown list, choose the topic that you created previously. 5. Choose **Topic linking**. 6. After you link the topic, choose **Build visual** to open the Amazon Q **Build a Visual** window. 7. In the prompt bar. write your analysis questions . The example questions used for this pattern are the following: • Show member distribut ion by region	Migration engineer

Task	Description	Skills required
	• Show members distribution by age • Show member distribution by gender • Show member distribution by plan type • Show member not completed preventive immunization After you enter your questions, choose **Build**. Amazon Q in QuickSight creates the visuals. 8. To add the visuals to your visual dashboard, choose **ADD TO ANALYSIS**. When you're finished, you can publish your dashboard to share with others in your organization. For examples, see *Mainframe visual dashboard* in the [Additional information](#) section.	

Create a data story with Amazon Q in QuickSight from the mainframe data

Task	Description	Skills required
Create a data story.	Create a data story to explain insights from the previous	Migration engineer

Task	Description	Skills required
	analysis, and generate a recommendation to increase preventive immunization for members: 1. To create the data story, follow the instructions in the [AWS documentation](#). 2. For the data story prompt, use the following: `Build a data story about Region with most numbers of members. Also show the member distribution by medical plan, vision plan, dental plan. Recommend how to motivate members to complete immunizat ion. Include 4 points of supportin g data for this pattern.` You can also build your own prompt to generate data stories for other business insights. 3. Choose **Add visuals**, and add the visuals that are relevant to the data story. For this pattern, use the	

Task	Description	Skills required
	visuals that you created previously. 4. Choose **Build**. 5. For example data story output, see *Data story output* in the [Additional information](#) section.	
View the generated data story.	To view the generated data story, follow the instructions in the [AWS documentation](#).	Migration lead
Edit a generated data story.	To change the formatting, layout, or visuals in a data story, follow the instructions in the [AWS documentation](#).	Migration lead
Share a data story.	To share a data story, follow the instructions in the [AWS documentation](#).	Migration engineer

Troubleshooting

Issue	Solution
Unable to discover the mainframe files or datasets entered in **Data sets search criteria** for **Create transfer task** in AWS Mainframe Modernization File Transfer with BMC.	1. First, check the connection by choosing **Data transfer endpoints** on the AWS Mainframe Modernization Transfer with BMC console. If the last heartbeat time is greater than two minutes, the connectio n for file transfer hasn't been establish ed. If the last heartbeat time is less than 2 minutes for the agent running on the mainframe, the connection to the agent is successful. Proceed to step 2.

Issue	Solution
	2. Check the AWS Secrets Manager setup. A secret key must be configured in Secrets Manager with a **key** of `userId` (uppercase I) with a **value** of the mainframe's user ID and a **key** of `password` with the **value** of the mainframe password. The `userId` and `password` secret keys are case-sensitive and must be entered as is.

Related resources

To convert mainframe data types such as PACKED-DECIMAL (COMP-3) or BINARY (COMP or COMP-4) to a data type supported by QuickSight, see the following patterns:

- Convert and unpack EBCDIC data to ASCII on AWS by using Python
- Convert mainframe files from EBCDIC format to character-delimited ASCII format in Amazon S3 using AWS Lambda

Additional information

S3CopyLambda.py

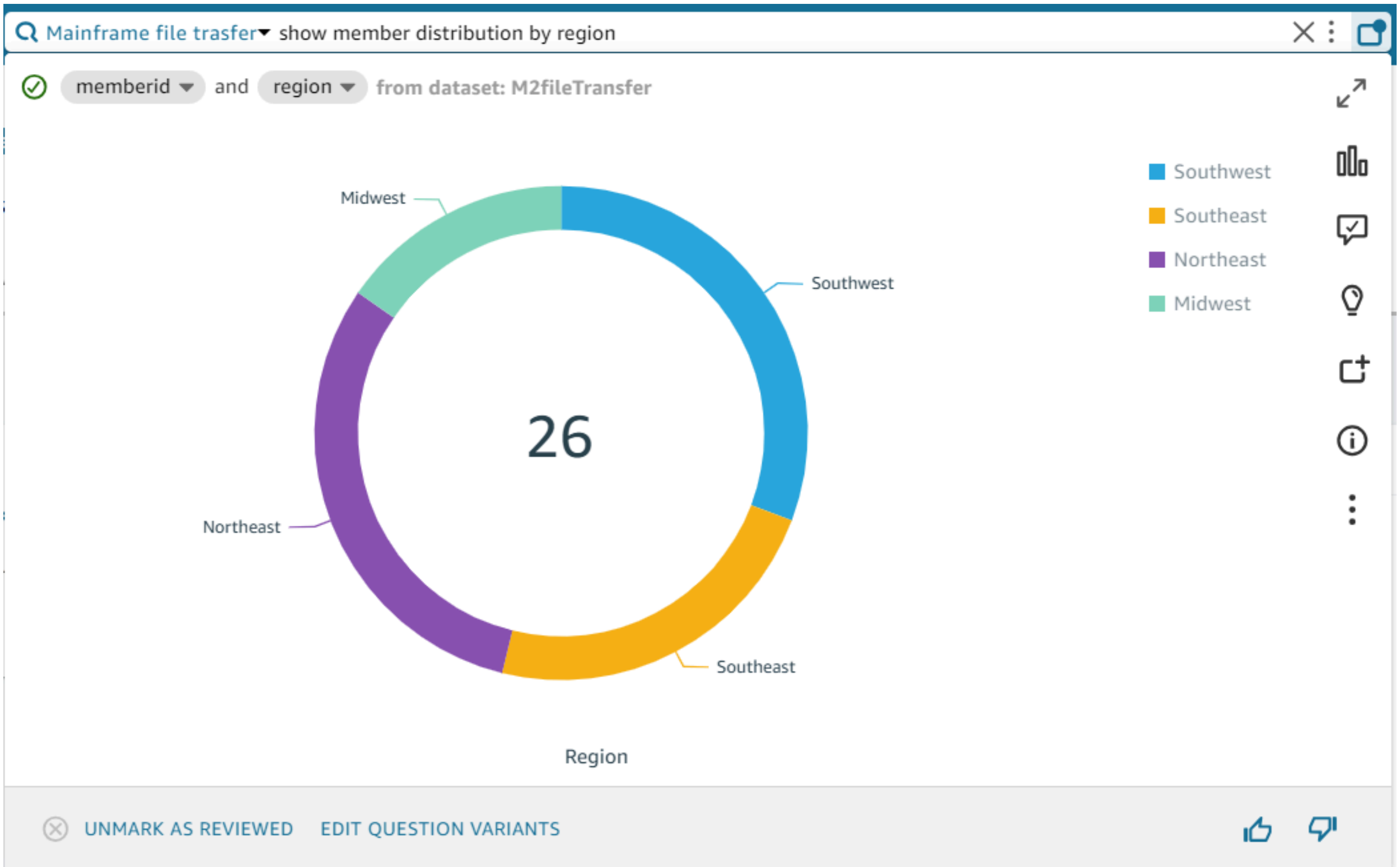
The following Python code was generated by using a prompt with Amazon Q Developer in an IDE:

```
#Create a lambda function triggered by S3. display the S3 bucket name and key
import boto3
s3 = boto3.client('s3')
def lambda_handler(event, context):
print(event)
bucket = event['Records'][0]['s3']['bucket']['name']
key = event['Records'][0]['s3']['object']['key']
print(bucket, key)
#If key starts with object_created, skip copy, print "copy skipped". Return lambda with
 key value.
if key.startswith('object_created'):
print("copy skipped")
return {
```

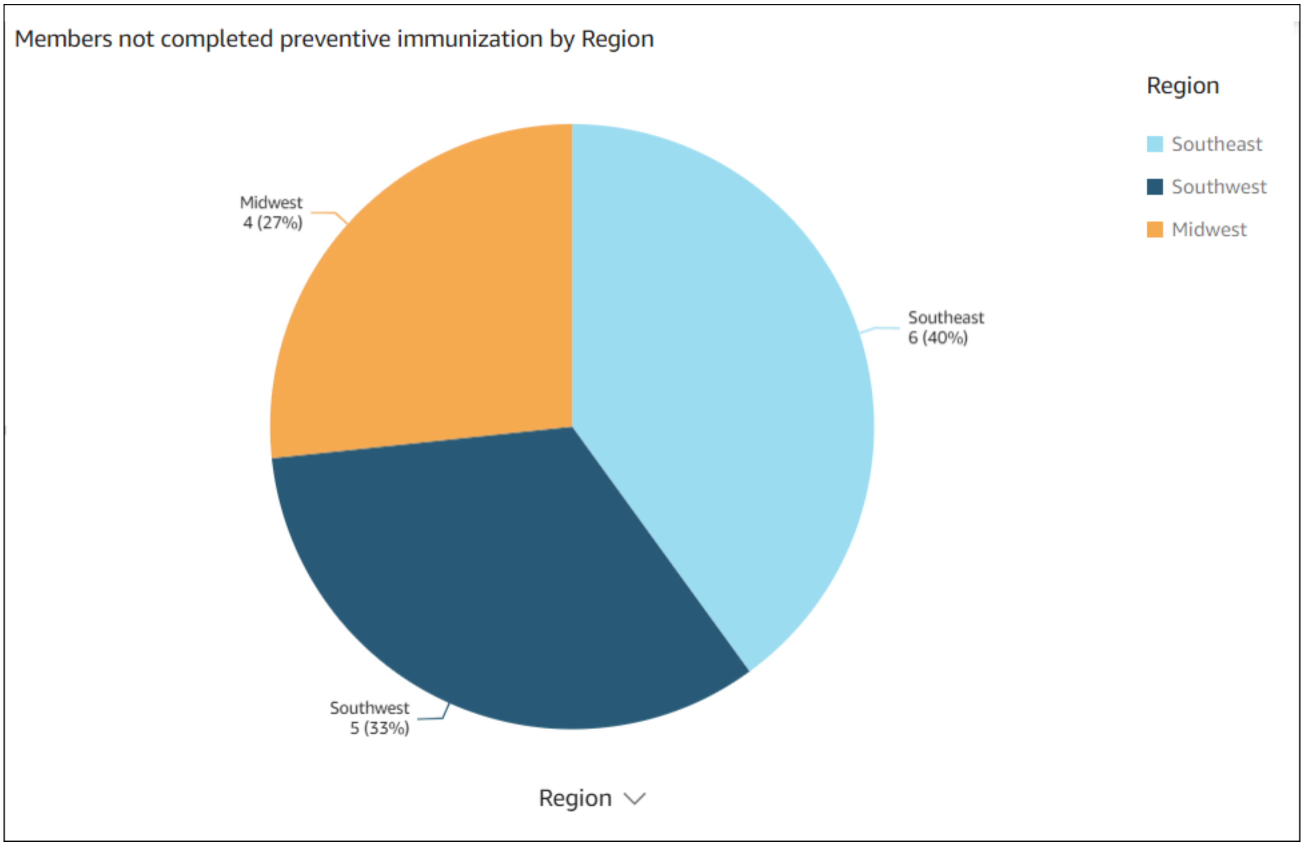
```
'statusCode': 200,
'body': key
}
# Copy the file from the source bucket to the destination bucket.
 Destination_bucket_name = 'm2-filetransfer-final-opt-bkt'. Destination_file_key =
 'healthdata.csv'
copy_source = {'Bucket': bucket, 'Key': key}
s3.copy_object(Bucket='m2-filetransfer-final-opt-bkt', Key='healthdata.csv',
 CopySource=copy_source)
print("file copied")
#Delete the file from the source bucket.
s3.delete_object(Bucket=bucket, Key=key)
return {
'statusCode': 200,
'body': 'Copy Successful'
}
```

Mainframe visual dashboard

The following data visual was created by Amazon Q in QuickSight for the analysis question `show member distribution by region`.



The following data visual was created by Amazon Q in QuickSight for the question `show member distribution by Region who have not completed preventive immunization, in pie chart`.

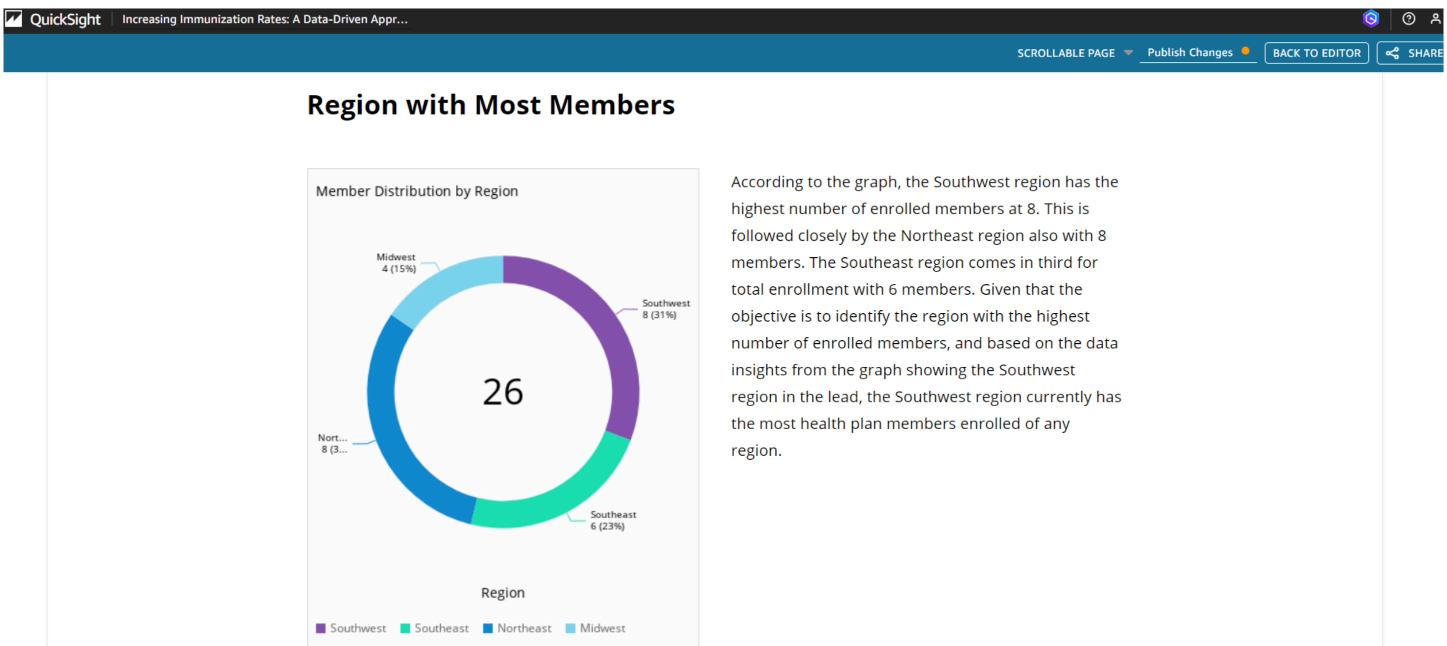


Data story output

The following screenshots show sections of the data story created by Amazon Q in QuickSight for the prompt `Build a data story about Region with most numbers of members. Also show the member distribution by medical plan, vision plan, dental plan. Recommend how to motivate members to complete immunization. Include 4 points of supporting data.`

In the introduction, the data story recommends choosing the region with the most members to gain the greatest impact from immunization efforts.

The data story provides an analysis of member numbers for the top three regions, and names the Southwest as the leading region for focusing on immunization efforts.



Note

The Southwest and Northeast regions each have eight members. However, the Southwest has more members that aren't fully vaccinated, so it has more potential to benefit from initiatives to increase immunization rates.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Implement Microsoft Entra ID-based authentication in an AWS Blu Age modernized mainframe application

Vishal Jaswani and Rimpy Tewani, Amazon Web Services

Summary

Mainframe applications that are modernized by using refactoring patterns, such as those by AWS Mainframe Modernization Refactor with AWS Blu Age, require careful integration of authentication mechanisms into the new application architecture. This integration is typically addressed as a post-modernization activity. The task can be complex and often involves the migration or externalization of existing authentication systems to align with modern security standards and cloud-native practices. Developers need to consider how to implement authentication effectively while they work within the constraints of the modernized application's runtime environment and libraries. After modernization, AWS provides ways to make it easier for you to integrate your AWS Blu Age modern code with identity and access management systems such as Amazon Cognito and Microsoft Entra ID (formerly known as Azure AD).

This pattern explains how to implement an authentication mechanism in your modernized application when the authentication provider is Microsoft Entra ID, without spending time on research and trials. The pattern provides:

- Field-tested and relevant Angular libraries from the Microsoft Authentication Library (MSAL) and other Microsoft Entra ID documentation that are essential to the authentication implementation.
- Configurations required on the AWS Blu Age Runtime to enable Spring Security by using OAuth 2.0.
- A library that captures authenticated users' identities and passes them to the AWS Blu Age Runtime.
- Security measures that we recommend implementing.
- Troubleshooting tips for commonly encountered problems with the Microsoft Entra ID setup.

> **Note**
>
> This pattern uses the AWS Blu Age OAuth extension library, which is provided to customers as part of their AWS Professional Services engagement. This library isn't part of the AWS Blu Age Runtime.

Prerequisites and limitations

Prerequisites

- A modernized application that was produced by AWS Blu Age mainframe modernization refactoring tools. This pattern uses CardDemo as a sample open source mainframe application.
- The AWS Blu Age OAuth extension library, which is provided by the AWS Blu Age team during your engagement with AWS Professional Services.
- An active AWS account to deploy and test the modernized application.
- Familiarity with AWS Blu Age configuration files and Microsoft Entra ID fundamentals.

Limitations

- This pattern covers OAuth 2.0 authentication and basic token-based authorization flows. Advanced authorization scenarios and fine-grained access control mechanisms are not in scope.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.

Product versions

This pattern was developed by using:

- AWS Blu Age Runtime version 4.1.0 (the pattern also works with later versions that are backward compatible)
- MSAL library version 3.0.23
- Java Development Kit (JDK) version 17
- Angular version 16.1

Architecture

Source technology stack

In typical mainframe environments, authentication is implemented through user profiles. These profiles identify users to the system, define who can sign in, and specify which functions users can perform on system resources. User profiles are managed by security officers or security administrators.

Target technology stack

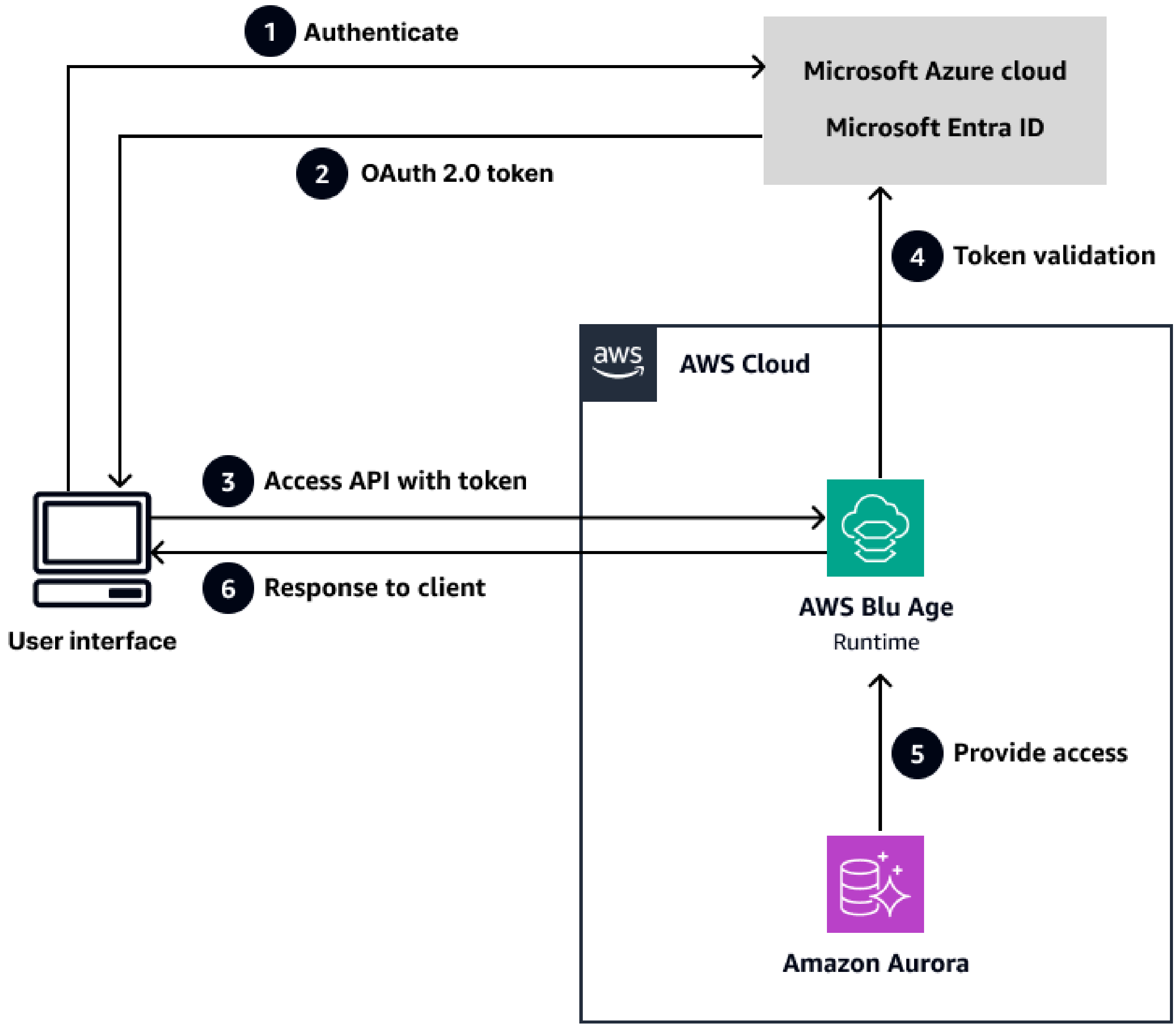
- Microsoft Entra ID
- Modernized Java Spring Boot-based backend
- AWS Blu Age Runtime
- Spring Security with OAuth 2.0
- Angular single-page application (SPA)

Target architecture

AWS Blu Age runtime supports OAuth 2.0-based authentication by default, so the pattern uses that standard to protect backend APIs.

The following diagram illustrates the process flow.

> **Note**
>
> The diagram includes Amazon Aurora as an example of database modernization although Aurora isn't included in the steps for this pattern.



where:

1. A user tries to authenticate with Microsoft Entra ID.
2. Microsoft Entra ID returns refresh, access, and ID tokens that the application uses in subsequent calls.
3. The MSAL interceptor includes the access token in the `Authorization` header of an HTTPS request to call the AWS Blu Age Runtime.

4. The AWS Blu Age `extension-oauth` library extracts the user information from the header by using an AWS Blu Age Runtime configuration file (`application-main.yml`) and places this information in a `SharedContext` object so that the business logic can consume it.

> **Note**
>
> `SharedContext` is a runtime component provided by AWS Blu Age that manages application context and state information across the modernized application. For more information about AWS Blu Age Runtime components and updates, see [AWS Blu Age release notes] in the AWS Mainframe Modernization documentation. For more information about the `application-main.yml` file, see [Set up configuration for AWS Blu Age Runtime] in the AWS Mainframe Modernization documentation.

5. The AWS Blu Age Runtime checks if the token is present.
   a. If the token is present, it checks the validity of the token by communicating with Microsoft Entra ID.
   b. If the token isn't present, the AWS Blu Age Runtime returns an error with HTTP status code 403.
6. If the token is valid, the AWS Blue Age Runtime allows the business logic to continue. If the token is invalid, the AWS Blu Age Runtime returns an error with HTTP status code 403.

OAuth 2.0 workflow

For a high-level diagram of the OAuth 2.0 workflow, see the Microsoft Entra documentation.

Tools

AWS services

- AWS Mainframe Modernization provides tools and resources to help you plan and implement migration and modernization from mainframes to AWS managed runtime environments. You can use this service's refactoring features, which are provided by AWS Blu Age, to convert and modernize your legacy mainframe applications.

Code repository

The CardDemo application has been updated to demonstrate integration with Microsoft Entra ID. You can access the code from the GitHub repository for this pattern.

Backend configuration

This pattern requires changes to the `application-main.yml` configuration file to enable Spring Security by using OAuth 2.0 on the backend application. The `.yml` file looks like this:

```
gapwalk-application.security: enabled
gapwalk-application:
  security:
    identity: oauth
    issuerUri: ${issuerUrl}
    claim:
      claims:
        -
          claimName: upn
          claimMapValue: username
spring:
  autoconfigure:
    exclude:
     -
 org.springframework.boot.autoconfigure.security.oauth2.client.servlet.OAuth2ClientAutoConfigur
     -
 org.springframework.boot.autoconfigure.security.oauth2.resource.servlet.OAuth2ResourceServerAu
  security:
    oauth2:
      client:
        registration:
          azure:
            client-id: {clientId}
            client-secret: ${clientSecret}
            provider: azure
            authorization-grant-type: authorization_code
            redirect-uri: ${redirectUri}
            scope: openid

        provider:
          azure:
            authorization-uri: ${gapwalk-application.security.issuerUri}/oauth2/v2.0/
authorize
            token-uri:  ${gapwalk-application.security.issuerUri}/oauth2/v2.0/token
            jwk-set-uri: ${gapwalk-application.security.issuerUri}/discovery/v2.0/keys
      resourceserver:
        jwt:
          jwk-set-uri: ${gapwalk-application.security.issuerUri}/discovery/v2.0/keys
```

AWS Blu Age OAuth extension filter library

The AWS Blu Age OAuth extension library is provided by the AWS Blu Age team during your engagement with [AWS Professional Services](https://aws.amazon.com/professional-services/).

This library reads the `claim.claims` configuration in the `application-main.yml` fie that's shown in the previous code block. This configuration is a list. Each item in the list provides two values: `claimName` and `claimMapValue`. `claimName` represents a key name in a JSON Web Token (JWT) sent by the frontend, and `claimMapValue` is the name of the key in `SharedContext`. For example, if you want to capture the user ID on the backend, set `claimName` to the key name in the JWT that holds the `userId` that's provided by Microsoft Entra ID, and set `claimMapValue` to the key name to fetch the user ID in the backend code.

For example, if you set `UserId` in `claimMapValue`, you can use the following code to extract the user ID:

```
SharedContext.get().getValue("userId", [UserId]);
```

Best practices

In your implementation of this pattern, take the following important security considerations into account.

> **Important**
>
> This pattern provides a foundation for authentication integration. We recommend that you implement security measures in addition to those discussed in this section based on your business requirements before you deploy it to production.

- **AWS configuration security.** Move sensitive configuration values from `application-main.yml` to AWS Secrets Manager. For example, configure the following properties by using Secrets Manager:

```
security:
    oauth2:
      client:
        registration:
          azure:
```

```
            client-id: {clientId}
            client-secret: ${clientSecret}
```

For more information about how you can use Secrets Manager to configure AWS Blu Age parameters, see AWS Blu Age Runtime secrets in the AWS Mainframe Modernization documentation.

- **Runtime environment protection.** Configure the modernized application environment with proper AWS security controls:

```
server:
  tomcat:
    remoteip:
     protocol-header: X-Forwarded-Proto
     remote-ip-header: X-Forwarded-For
  forward-headers-strategy: NATIVE
```

- **Amazon CloudWatch logging.** Consider adding the file `logback-spring.xml to src/main/resources`:

```
<configuration>
 <appender name="CLOUDWATCH"
 class="com.amazonaws.services.logs.logback.CloudWatchAppender">
   <logGroup>/aws/bluage/application</logGroup>
   <logStream>${AWS_REGION}-${ENVIRONMENT}</logStream>
   <layout>
    <pattern>%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n</
pattern>
   </layout>
 </appender>

 <root level="INFO">
 <appender-ref ref="CLOUDWATCH"/>
 </root>
</configuration>
```

For information about enabling tracing with CloudWatch, see Enable trace to log correlation in the CloudWatch documentation.

- **Token configuration and handling.** Configure token lifetimes in Microsoft Entra ID to align with your security requirements. Set access tokens to expire within 1 hour and refresh tokens to expire within 24 hours. In the AWS Blu Age Runtime configuration (`application-main.yml`), make sure that JWT validation is configured properly with the exact issuer URI and audience values from your Entra ID application registration.

When a token expires and is refreshed:

1. The Angular application's error interceptor handles the 401 response by obtaining a new token through MSAL.
2. The new token is sent with the subsequent request.
3. The AWS Blu Age Runtime's OAuth filter validates the new token and automatically updates `SharedContext` with the current user information. This ensures that business logic continues to have access to valid user context through `SharedContext.get().getValue()` calls.

For more information about the AWS Blu Age Runtime components and their updates, see AWS Blu Age release notes.

- **AWS Blu Age Runtime security.** The `oauth2-ext` library provided by AWS Blu Age must be placed in the correct shared directory location (`{app-server-home}/shared/`) with proper file permissions. Verify that the library successfully extracts user information from JWTs by checking the `SharedContext` object population in your logs.
- **Specific claims configuration.** In `application-main.yml`, define the claims that you need from Microsoft Entra ID explicitly. For example, to capture the user's email and roles, specify:

```
gapwalk-application:
  security:
    claim:
      claims:
        - claimName: upn
          claimMapValue: username
        - claimName: roles
          claimMapValue: userRoles
        - claimName: email
          claimMapValue: userEmail
```

- **Error handling.** Add error handling to address authentication failures in your Angular application; for example:

```
@Injectable()
export class AuthErrorInterceptor implements HttpInterceptor {
  intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
 {
    return next.handle(request).pipe(
```

```
      catchError((error: HttpErrorResponse) => {
        if (error.status === 401) {
          // Handle token expiration
          this.authService.login();
        }
        if (error.status === 403) {
          // Handle unauthorized access
          this.router.navigate(['/unauthorized']);
        }
        return throwError(() => error);
      })
    );
  }
}
```

- **Session timeout configuration.** Configure session timeout settings in both the AWS Blu Age Runtime and Microsoft Entra ID. For example, add the following code to your `application-main.yml` file:

```
server:
  servlet:
    session:
      timeout: 3600 # 1 hour in seconds
```

- **MsalGuard.** You must implement the MsalGuard feature for all protected routes to prevent unauthorized access. For example:

```
const routes: Routes = [
    { path: '', redirectTo: '/transaction-runner', pathMatch: 'full' },
    { path: 'transaction-runner', component: TransactionRunnerComponent,
 canActivate:guards },
    { path: 'user-info', component: UserInfoComponent, canActivate:guards },
    { path: 'term/:transid/:commarea', component: TermComponent,
 canActivate:guards },
 { path: 'code', component: TransactionRunnerComponent  }
];
```

  Routes that don’t have MsalGuard protection will be accessible without authentication, potentially exposing sensitive functionality. Make sure that all routes that require authentication include the guards in their configuration.

Epics

Set up a Microsoft Entra ID

Task	Description	Skills required
Set up a Microsoft Azure account to create an Entra ID.	For options and instructi ons, see the Microsoft Azure website.	App developer
Set up a Microsoft Entra ID in your application.	To learn how to add Microsoft Entra ID B2C (Azure AD B2C) authentication to your Angular SPA, see the Microsoft documentation. Specifically: 1. Register your application and record identifiers. 2. Expose the AWS Blu Age transaction endpoint by adding and exposing a custom scope. 3. After you define the scope and the API URL, configure these values in the `apiUri` and `apiScope` properties of the `environment.ts` file in the code repository.	App developer

Clone the repository and deploy your AWS Blu Age code

Task	Description	Skills required
Clone the GitHub repositor y to get the Angular code required for authentication.	Run the following command to clone the GitHub repositor y that's provided with this	App developer

Task	Description	Skills required
	pattern into your local current working directory: `git clone https://github.com/aws-samples/sample-microsoft-entra-id-based-auth-in-aws-bluage-modernized-mainframe-app.git`	
Deploy the AWS Blu Age modernized code on a Tomcat server to implement authentication.	To set up the local environment that includes Tomcat and the Angular development server, follow the installation steps provided by the AWS Blu Age team as part of your customer engagement with AWS Professional Services.	App developer

Build the authentication solution

Task	Description	Skills required
Enable AWS Blu Age Runtime security to protect the AWS Blu Age REST API endpoints.	Configure the `application-main.yml` file that the AWS Blu Age Runtime uses as follows. For an example of this file, see the Code repository section earlier in this pattern. • `spring:security:oauth2:client:*` enables Spring Security OAuth 2.0 client support.	App developer

Task	Description	Skills required
	Replace the settings with values that are specific to your Microsoft Entra ID setup. • `spring:security:oauth2:resourceserver` enables Spring Security OAuth 2.0 resource server support. • `gapwalk-application.security.issuerUri` specifies the URL of the identity provider for fetching configuration information such as the authorization endpoint, the token endpoint, and so on. • `gapwalk-application.security.identity` should be set to `oauth`. • `gapwalk-application.security.claim.claims` specifies the list of claims required to capture user IDs and user names. For an example, see the [Code repository] section earlier in this pattern. • `gapwalk-application.security` should be set to `enabled`. • `spring.autoconfigure.exclude` should	

Task	Description	Skills required
	be removed to enable authentication.	
Incorporate the example code from your local environment into your Blu Age modernized Angular code base.	For information about how to incorporate the example into your AWS Blu Age modernized Angular code base, see the [Code repository](#) section earlier in this pattern.	App developer

Place the `oauth2-ext` library in the shared directory.	Place the `oauth2-ext` library in the shared directory of the application server so that your AWS Blu Age modernized application can use it. Run the following commands:
	`cd oauth2-ext/target`
	`cp extension-oauth-filter-<version>.jar /{app-server-home}/shared/`
		App developer

Deploy the authentication solution

Task	Description	Skills required

Deploy the frontend application.	Run the following commands to start the frontend application locally:
	`npm install`
	`ng serve --ssl`
	`npm start`
		App developer

Task	Description	Skills required
	Note Adding the `--ssl` flag to the `ng serve` command ensures that the developme nt server uses HTTPS, which is more secure than other protocols and provides a better simulation of a production environme nt.	
Start the backend application.	Start Tomcat server in Eclipse.	App developer

Test the application

Task	Description	Skills required
Test login functionality.	Access the locally deployed application at `http://localhost:4200` to verify that users are asked to confirm their identity. **Note** HTTP is used here for demonstra tion purposes. In a production or other publicly accessible environment, you	App developer

Task	Description	Skills required
	must use HTTPS for security. Even for local development, we recommend that you set up HTTPS when possible. The Microsoft login prompt should appear, and users who are configured in Microsoft Entra ID should be allowed to access the application.	

Task	Description	Skills required
Test the authorization header in the request.	**Note** The following steps use the [CardDemo](#) application as an example. Testing steps for other modern applications will vary. 1. Launch transaction `CC00` in the CardDemo application. 2. Sign in with user credentia ls. For example, if you're using the [AWS Blu Age level 3 self-paced workshop](#), you can use the name `USER0001` and password `PASSWORD` for credentials. 3. Open the **Developer Tools** window in the same tab of your browser. 4. Open the **Network** tab and check the request sent to the backend `http://localhost:8080/gapwalk-application/transaction` . 5. Check the request header and verify that you see an authorization header with the value `Bearer`	App developer

Task	Description	Skills required
	`<token>` where `<token>` is a value that's generated by Microsoft Entra ID.	
Test the logout functionality.	Choose **Quit** to log out, and try to access the application again. It should present a new login prompt.	App developer

Troubleshooting

Issue	Solution
The token issued by Microsoft Entra ID isn't compatible with Spring Boot OAuth 2.0 security.	For a resolution to the issue, see Microsoft Entra ID OAuth Flow on the OAuth blog.
General token-related questions.	To decode and view the contents of a JWT token, use the https://jwt.io/ website.

Related resources

- For information about refactoring your application by using AWS Blu Age, see the AWS Mainframe Modernization documentation.
- To understand how OAuth 2.0 works, see the OAuth 2.0 website.
- For an overview of the Microsoft Authentication Library (MSAL), see the Microsoft Entra documentation.
- For information about user profiles on an AS/400 system, see the IBM i (AS400) tutorial.
- For the OAuth 2.0 and OpenID Connect (OIDC) authentication flow in the Microsoft identity platform, see the Microsoft Entra documentation.

Integrate Stonebranch Universal Controller with AWS Mainframe Modernization

Vaidy Sankaran and Pablo Alonso Prieto, Amazon Web Services

Robert Lemieux and Huseyin Gomleksizoglu, Stonebranch

Summary

This pattern explains how to integrate Stonebranch Universal Automation Center (UAC) workload orchestration with Amazon Web Services (AWS) Mainframe Modernization service. AWS Mainframe Modernization service migrates and modernizes mainframe applications to the AWS Cloud. It offers two patterns: AWS Mainframe Modernization Replatform with Micro Focus Enterprise technology and AWS Mainframe Modernization Automated Refactor with AWS Blu Age.

Stonebranch UAC is a real-time IT automation and orchestration platform. UAC is designed to automate and orchestrate jobs, activities, and workflows across hybrid IT systems, from on-premises to AWS. Enterprise clients using mainframe systems are transitioning to cloud-centric modernized infrastructures and applications. Stonebranch's tools and professional services facilitate the migration of existing schedulers and automation capabilities to the AWS Cloud.

When you migrate or modernize your mainframe programs to the AWS Cloud using AWS Mainframe Modernization service, you can use this integration to automate batch scheduling, increase agility, improve maintenance, and decrease costs.

This pattern provides instructions for integrating Stonebranch scheduler with mainframe applications migrated to the AWS Mainframe Modernization service Micro Focus Enterprise runtime. This pattern is for solutions architects, developers, consultants, migration specialists, and others working in migrations, modernizations, operations, or DevOps.

Targeted outcome

This pattern focuses on providing the following target outcomes:

- The ability to schedule, automate, and run mainframe batch jobs running in AWS Mainframe Modernization service (Microfocus runtime) from Stonebranch Universal Controller.
- Monitor the application's batch processes from the Stonebranch Universal Controller.
- Start/Restart/Rerun/Stop batch processes automatically or manually from the Stonebranch Universal Controller.

- Retrieve the results of the AWS Mainframe Modernization batch processes.
- Capture the AWS CloudWatch logs of the batch jobs in Stonebranch Universal Controller.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A Micro Focus Bankdemo application with job control language (JCL) files, and a batch process deployed in a AWS Mainframe Modernization service (Micro Focus runtime) environment
- Basic knowledge of how to build and deploy a mainframe application that runs on Micro Focus Enterprise Server
- Basic knowledge of Stonebranch Universal Controller
- Stonebranch trial license (contact Stonebranch)
- Windows or Linux Amazon Elastic Compute Cloud (Amazon EC2) instances (for example, xlarge) with a minimum of four cores, 8 GB memory, and 2 GB disk space
- Apache Tomcat version 8.5.x or 9.0.x
- Oracle Java Runtime Environment (JRE) or OpenJDK version 8 or 11
- Amazon Aurora MySQL–Compatible Edition
- Amazon Simple Storage Service (Amazon S3) bucket for export repository
- Amazon Elastic File System (Amaon EFS) for agent Stonebranch Universal Message Service (OMS) connections for high availability (HA)
- Stonebranch Universal Controller 7.2 Universal Agent 7.2 Installation Files
- AWS Mainframe Modernization task scheduling template (latest released version of the .zip file)

Limitations

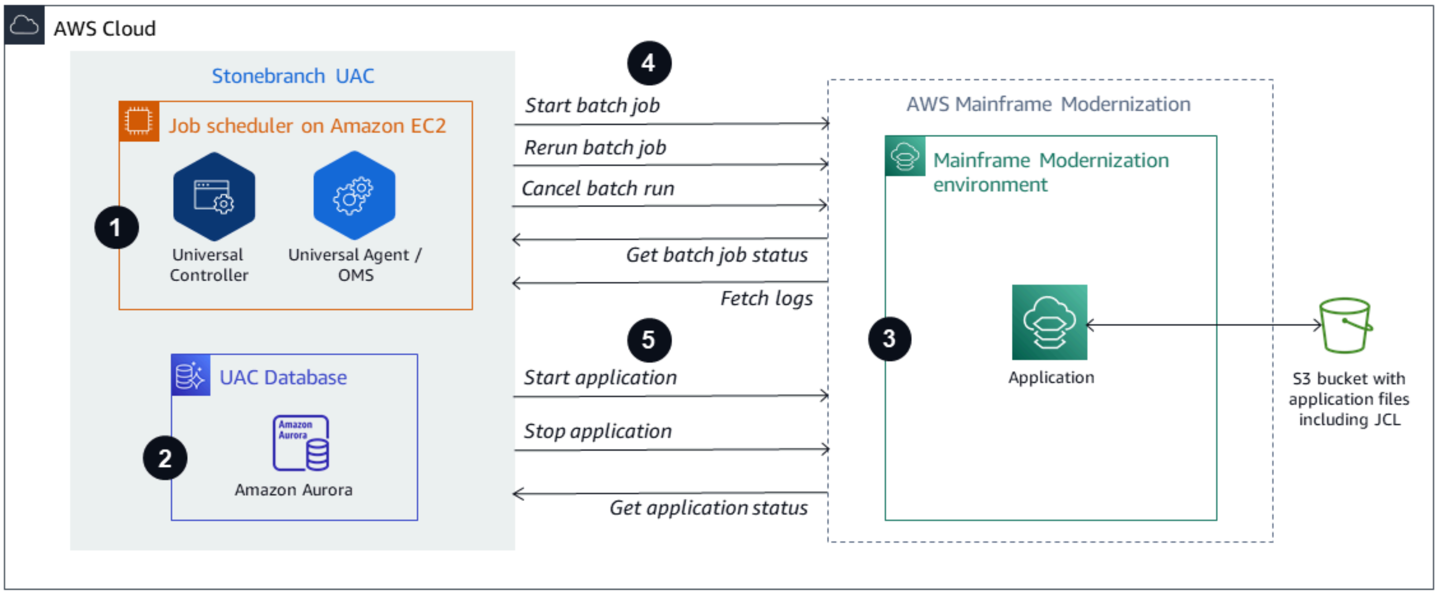
- The product and solution has been tested and compatibility validated only with OpenJDK 8 and 11.
- The aws-mainframe-modernization-stonebranch-integration task scheduling template will work only with AWS Mainframe Modernization service.
- This task scheduling template will work on only a Unix, Linux, or Windows edition of Stonebranch agents.

- Some AWS services aren't available in all AWS Regions. For Region availability, see [AWS services by Region](). For specific endpoints, see the [Service endpoints and quotas]() page, and choose the link for the service.

Architecture

Target state architecture

The following diagram shows the example AWS environment that is required for this pilot.



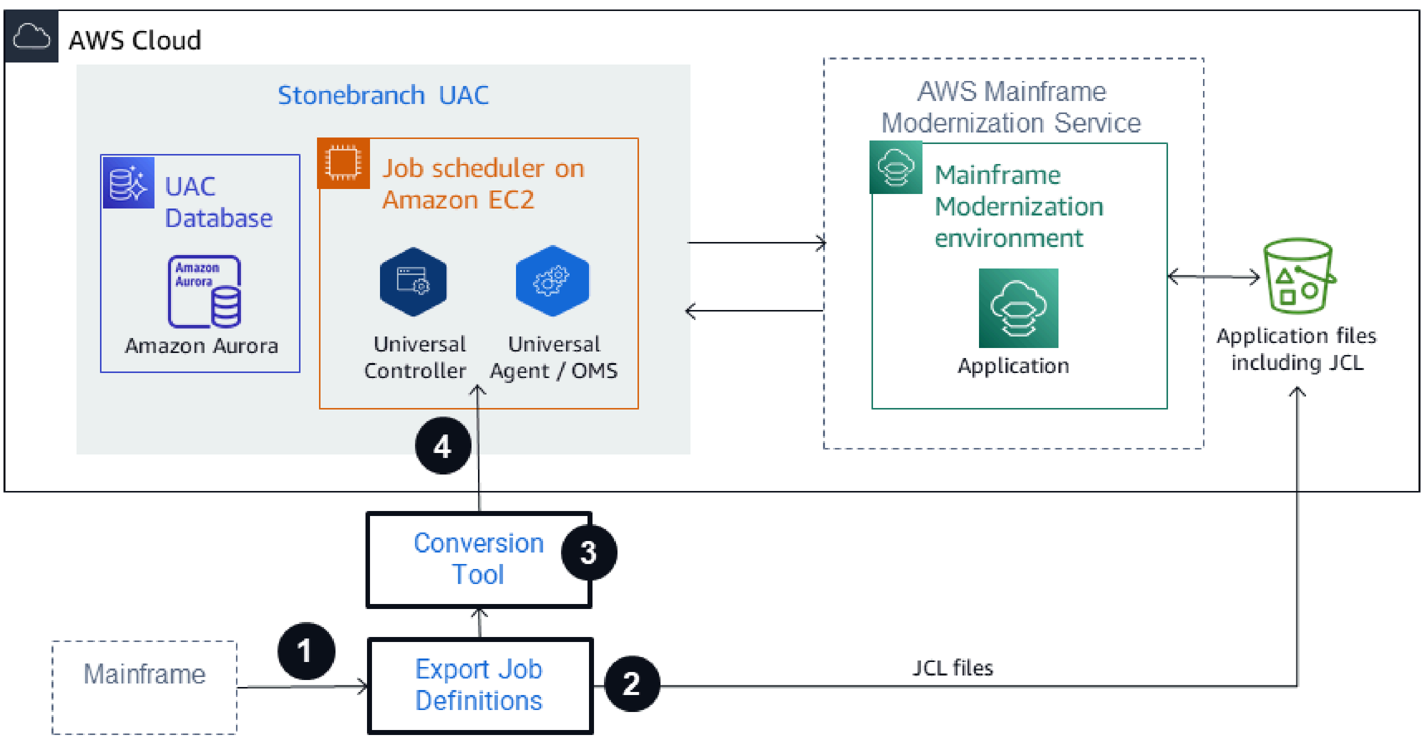
1. Stonebranch Universal Automation Center (UAC) includes two main components: Universal Controller and Universal Agents. Stonebranch OMS is used as a message bus between the controller and individual agents.
2. Stonebranch UAC Database is used by Universal Controller. The database can be MySQL, Microsoft SQL Server, Oracle, or Aurora MySQL–Compatible.
3. AWS Mainframe Modernization service – Micro Focus runtime environment with the [BankDemo application deployed](). The BankDemo application files will be stored in an S3 bucket. This bucket also contains the mainframe JCL files.
4. Stonebranch UAC can run the following functions for the batch run:
   a. Start a batch job using the JCL file name that exists in the S3 bucket linked to the AWS mainframe modernization service.
   b. Get the status of the batch job run.
   c. Wait until the batch job run is completed.

   d. Fetch logs of the batch job run.
   e. Rerun the failed batch jobs.
   f. Cancel the batch job while the job is running.

5. Stonebranch UAC can run the following functions for the application:
   a. Start Application
   b. Get Status of the Application
   c. Wait until the Application is started or stopped
   d. Stop Application
   e. Fetch Logs of Application operation

Stonebranch jobs conversion

The following diagram represents Stonebranch's job conversion process during the modernization journey. It describes how the job schedules and tasks definitions are converted into a compatible format that can run AWS Mainframe Modernization batch tasks.

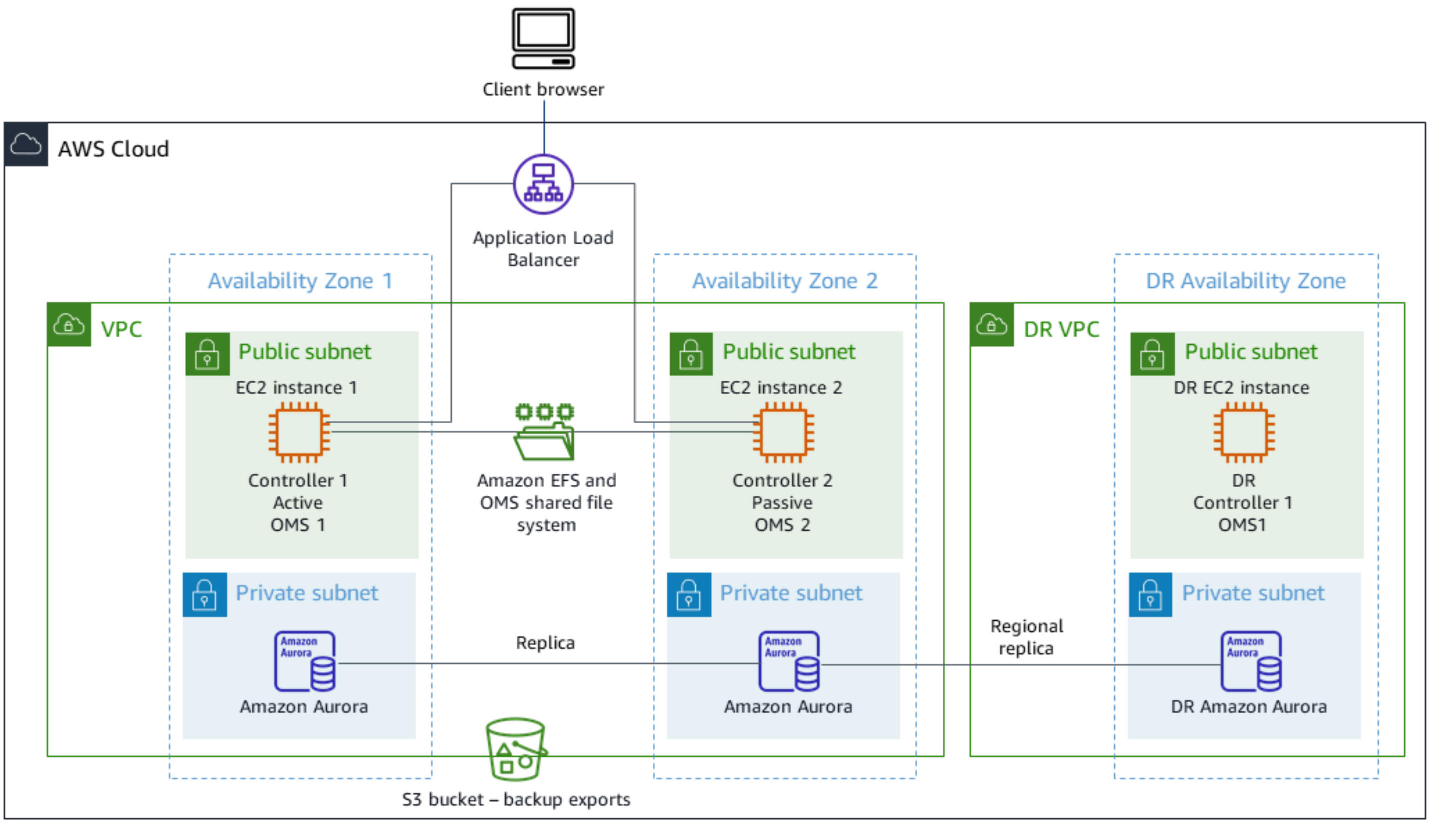


1. For the conversion process, the job definitions are exported from the existing mainframe system.
2. JCL files can be uploaded to the S3 bucket for the Mainframe Modernization application so that these JCL files can be deployed by the AWS Mainframe Modernization service.

3. The conversion tool converts the exported job definitions to UAC tasks.
4. After all the task definitions and job schedules are created, these objects will be imported to the Universal Controller. The converted tasks then run the processes in the AWS Mainframe Modernization service instead of running them on the mainframe.

Stonebranch UAC architecture

The following architecture diagram represents an active-active-passive model of high availability (HA) Universal Controller. Stonebranch UAC is deployed in multiple Availability Zones to provide high availability and support disaster recovery (DR).



Universal Controller

Two Linux servers are provisioned as Universal Controllers. Both connect to the same database endpoint. Each server houses a Universal Controller application and OMS. The most recent version of Universal Controller is used at the time it is provisioned.

The Universal Controllers are deployed in the Tomcat webapp as the document ROOT and are served on port 80. This deployment eases the configuration of the frontend load balancer.

HTTP over TLS or HTTPS is enabled using the Stonebranch wildcard certificate (for example, `https://customer.stonebranch.cloud`). This secures communication between the browser and the application.

OMS

A Universal Agent and OMS (Opswise Message Service) reside on each Universal Controller server. All deployed Universal Agents from the customer end are set up to connect to both OMS services. OMS acts as a common messaging service between the Universal Agents and the Universal Controller.

Amazon EFS mounts a spool directory on each server. OMS uses this shared spool directory to keep the connection and task information from controllers and agents. OMS works in a high-availability mode. If the active OMS goes down, the passive OMS has access to all the data, and it resumes active operations automatically. Universal Agents detect this change and automatically connect to the new active OMS.

Database

Amazon Relational Database Service (Amazon RDS) houses the UAC database, with Amazon Aurora MySQL–Compatible as its engine. Amazon RDS is helps in managing and offering scheduled backups at regular intervals. Both Universal Controller instances connect to the same database endpoint.

Load balancer

An Application Load Balancer is set-up for each instance. The load balancer directs traffic to the active controller at any given moment. Your instance domain names point to the respective load balancer endpoints.

URLs

Each of your instances has a URL, as shown in the following example.

Environment	Instance
Production	`customer.stonebranch.cloud`
Development (non-production)	`customerdev.stonebranch.cloud`
Testing (non-production)	`customertest.stonebranch.cloud`

> **Note**
>
> Non-production instance names can be set based on your needs.

High availability

High availability (HA) is the ability of a system to operate continuously without failure for a designated period of time. Such failures include, but are not limited to, storage, server communication response delays caused by CPU or memory issues, and networking connectivity.

To meet HA requirements:

- All EC2 instances, databases, and other configurations are mirrored across two separate Availability Zones within the same AWS Region.
- The controller is provisioned through an Amazon Machine Image (AMI) on two Linux servers in the two Availability Zones. For example, if you are provisioned in the Europe eu-west-1 Region, you have a Universal Controller in Availability Zone eu-west-1a and Availability Zone eu-west-1c.
- No jobs are allowed to run directly on the application servers and no data is allowed to be stored on these servers.
- The Application Load Balancer runs health checks on each Universal Controller to identify the active one and direct traffic to it. In the event that one server incurs issues, the load balancer automatically promotes the passive Universal Controller to an active state. The load balancer then identifies the new active Universal Controller instance from the health checks and starts directing traffic. The failover happens within four minutes with no job loss, and the frontend URL remains the same.
- The Aurora MySQL–Compatible database service stores Universal Controller data. For production environments, a database cluster is built with two database instances in two different Availability Zones within a single AWS Region. Both Universal Controllers use a Java Database Connectivity (JDBC) interface that points to a single database cluster endpoint. In the event that one database instance incurs issues, the database cluster endpoint dynamically points to the healthy instance. No manual intervention is required.

Backup and purge

Stonebranch Universal Controller is set to back up and purge old data following the schedule shown in the table.

Type	Schedule
Activity	7 days
Audit	90 days
History	60 days

Backup data older than the dates shown is exported to .xml format and stored in the file system. After the backup process is complete, older data is purged from the database and archived in an S3 bucket for up to one year for production instances.

You can adjust this schedule in your Universal Controller interface. However, increasing these time-frames might cause a longer downtime during maintenance.

Tools

AWS services

- AWS Mainframe Modernization is an AWS cloud-native platform that helps you modernize your mainframe applications to AWS managed runtime environments. It provides tools and resources to help you plan and implement migration and modernization.
- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon EC2 instances.
- Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud. This pattern uses Amazon Aurora MySQL–Compatible Edition.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon EC2 instances, containers, and IP addresses in one or more Availability Zones. This pattern uses an Application Load Balancer.

Stonebranch

- Universal Automation Center (UAC) is a system of enterprise workload automation products. This pattern uses the following UAC components:
  - Universal Controller, a Java web application running in a Tomcat web container, is the enterprise job scheduler and workload automation broker solution of Universal Automation Center. The Controller presents a user interface for creating, monitoring, and configuring Controller information; handles the scheduling logic; processes all messages to and from Universal Agents; and synchronizes much of the high availability operation of Universal Automation Center.
  - Universal Agent is a vendor-independent scheduling agent that collaborates with existing job scheduler on all major computing platforms, both legacy and distributed. All schedulers that run on z/Series, i/Series, Unix, Linux, or Windows are supported.
- Universal Agent is a vendor-independent scheduling agent that collaborates with existing job scheduler on all major computing platforms, both legacy and distributed. All schedulers that run on z/Series, i/Series, Unix, Linux, or Windows are supported.
- Stonebranch aws-mainframe-modernization-stonebranch-integration AWS Mainframe Modernization Universal Extension is the integration template to run, monitor and rerun batch jobs in AWS Mainframe Modernization platform.

Code

The code for this pattern is available in the aws-mainframe-modernization-stonebranch-integration GitHub repository.

Epics

Install Universal Controller and Universal Agent on Amazon EC2

Task	Description	Skills required
Download the installation files.	Download the installation from Stonebranch servers. To get the installation files, contact with Stonebranch.	Cloud architect
Launch the EC2 instance.	You will need about 3 GB of extra space for the Universal Controller and Universal	Cloud architect

Task	Description	Skills required
	Agent installations. So provide at least 30 GB of disk space for the instance. Add port 8080 to the security group so that it’s accessible.	

Task	Description	Skills required

Check prerequisites.	Before the installation, do the following:
	
	1. Install Java as described in [Downloading Java Runtime Environment]().
	`$ sudo yum -y update`
	`$ sudo yum install java-11-amazon-corretto`
	
	Be sure to use one of the supported JAVA versions. The previous command should install java-11. Check the Java version and be sure you are using version 11 before continuing.
	2. As described in [Installing Apache Tomcat](), run the following commands.
	`$ sudo yum install tomcat tomcat-admin-webapps`
	`$ sudo systemctl enable tomcat`
	`$ sudo systemctl start tomcat`
	3. Create an Amazon Aurora database as described in [Creating and connecting to an Aurora MySQL DB cluster](). Use Amazon
		Cloud administrator, Linux administrator

Task	Description	Skills required
	Aurora MySQL-Compatible Edition. Choose a Master username and Master password. Keep the default values for the rest of the settings.	

Task	Description	Skills required

Install Universal Controller.	1. Upload the `universal-controller-7.2.0.0.tar` installation file to the EC2 instance.
	2. Unarchive the installation files to a `temp` folder.
	`$ tar -xvf universal-controller-7.2.0.0.tar`
	3. Give the installation script run permission.
	`$ chmod a+x install-controller.sh`
	4. Install the controller. This example uses the following command to install Universal Controller under `/usr/share/tomcat`. Use the Amazon Aurora database that you created in the previous steps.
	`$ sudo ./install-controller.sh --tomcat-dir /usr/share/tomcat/ --controller-file universal-controller-7.2.0.0-build.145.war --dbuser admin --dbpass "****" --dbname uc --rdbms mysql --dburl`
		Cloud architect, Linux administrator

Task	Description	Skills required

	``` jdbc:mysql://database-2-instance-1.cih63miincgy.us-east-1.rds.amazonaws.com:3306/ ```
	The last line of the output of the script should be "Installation complete."
	5. Navigate to the following URL in the EC2 instance.
	``` http://<public_ip>:8080/uc ```
	6. On the login screen, enter **ops.admin** in the **Username** section, and keep the **Password** field empty.
	7. Set a new password for the `ops.admin` user.	

Task	Description	Skills required

Install Universal Agent.	1. Upload the `sb-7.2.0.1-linux-3.10-x86_64.tar.Z` installation file to the EC2 instance.
	2. Log in to the EC2 instance.
	3. Unarchive the Universal Agent installation package.
	`$ zcat sb-7.2.0.1-linux-3.10-x86_64.tar.Z | tar xvf -`
	4. Run the following command.
	`$ sudo ./unvinst --oms_servers 7878@localhost --oms_autostart yes --python yes`
	5. Create a PAM file.
	`$ cp /etc/pam.d/login /etc/pam.d/ucmd`
	6. Enable Autostart for Universal Agent.
	`$ /sbin/restorecon -v /etc/rc.d/init.d/ubrokerd`
		Cloud administrator, Linux administrator

Task	Description	Skills required
Add OMS to Universal Controller.	1. Log in to Universal Controller with the `ops.admin` user. 2. Choose the **Services** menu at the top left corner of the screen, and then choose the **OMS Servers** menu in the **System** 3. In the OMS Server Address field, type `localhost`, and then save. 4. You will see the status of the OMS server as **Connected** and the **Session Status** as **Operational**.	Universal Controller administrator

Import AWS Mainframe Modernization Universal Extension and create a task

Task	Description	Skills required
Import Integration Template.	For this step, you need the [AWS Mainframe Modernization Universal Extension](). Ensure the latest released version of the .zip file is downloaded. 1. Log in to the Universal Controller with the `ops.admin` user.	Universal Controller administrator

Task	Description	Skills required
	2. Navigate to **Services**, **Import Integration Template**. 3. Select the Integrati on Template .zip file (`aws_mainframe_mode rnization_stonebra nch_extension.zip`), and choose **Import**. After the Integration Template is imported, you will see **AWS Mainframe Modernization Tasks** under **Available Services**.	

Task	Description	Skills required
Enable resolvable credentials.	1. Navigate to **Services**, **AWS Mainframe Modernization Tasks**. 2. On the right panel, fill in the required fields: • **Name**: New Mainframe Modernization Task • **Agent**: Select the only agent (AGNT0001). Under **AWS Mainframe Modernization Details**: • **Action**: List Environme nts • **AWS Credentials**: If you have an AWS Identity and Access Managemen t (IAM) role added to the EC2 instance, you can keep this field empty. If you will use `AWSAccess KeyID` and `AWSSecret Key` , choose the icon **()** next to the field. In the **Credential Details** window that opens, enter the following information and then save. • **Name**: AWS Mainframe Modernization Credentia ls	Universal Controller administr ator

Task	Description	Skills required
	• **Runtime User**: Write the AWS access key ID in this field. • **Runtime Password**: Write the AWS secret key in this field. • **End Point**: Be sure that the endpoint has the correct AWS Region. The default is `https://m2.us-east-1.amazonaws.com` . • **Region**: Enter the Region of the AWS Mainframe Modernization service. The default is `us-east-1` . 3. Keep the default values in the rest of the fields, and save the task.	

Task	Description	Skills required
Launch the task.	1. At the top of the right panel, choose **Launch Task**. 2. In the **Confirm** window, choose **Launch**. After that, the Universal Controller Console will display a message similar to the following message. *2022-08-24 10:11:49 AM* *Successfully launched the Universal task "New Mainframe Modernization Task" with task instance sys_id 1661291493634146313NC8E38DB8OZJY.* 3. Navigate to the **Instances** If you don't see the **Instances** tab, choose the right arrow to scroll right. 4. Open the context (right-click) menu for the task instance in the list, choose **Retrieve Output**, and then choose **Submit** in the **Retrieve Output** 5. In the **Retrieve Output** window, you will see the list of environments in STDOUT.	Universal Controller administrator

Test starting a batch job

Task	Description	Skills required
Create a task for the batch job.	1. Navigate to **Services**, **AWS Mainframe Modernization Tasks**. 2. On the right panel, fill in the required fields: • **Name**: New Mainframe Modernization Task • **Agent**: Select the only agent (AGNT0001). Under **AWS Mainframe Modernization Details**: • **Action**: Start Batch (or Start Batch and Wait to run the batch job and wait until the task completes in AWS) • **AWS Credentials**: If you have an IAM role added to the EC2 instance, you can keep this field empty. If you will use `AWSAccessKeyID` and `AWSSecretKey`, choose the icon **()** next to the field. • **End Point**: Be sure that the endpoint has the correct AWS Region. The default is https://m2.us-east-1.amazonaws.com.	Universal Controller administrator

Task	Description	Skills required
	• **Region**: Enter the Region of the AWS Mainframe Modernization service. The default is `us-east-1` . • **Application**: Choose the icon next to the field **()**, and choose **Submit** in the **Refresh Applicati on Choices**. This will connect to the AWS Mainframe Moderniza tion service and return the list of applications. Now you can select the application from the dropdown list. Select the application you want to run the batch job. • **JCL File Name**: `RUNHELLO.jcl` • **Wait for Success or Failure**: If this option is selected, the task will wait until the status of the batch job is success or failure. • **Polling Interval**: This is the amount of time between each polling. • **Fetch Execution Logs**: If selected, logs will be fetched automatically	

Task	Description	Skills required
	when the batch job has completed. • **Log Format**: This is the format of the logs to be printed out. It can be Text or JSON format. 3. Keep the default values in the rest of the fields, and save the task.	

Task	Description	Skills required
Launch the task.	1. At the top of the right panel, choose **Launch Task**. 2. In the **Confirm** window, choose **Launch**. After that, the Universal Controlle r Console will display a message similar to the following message. *2022-08-24 11:11:59 AM* *Successfully launched the Universal task "Mainframe Modernization Start Batch" with task instance sys_id <sys id>.* 3. Navigate to the **Instances** If you don’t see the **Instances** tab, choose the right arrow to scroll right. 4. Open the context (right-click) menu for the task instance in the list, choose **Retrieve Output**, and then choose **Submit** in the **Retrieve Output** 5. In the **Retrieve Output** window, you will see the list of environments in STDOUT.	Universal Controller administr ator

Create a workflow for multiple tasks

Task	Description	Skills required
Copy the tasks.	1. Open the context (right click) menu for the task that you want to create copies of, and choose **Copy**. 2. In the **Copy AWS Mainframe Modernization Task** window enter the following new name for the new task: Mainframe Modernization Start Batch - *RUNAWS2.* 3. Copy the task again, using the following name: Mainframe Modernization Start Batch - *RUNAWS3.* 4. Copy with the task again, using the following name: Mainframe Modernization Start Batch - *RUNAWS4.* 5. Copy the task a final time, using the following name: Mainframe Modernization Start Batch - *FOOBAR.*	Universal Controller administrator
Update tasks.	1. Open (double-click) the Mainframe Modernization Start Batch - RUNAWS2 task, change the **JCL File Name** field to `RUNAWS2.jcl` , and save. 2. Open (double-click) the Mainframe Modernization	Universal Controller administrator

Task	Description	Skills required
	Start Batch - RUNAWS3 task, change the **JCL File Name** field to `RUNAWS3.jcl`, and save. 3. Open (double-click) the Mainframe Modernization Start Batch - RUNAWS4 task, change the **JCL File Name** field to `RUNAWS4.jcl`, and save. 4. Open (double-click) the Mainframe Modernization Start Batch - FOOBAR task, change the **JCL File Name** field to `MISSING.jcl`, and save. **This task will fail because the JCL File Name value is incorrect.**	

Task	Description	Skills required
Create a workflow.	1. Navigate to **Services**, **Workflows**. 2. On the right panel, enter **Mainframe Modernization Workflow** in the **Name** field, and save. 3. In the right panel, choose **Edit Workflow**. 4. On the **Workflow Editor Tab**, the **Add Task** button **(+)**. 5. In the **Task Find** window, choose **Search** to see all the tasks in the Universal Controller. 6. Click the icon next to Mainframe Modernization Start Batch Task, and drag the icon into an empty place in the **Workflow Editor**. 7. Repeat the same action for the other Mainframe Modernization tasks and place them as shown in the *Additional information* section. 8. Choose the **Connect** button **()**, and connect the tasks together. To connect a task with another, click in the middle of a task, and drag it to the target task.	Universal Controller administr ator

Task	Description	Skills required
	9. Connect the tasks as shown in the *Additional informati on* section, and save the workflow. 10Right-click an empty place in the Workflow Editor, choose **Launch Workflow**, and then choose **OK**.	
Check the status of the workflow.	1. On the left menu, choose the **Activity** 2. In the middle of the window, choose **Start**. You will see the list of task instances in the list. 3. Open (double-click) Mainframe Moderniza tion Workflow in the list, or open the context (right-click) menu and choose **Workflow Task Commands**, **View Workflow**. You will see the tasks as shown in the Additional information section. The second task was expected to fail because you used a missing JCL file.	Univeral Controller administr ator

Troubleshoot failed batch jobs and rerun

Task	Description	Skills required
Fix the failed task and rerun.	1. Open (double-click) the failed task to see the error for the task. 2. You have two options for fixing the failed task. • Fix the JCL file name, and set it to `FOOBAR.jcl` . • Add the correct JCL file name to the **JCL File Name (Temp)**. This field will overwrite the **JCL File Name** field. For this pilot, choose the second option, and save the task instance. 3. In the **Workflow Monitor**, open the context (right-click) menu for the failed task, and choose **Commands**, **Re-run**. 4. After that, all the tasks will complete successfully.	Universal Controller administrator

Create Start Application and Stop Application tasks

Task	Description	Skills required
Create the Start Application action.	1. Navigate to **Services**, **AWS Mainframe Modernization Tasks**. 2. On the right panel, fill in the required fields. • **Name**: Mainframe Modernization Start Application • **Agent**: Select the only agent (AGNT0001) Under **AWS Mainframe Modernization Details**: • **Action**: Start Application • **AWS Credentials**: If you have an IAM role added to the EC2 instance, you can keep this field empty. If you will use `AWSAccessKeyID` and `AWSSecretKey`, select the credential that you created before. • **End Point**: Be sure that the endpoint has the correct Region. The default is `https://m2.us-east-1.amazonaws.com`. • **Region**: Enter the Region of the AWS Mainframe	Universal Controller administrator

Task	Description	Skills required
	Modernization service. The default is `us-east-1` . • **Application**: Choose the icon next to the field **()**, and choose **Submit** in the **Refresh Application Choices**. This will connect to the AWS Mainframe Modernization service and return the list of applications. Now you can select the application from the dropdown list. Select the application you want to run the batch job. • **Wait for Success or Failure**: If this option is selected, the task will wait until the status of the batch job is success or failure. • **Polling Interval**: This is the amount of time between each polling. • **Fetch Execution Logs**: If selected, logs will be fetched automatically when the batch job has completed. • **Log Format:** This is the format of the logs to be	

Task	Description	Skills required
	printed out. It can be Text or JSON format. 3. Keep the default values in the rest of the fields, and save the task. 4. Now copy this task and create a task for Stop Application. Change the name to **Mainframe Moderinization Stop Application**, and change the action to **Stop Application**.	

Create a Cancel Batch Execution task

Task	Description	Skills required
Create the Cancel Batch action.	1. Navigate to **Services**, **AWS Mainframe Modernization Tasks**. 2. On the right panel, fill in the required fields. • **Name**: Mainframe Modernization Cancel Batch Execution • **Agent**: Select the only agent (AGNT0001) Under **AWS Mainframe Modernization Details**: • **Action**: Cancel Batch Execution	

Task	Description	Skills required
	• **AWS Credentials**: If you have an IAM role added to the EC2 instance, you can keep this field empty. If you will use `AWSAccessKeyID` and `AWSSecretKey` , select the credential that you created before. • **End Point**: Be sure that the endpoint has the correct Region. The default is `https://m2.us-east-1.amazonaws.com` . • **Region**: Enter the Region of the AWS Mainframe Modernization service. The default is `us-east-1` . • **Application**: Choose the icon next to the field **()**, and choose **Submit** in the **Refresh Application Choices**. This will connect to the AWS Mainframe Modernization service and return the list of applications. Now you can select the application from the dropdown list. Select the application you want to run the batch job.	

Task	Description	Skills required
	• **Wait for Success or Failure**: If this option is selected, the task will wait until the status of the batch job is success or failure. • **Polling Interval**: This is the amount of time between each polling. • **Fetch Execution Logs**: If selected, logs will be fetched automatically when the batch job has completed. • **Log Format**: This is the format of the logs to be printed out. It can be Text or JSON format. 3. Keep the default values in the rest of the fields, and save the task.	

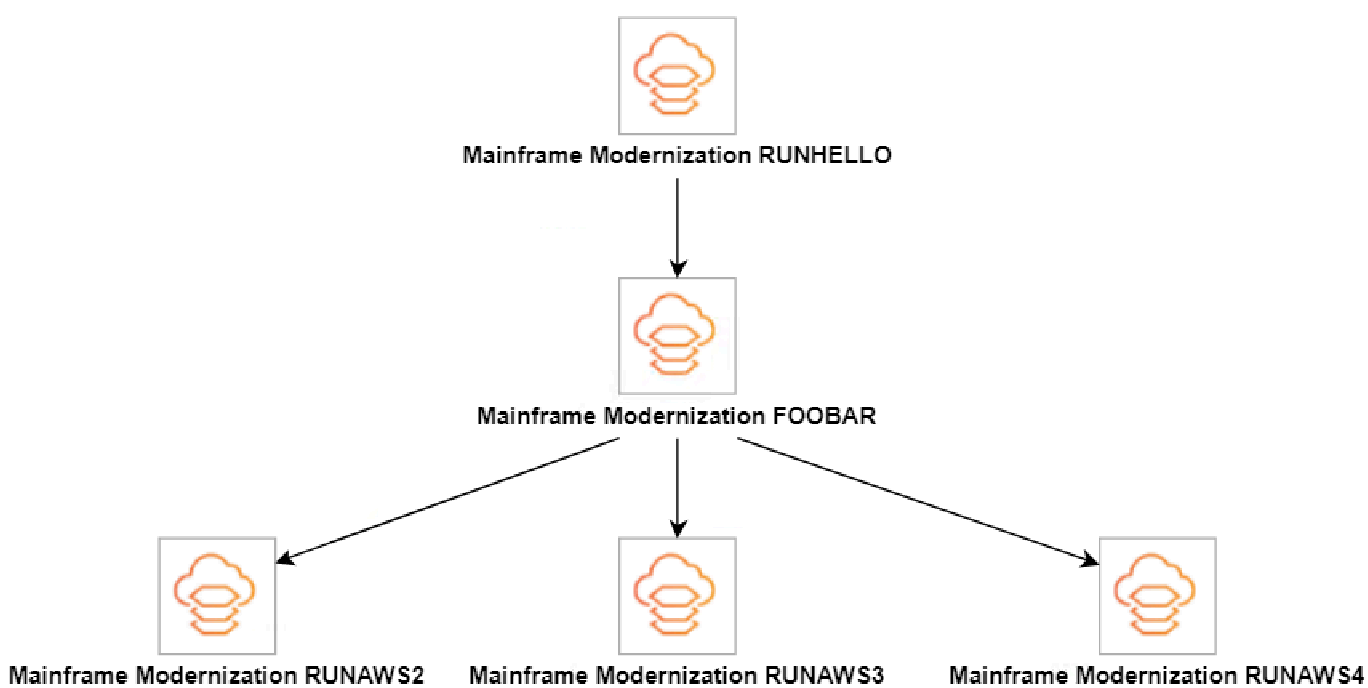
Related resources

- Universal Controller
- Universal Agent
- LDAP Settings
- SAML Single Sign-On
- Xpress Conversion Tool

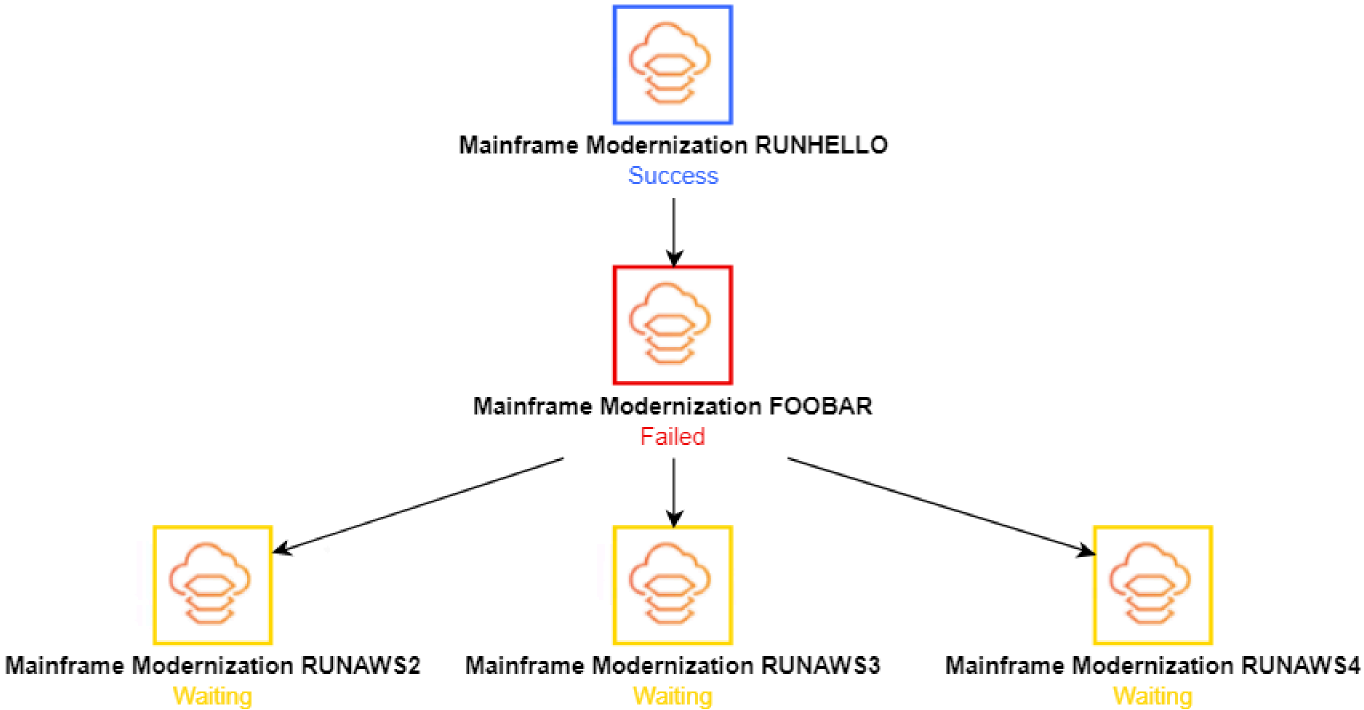
Additional information

Icons in the Workflow Editor

All tasks connected



Workflow status



Migrate and replicate VSAM files to Amazon RDS or Amazon MSK using Connect from Precisely

Prachi Khanna and Boopathy GOPALSAMY, Amazon Web Services

Summary

This pattern shows you how to migrate and replicate Virtual Storage Access Method (VSAM) files from a mainframe to a target environment in the AWS Cloud by using Connect from Precisely. The target environments covered in this pattern include Amazon Relational Database Service (Amazon RDS) and Amazon Managed Streaming for Apache Kafka (Amazon MSK). Connect uses change data capture (CDC) to continuously monitor updates to your source VSAM files and then transfer these updates to one or more of your AWS target environments. You can use this pattern to meet your application modernization or data analytics goals. For example, you can use Connect to migrate your VSAM application files to the AWS Cloud with low latency, or migrate your VSAM data to an AWS data warehouse or data lake for analytics that can tolerate synchronization latencies that are higher than required for application modernization.

Prerequisites and limitations

Prerequisites

- IBM z/OS V2R1 or later
- CICS Transaction Server for z/OS (CICS TS) V5.1 or later (CICS/VSAM data capture)
- IBM MQ 8.0 or later
- Compliance with z/OS security requirements (for example, APF authorization for SQData load libraries)
- VSAM recovery logs turned on
- (Optional) CICS VSAM Recovery Version (CICS VR) to automatically capture CDC logs
- An active AWS account
- An Amazon Virtual Private Cloud (VPC) with a subnet that's reachable by your legacy platform
- A VSAM Connect license from Precisely

Limitations

- Connect doesn't support automatic target table creation based on source VSAM schemas or copybooks. You must define the target table structure for the first time.

- For non-streaming targets such as Amazon RDS, you must specify the conversion source to target mapping in the Apply Engine configuration script.
- Logging, monitoring, and alerting functions are implemented through APIs and require external components (such as Amazon CloudWatch) to be fully operational.

Product versions

- SQData 40134 for z/OS
- SQData 4.0.43 for the Amazon Linux Amazon Machine Image (AMI) on Amazon Elastic Compute Cloud (Amazon EC2)

Architecture

Source technology stack

- Job Control Language (JCL)
- z/OS Unix shell and Interactive System Productivity Facility (ISPF)
- VSAM utilities (IDCAMS)

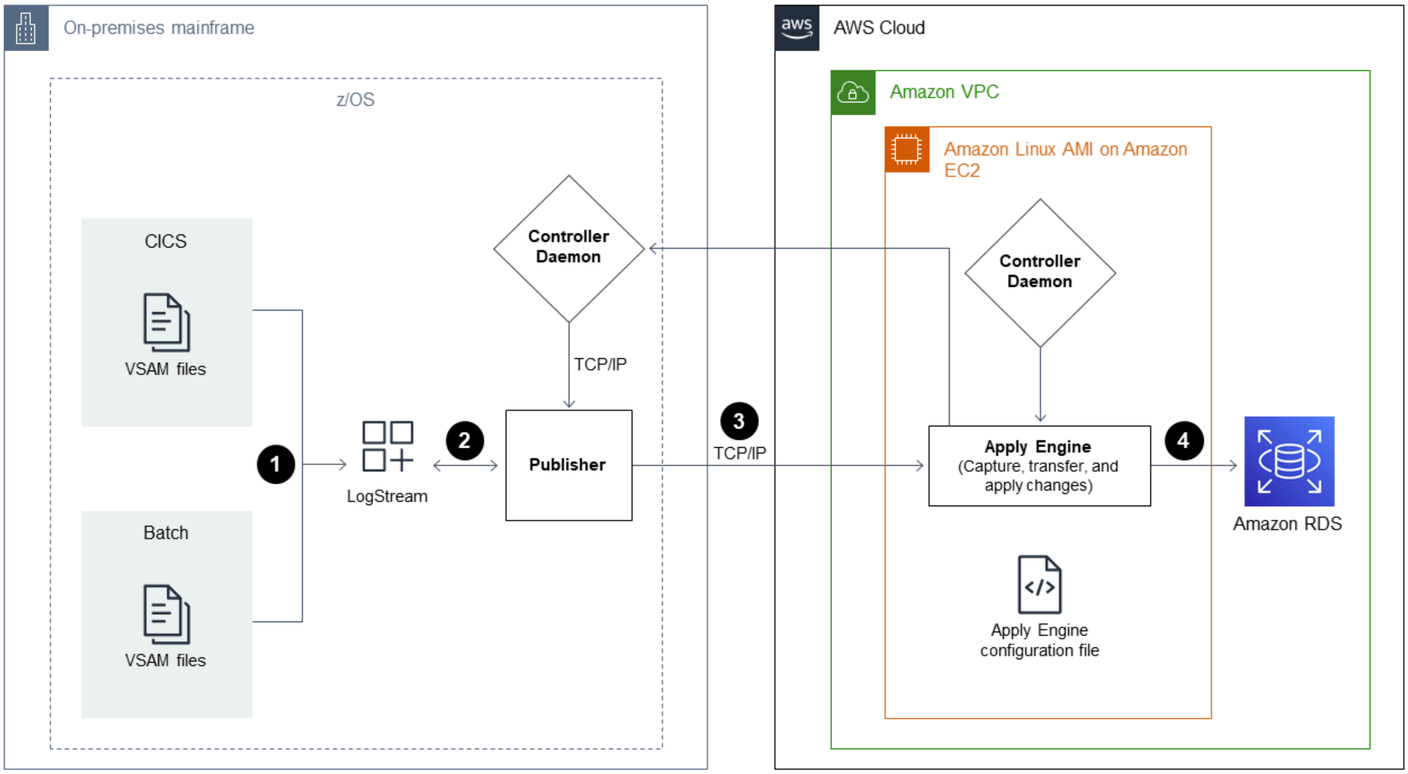
Target technology stack

- Amazon EC2
- Amazon MSK
- Amazon RDS
- Amazon VPC

Target architecture

Migrating VSAM files to Amazon RDS

The following diagram shows how to migrate VSAM files to a relational database, such as Amazon RDS, in real time or near real time by using the CDC agent/publisher in the source environment (on-premises mainframe) and the Apply Engine in the target environment (AWS Cloud).



The diagram shows the following batch workflow:

1. Connect captures changes to a file by comparing VSAM files from backup files to identify changes and then sends the changes to the logstream.
2. The publisher consumes data from the system logstream.
3. The publisher communicates captured data changes to a target engine through TCP/IP. The Controller Daemon authenticates communication between the source and target environments.
4. The Apply Engine in the target environment receives the changes from the Publisher agent and applies them to a relational or non-relational database.

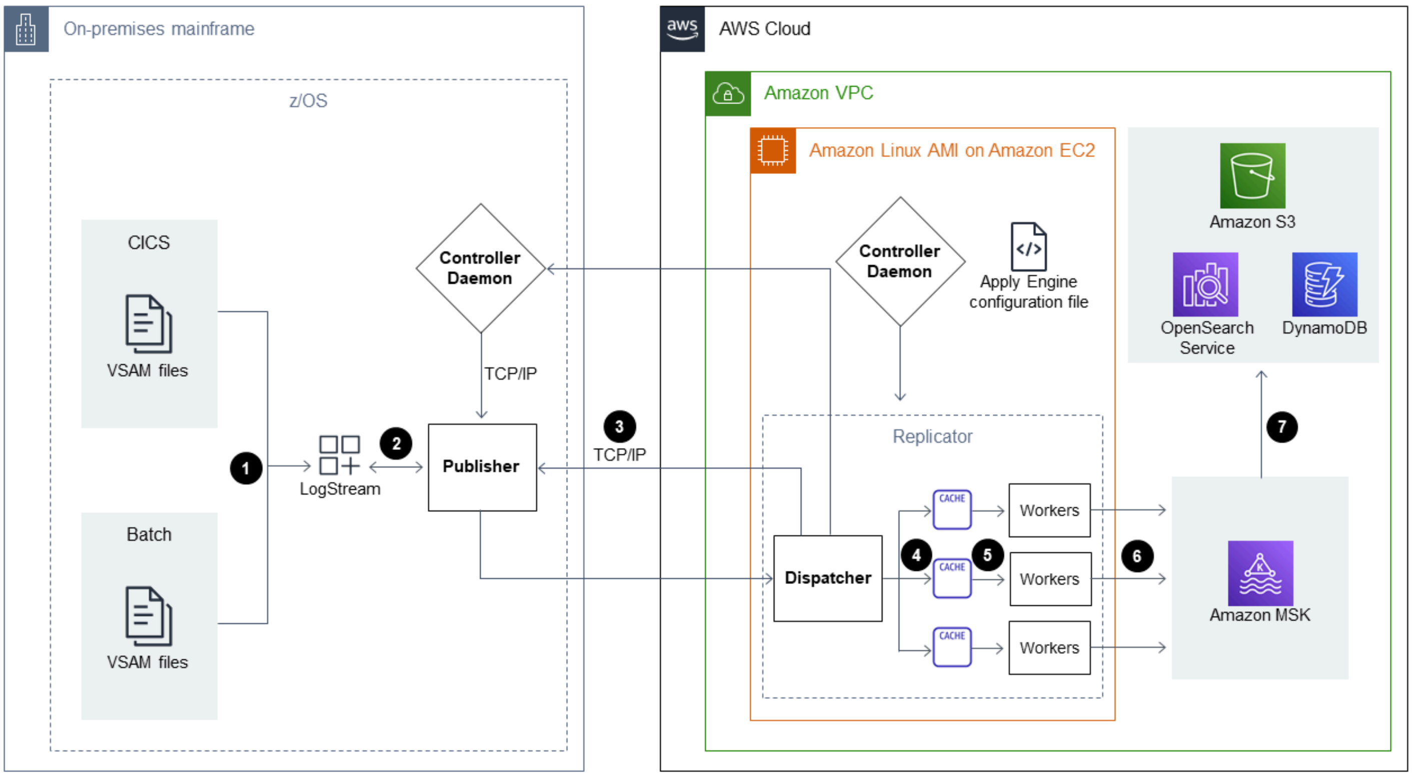
The diagram shows the following online workflow:

1. Connect captures changes in the online file by using a log replicate and then streams captured changes to a logstream.
2. The publisher consumes data from the system logstream.
3. The publisher communicates captured data changes to the target engine through TCP/IP. The Controller Daemon authenticates communication between the source and target environments.

4. The Apply Engine in the target environment receives the changes from the Publisher agent and then applies them to a relational or non-relational database.

Migrating VSAM files to Amazon MSK

The following diagram shows how to stream VSAM data structures from a mainframe to Amazon MSK in high-performance mode and automatically generate JSON or AVRO schema conversions that integrate with Amazon MSK.



The diagram shows the following batch workflow:

1. Connect captures changes to a file by using CICS VR or by comparing VSAM files from backup files to identify changes. Captured changes are sent to the logstream.
2. The publisher consumes data from the system logstream.
3. The publisher communicates captured data changes to the target engine through TCP/IP. The Controller Daemon authenticates communication between the source and target environments.
4. The Replicator Engine that's operating in parallel processing mode splits the data to a unit of work cache.
5. Worker threads capture the data from the cache.

6. Data is published to Amazon MSK topics from the worker threads.
7. Users apply changes from Amazon MSK to targets such as Amazon DynamoDB, Amazon Simple Storage Service (Amazon S3), or Amazon OpenSearch Service by using connectors.

The diagram shows the following online workflow:

1. Changes in the online file are captured by using a log replicate. Captured changes are streamed to the logstream.
2. The publisher consumes data from the system logstream.
3. The publisher communicates captured data changes to the target engine through TCP/IP. The Controller Daemon authenticates communication between the source and target environments.
4. The Replicator Engine that's operating in parallel processing mode splits the data to a unit of work cache.
5. Worker threads capture the data from the cache.
6. Data is published to Amazon MSK topics from the worker threads.
7. Users apply changes from Amazon MSK to targets such as DynamoDB, Amazon S3, or OpenSearch Service by using connectors.

Tools

- Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that helps you build and run applications that use Apache Kafka to process streaming data.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.

Epics

Prepare the source environment (mainframe)

Task	Description	Skills required
Install Connect CDC 4.1.	1. Contact the Precisely Support team to obtain a license and installation packages.	IBM Mainframe Developer/ Admin

Task	Description	Skills required
	2. Use example JCLs to install Connect CDC 4.1. For instructions, see [Install Connect CDC (SQData) using JCL](#) in the Precisely documentation. 3. Run the `SETPROG APF` command to authorize the Connect load libraries SQDATA.V4nnn.LOADLIB.	
Set up the zFS directory.	To set up a zFS directory, follow the instructions from [zFS variable directories](#) in the Precisely documentation. **Note** Controller Daemon and Capture/Publisher agent configurations are stored in the z/OS UNIX Systems Services file system (referred to as zFS). The Controller Daemon, Capture, Storage, and Publisher agents require a predefined zFS directory structure for storing a small number of files.	IBM Mainframe Developer/Admin

Task	Description	Skills required
Configure TCP/IP ports.	To configure TCP/IP ports, follow the instructions from [TCP/IP ports](#) in the Precisely documentation. **Note** The Controller Daemon requires TCP/IP ports on source systems. The ports are referenced by the engines on the target systems (where captured change data is processed).	IBM Mainframe Developer/Admin

Task	Description	Skills required
Create a z/OS logstream.	To create a z/OS logstream, follow the instructions from Create z/OS system logStreams in the Precisely documentation. **Note** Connect uses the logstream to capture and stream data between your source environment and target environment during migration. For an example JCL that creates a z/OS LogStream, see Create z/OS system logStreams in the Precisely documentation.	IBM Mainframe Developer
Identify and authorize IDs for zFS users and started tasks.	Use RACF to grant access to the OMVS zFS file system. For an example JCL, see Identify and authorize zFS user and started task IDs in the Precisely documentation.	IBM Mainframe Developer/ Admin

Task	Description	Skills required
Generate z/OS public/private keys and the authorized key file.	Run the JCL to generate the key pair. For an example, see *Key pair example* in the *Additional information* section of this pattern. For instructions, see Generate z/OS public and private keys and authorized key file in the Precisely documentation.	IBM Mainframe Developer/ Admin

Activate the CICS VSAM Log Replicate and attach it to the logstream.	Run the following JCL script:
	
	`//STEP1 EXEC PGM=IDCAMS`
	`//SYSPRINT DD SYSOUT=*`
	`//SYSIN DD *`
	`ALTER SQDATA.CICS.FILEA -`
	`LOGSTREAMID(SQDATA.VSAMCDC.LOG1) -`
	`LOGREPLICATE`
		IBM Mainframe Developer/ Admin

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Activate the VSAM File Recovery Log through an FCT.</td><td>Modify the File Control Table (FCT) to reflect the following parameter changes:
<pre>
 Configure FCT Parms
   CEDA ALT FILE(name)
 GROUP(groupname)
   DSNAME(data set
 name)
   RECOVERY(NONE|BACK
OUTONLY|ALL)
   FWDRECOVLOG(NO|1-9
9)
   BACKUPTYPE(STATIC|
DYNAMIC)
   RECOVERY PARAMETERS
   RECOVery : None |
 Backoutonly | All
   Fwdrecovlog : No |
 1-99
   BAckuptype : Static
 | Dynamic
</pre></td><td>IBM Mainframe Developer/ Admin</td></tr>
<tr><td>Set up CDCzLog for the Publisher agent.</td><td>1. Create the CDCzLog Publisher CAB file.
2. Encrypt the published data.
3. Prepare the CDCzLog Publisher Runtime JCL.</td><td>IBM Mainframe Developer/ Admin</td></tr>
</table>

Task	Description	Skills required
Activate the Controller Daemon.	1. Open the ISPF panel and run the following command to open the Precisely menu: `EXEC 'SQDATA.V4nnnnn.ISPFLIB(SQDC$STA)' 'SQDATA.V4nnnnn'` 2. To set up the Controller Daemon, choose option 2 from the menu.	IBM Mainframe Developer/Admin
Activate the publisher.	1. Open the ISPF panel and run the following command to open the Precisely menu: `EXEC 'SQDATA.V4nnnnn.ISPFLIB(SQDC$STA)' 'SQDATA.V4nnnnn'` 2. To set up the publisher, choose option 3 from the menu and **I** for insert.	IBM Mainframe Developer/Admin

Task	Description	Skills required
Activate the logstream.	1. Open the ISPF panel and run the following command to open the Precisely menu: `EXEC 'SQDATA.V4nnnnn.ISPFLIB(SQDC$STA)' 'SQDATA.V4nnnnn'` 2. To set up the logstream, choose option 4 from the menu and **I** for insert. Then, enter the name of the logstream created in the preceding steps.	IBM Mainframe Developer/Admin

Prepare the target environment (AWS)

Task	Description	Skills required
Install Precisely on an EC2 instance.	To install Connect from Precisely on the Amazon Linux AMI for Amazon EC2, follow the instructions from [Install Connect CDC (SQData) on UNIX](#) in the Precisely documentation.	General AWS
Open TCP/IP ports.	To modify the security group to include the Controller Daemon ports for inbound and outbound access, follow the instructions from [TCP/IP](#) in the Precisely documentation.	General AWS

Task	Description	Skills required
Create file directories.	To create file directories, follow the instructions from [Prepare target apply environment](#) in the Precisely documentation.	General AWS

Create the Apply Engine configuration file.	Create the Apply Engine configuration file in the working directory of the Apply Engine. The following example configuration file shows Apache Kafka as the target:
	`builtin.features=SASL_SCRAM`
	`  security.protocol=SASL_SSL`
	`  sasl.mechanism=SCRAM-SHA-512`
	`  sasl.username=`
	`  sasl.password=`
	`  metadata.broker.list=`
	Note
	For more information, see [Security](#) in the Apache Kafka documentation.
		General AWS

Task	Description	Skills required
Create scripts for Apply Engine processing.	Create the scripts for the Apply Engine to process source data and replicate source data to the target. For more information, see Create an apply engine script in the Precisely documentation.	General AWS
Run the scripts.	Use the `SQDPARSE` and `SQDENG` commands to run the script. For more information, see Parse a script for zOS in the Precisely documentation.	General AWS

Validate the environment

Task	Description	Skills required
Validate the list of VSAM files and target tables for CDC processing.	1. Validate VSAM files, including replication logs, recovery logs, FCT parameters, and the logstream. 2. Validate target database tables, including whether the tables are created as per the required schema definition, table access, and other criteria.	General AWS, Mainframe
Verify that the Connect CDC SQData product is linked.	Run a testing job and verify that the return code from this job is 0 (Successful).	General AWS, Mainframe

Task	Description	Skills required
	Note Connect CDC SQData Apply Engine status messages should show active connectio n messages.	

Run and validate test cases (Batch)

Task	Description	Skills required
Run the batch job in the mainframe.	Run the batch application job using a modified JCL. Include steps in the modified JCL that do the following: 1. Take a backup of the data files. 2. Compare the backup file with the modified data files, generate the delta file, and then note the delta record count from the messages. 3. Push the delta file to the z/ OS logstream. 4. Run the JCL. For an example JCL, see [Prepare file compare capture JCL](#) in the Precisely documenta tion.	General AWS, Mainframe

Task	Description	Skills required
Check the logstream.	Check the logstream to confirm that you can see the change data for the completed mainframe batch job.	General AWS, Mainframe
Validate the counts for the source delta changes and target table.	To confirm the records are tallied, do the following: 1. Gather the source delta count from the batch JCL messages. 2. Monitor the Apply Engine for record level counts of the number of records inserted, updated, or deleted in the VSAM file. 3. Query the target table for record counts. 4. Compare and tally all the different record counts.	General AWS, Mainframe

Run and validate test cases (Online)

Task	Description	Skills required
Run the online transaction in a CICS region.	1. Run the online transaction to validate the test case. 2. Validate the transaction execution code (RC=0 – Success).	IBM Mainframe Developer

Task	Description	Skills required
Check the logstream.	Confirm that the logstream is populated with specific record level changes.	AWS Mainframe Developer
Validate the count in the target database.	Monitor the Apply Engine for record level counts.	Precisely, Linux
Validate the record counts and data records in the target database.	Query the target database to validate the record counts and data records.	General AWS

Related resources

- VSAM z/OS (Precisely documentation)
- Apply engine (Precisely documentation)
- Replicator engine (Precisely documentation)
- The log stream (IBM documentation)

Additional information

Configuration file example

This is an example configuration file for a logstream where the source environment is a mainframe and the target environment is Amazon MSK:

```
 -- JOBNAME -- PASS THE SUBSCRIBER NAME
 -- REPORT  progress report will be produced after "n" (number) of Source records
processed.

 JOBNAME VSMTOKFK;
 --REPORT EVERY 100;
 -- Change Op has been ‘I’ for insert, ‘D’ for delete , and ‘R’ for Replace. For RDS
it is 'U' for update
 -- Character Encoding on z/OS is Code Page 1047, on Linux and UNIX it is Code Page
819 and on Windows, Code Page 1252
```

```
 OPTIONS
 CDCOP('I', 'U', 'D'),
 PSEUDO NULL = NO,
 USE AVRO COMPATIBLE NAMES,
 APPLICATION ENCODING SCHEME = 1208;

 --       SOURCE DESCRIPTIONS

 BEGIN GROUP VSAM_SRC;
 DESCRIPTION COBOL ../copybk/ACCOUNT AS account_file;
 END GROUP;

 --       TARGET DESCRIPTIONS

 BEGIN GROUP VSAM_TGT;
 DESCRIPTION COBOL ../copybk/ACCOUNT AS account_file;
 END GROUP;

 --       SOURCE DATASTORE (IP & Publisher name)

 DATASTORE cdc://10.81.148.4:2626/vsmcdct/VSMTOKFK
 OF VSAMCDC
 AS CDCIN
 DESCRIBED BY GROUP VSAM_SRC ACCEPT ALL;

 --       TARGET DATASTORE(s) - Kafka and topic name

 DATASTORE 'kafka:///MSKTutorialTopic/key'
 OF JSON
 AS CDCOUT
 DESCRIBED BY GROUP VSAM_TGT FOR INSERT;

 --       MAIN SECTION

 PROCESS INTO
 CDCOUT
 SELECT
 {
 SETURL(CDCOUT, 'kafka:///MSKTutorialTopic/key')
 REMAP(CDCIN, account_file, GET_RAW_RECORD(CDCIN, AFTER), GET_RAW_RECORD(CDCIN,
BEFORE))
 REPLICATE(CDCOUT, account_file)
 }
```

```
  FROM CDCIN;
```

Key pair example

This an example of how to run the JCL to generate the key pair:

```
//SQDUTIL EXEC PGM=SQDUTIL //SQDPUBL DD DSN=&USER..NACL.PUBLIC, //
 DCB=(RECFM=FB,LRECL=80,BLKSIZE=21200), // DISP=(,CATLG,DELETE),UNIT=SYSDA, //
 SPACE=(TRK,(1,1)) //SQDPKEY DD DSN=&USER..NACL.PRIVATE, //
 DCB=(RECFM=FB,LRECL=80,BLKSIZE=21200), // DISP=(,CATLG,DELETE),UNIT=SYSDA, //
 SPACE=(TRK,(1,1)) //SQDPARMS DD  keygen //SYSPRINT DD SYSOUT= //SYSOUT DD SYSOUT=* //
SQDLOG DD SYSOUT=* //*SQDLOG8 DD DUMMY
```

Modernize the CardDemo mainframe application by using AWS Transform

Santosh Kumar Singh and Cheryl du Preez, Amazon Web Services

Summary

AWS Transform is designed to accelerate the modernization of mainframe applications. It uses generative AI to streamline the mainframe modernization process. It automates complex tasks, such as: legacy code analysis, mainframe documentation, business rules extraction, monolithic applications decomposition into business domain, and code refactoring. It accelerates modernization projects by automating complex tasks, such as application analysis and migration sequence planning. When decomposing monolithic applications, AWS Transform intelligently sequences the mainframe application transformation, which helps you to transform business functions in parallel. AWS Transform can accelerate decision making and enhance operational agility and migration efficiency.

This pattern offers step-by-step instructions to help you test the mainframe modernization capabilities of AWS Transform by using CardDemo, which is a sample open source mainframe application.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS IAM Identity Center, enabled
- Permissions that allow administrators to enable AWS Transform
- Permissions that allow administrators to accept Amazon Simple Storage Service (Amazon S3) connection requests for the AWS Transform web application

Limitations

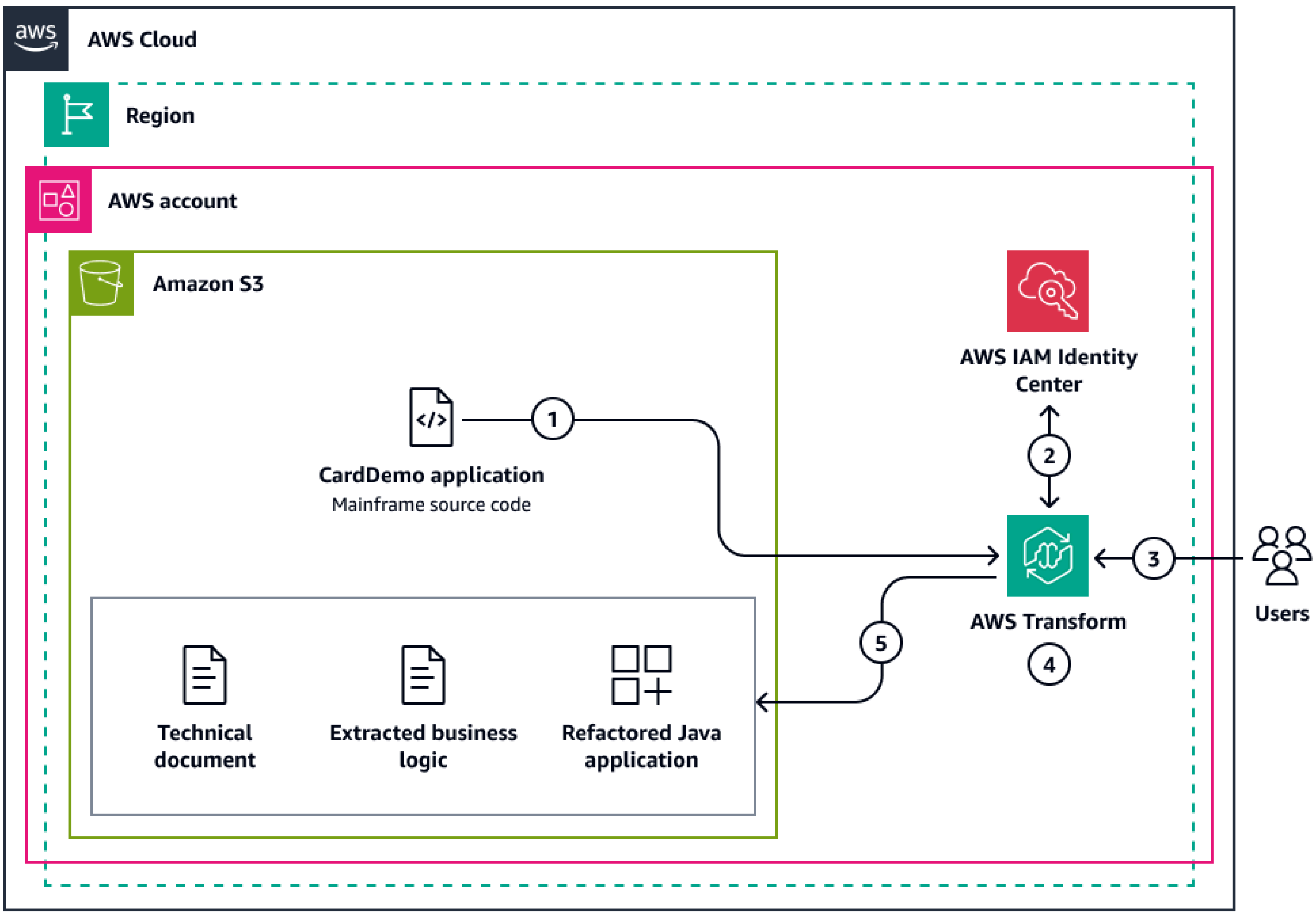
- AWS Transform is available only in some AWS Regions. For a complete list of supported Regions, see Supported Regions for AWS Transform.
- AWS Transform supports code analysis, document generation, business rules extraction, decomposition, and refactoring from Common Business-Oriented Language (COBOL) to Java. For

more information, see Capabilities and key features and Supported file types for transformation of mainframe applications.

- There is a service quota for mainframe transformation capabilities in AWS Transform. For more information, see Quotas for AWS Transform.
- In order to collaborate on a shared workspace, all users must be registered users of the same instance of AWS IAM Identity Center that is associated with your instance of the AWS Transform web application.
- The Amazon S3 bucket and AWS Transform must be in the same AWS account and Region.

Architecture

The following diagram shows the architecture that you set up in this pattern.



The diagram shows the following workflow:

1. AWS Transform uses a connector to access the CardDemo mainframe application, which is stored in an Amazon S3 bucket.
2. AWS Transform uses AWS IAM Identity Center to manage user access and authentication. The system implements multiple layers of security controls for authentication, authorization, encryption, and access management to help protect code and artifacts during processing. Users interact with the AWS Transform agent through a chat interface. You can provide instructions to the AI agent for specific tasks in English. For more information, see Human in the loop (HITL) in the AWS Transform documentation.
3. The AI agent interprets the user's instructions, creates a job plan, divides the job into executable tasks, and actions it autonomously. Users can review and approve the transformation. Transformation tasks include the following:
   - **Code analysis –** AWS Transform analyzes the code in each file for details such as file name, file type, lines of code, and their paths. The agent analyzes the source code, runs classifications, creates dependency mappings, and identifies any missing artifacts. It also identifies duplicate components.
   - **Document generation** – AWS Transform generates documentation for the mainframe application. By analyzing the code, it can automatically create detailed documentation of the application programs, including descriptions of the business logic, flows, integrations, and dependencies present in your legacy systems.
   - **Business logic extraction**– AWS Transform analyzes COBOL programs to document their core business logic, to help you understand the fundamental business logic.
   - **Code decomposition** – AWS Transform decomposes the code into domains that account for dependencies between programs and components. Grouping related files and programs within the same domain improves organization and helps preserve the application's logical structure when breaking it down into smaller components.
   - **Migration wave planning –** Based on the domains you created during the decomposition phase, AWS Transform generates a migration wave plan with recommended modernization order.
   - **Code refactoring –** AWS Transform refactors the code in all or selected domain files into Java code. The goal of this step is to preserve the critical business logic of the application while refactoring it to a modernized, cloud-optimized Java application.
4. AWS Transform stores the refactored code, generated documents, associated artifacts, and runtime libraries in your Amazon S3 bucket. You can do the following:
   - Access the runtime folder in your Amazon S3 bucket.

- Build and deploy the application by following the Build and deploy your modernized application post-refactoring in the AWS Transform documentation.
- Through the chat interface, request and download a sample AWS CloudFormation, AWS Cloud Development Kit (AWS CDK), or Hashicorp Terraform template. These templates can help you deploy the AWS resources that are necessary to support the refactored application.
- Use Reforge to improve the quality of refactored code by using large language models (LLMs). The refactoring engine preserves the functional equivalence of COBOL while transforming it into Java code. Reforge is an optional step that is available after the transformation. This step uses LLMs to restructure the code to closely resemble native Java, which can improve readability and maintainability. Reforge also adds human-readable comments to help you understand the code, and it implements modern coding patterns and best practices.

Tools

AWS services

- AWS Transform uses agentic AI to help you accelerate the modernization of legacy workloads, such as .NET, mainframe, and VMware workloads.
- AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to your AWS accounts and cloud applications.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code repository

You can use the open source AWS CardDemo mainframe application as a sample application to get started with mainframe modernization.

Best practices

- **Start small –** Begin with small, less complex code (15,000–20,000 lines of code) to get an understanding of how AWS Transform analyzes and transforms mainframe applications.
- **Combine with human expertise –** Use AWS Transform as an accelerator while applying human expertise for optimal results.
- **Review and test thoroughly –** Always review the transformed code carefully and run comprehensive tests to validate the functional equivalency after transformation.

- **Provide feedback** – To provide feedback and suggestions for improvement, use the **Send feedback** button in the AWS Management Console or create a case with AWS Support. For more information, see Creating a support case. Your input is valuable for service enhancements and future development.

Epics

Prepare the mainframe application

Task	Description	Skills required
Create a bucket.	Create an Amazon S3 bucket in the same AWS account and Region where AWS Transform is enabled. You use this bucket to store the mainframe application code, and AWS Transform uses this bucket to store the generated documents, refactored code and other files associated with the transformation. For instructions, see Creating a bucket in the Amazon S3 documentation.	General AWS
Prepare the sample mainframe application.	1. Enter the following command to clone the CardDemo repository to your local workstation: `git clone https://github.com/aws-samples/aws-mainframe-modernization-carddemo.git`	App developer, DevOps engineer

Task	Description	Skills required
	2. Create a new folder named `carddemo`. 3. Copy the `app` folder, which contains the mainframe source code, from the cloned repository into the `carddemo` folder. 4. Compress the `cardemo` folder into a ZIP file. 5. Upload the ZIP file to the Amazon S3 bucket that you created. For instructions, see [Uploading objects](#) in the Amazon S3 documenta tion.	

Configure IAM Identity Center and AWS Transform

Task	Description	Skills required
Add users to IAM Identity Center.	Add your prospective users to IAM Identity Center. Follow the instructions in [Adding users in IAM Identity Center](#) in the AWS Transform documentation.	AWS administrator
Enable AWS Transform and add users.	1. Enable AWS Transform. Follow the instructions in [Enable AWS Transform](#). 2. Add users to AWS Transform. Follow the	AWS administrator

Task	Description	Skills required
	instructions in [Adding users to AWS Transform]().	
Configure user access to the AWS Transform web applicati on.	Each user must accept the invitation to access the AWS Transform web application. Follow the instructions in [Accepting the invitation]() in the AWS Transform documenta tion.	App developer, App owner
Log in to the AWS Transform web application.	Follow the instructions in [Signing in to AWS Transform]().	App developer, App owner
Set up a workspace.	Set up a workspace where users can collaborate in the AWS Transform web applicati on. Follow the instructions in [Setting up your workspace]() in the AWS Transform documentation.	AWS administrator

Transform the mainframe application

Task	Description	Skills required
Create a transformation job.	Create a transformation job to modernize the CardDemo mainframe application. For instructions, see [Create and start a job]() in the AWS Transform documentation. When you are asked to set the objectives in the AWS Transform chat interface,	App developer, App owner

Task	Description	Skills required
	choose **Perform mainframe modernization (IBM z/OS to AWS)** and then choose **Analyze code, Generate technical documentation, Business logic, Decompose code, Plan Migration sequence and Transform code to Java**.	
Set up the connector.	Establish a connector to the Amazon S3 bucket that contains the CardDemo mainframe application. This connector allows AWS Transform to access resources in the bucket and perform consecutive transformation functions. For instructions, see Set up a connector in the AWS Transform documentation.	AWS administrator

Task	Description	Skills required
Perform code analysis.	1. On the **Specify asset location** page, enter the Amazon S3 bucket path for the `carddemo` ZIP file that you uploaded. 2. Choose **Approve and send to AWS Transform**. It starts analyzing the code. 3. Monitor the status on the **Worklog** tab. 4. When the analysis is complete, under **Analyze code** in the left navigatio n pane, choose **View code analysis results**. 5. (Optional) Choose **Download** to download the complete asset list, missing source code, and dependency file. For more information, see Code analysis in the AWS Transform documentation.	App developer, App owner

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Generate technical documentation.</td><td>In the left navigation pane, under Generate technical documentation, choose Select files and configure settings.
Expand COBOL or JCL, and then select one or multiple files.
Choose the documentation detail level:
Summary – Provides a high-level overview of each file in the scope. Also, gives a one-line summary of each file.
Detailed functional specification – Provides comprehensive details for each file in the mainframe application transformation scope. Some details include logic and flow, identified business rules, data flow, dependencies, input and output processing, and various transaction details.
Choose Send to AWS Transform.
Monitor the progress in the Worklog tab.</td><td>App developer, App owner</td></tr>
</table>

Task	Description	Skills required
	Note Document generation time varies from minutes to hours, depending on the number of files and lines of code. 6. When complete, choose **Review documentation results**. 7. Access the generated document through the web application or download it. The generated documenta tion is also stored in the Amazon S3 bucket. 8. Explore the generated documentation by asking specific questions in the chat interface. For example, you can say "Tell me the features of CBACT01C." For more information, see Generate technical documentation in the AWS Transform documentation.	

Task	Description	Skills required
Extract the business logic.	1. In the left navigation pane, expand **Extract business logic**, and then choose **Select files for business logic extraction**. 2. Expand **COBOL** or **JCL**, and then select one or multiple files. 3. Choose **Send to AWS Transform**. 4. Monitor the progress in the **Worklog** tab. **Note** Business logic extraction time varies from minutes to hours, depending on the number of files and lines of code. 5. When extraction is complete, choose **Review business logic extraction results**. 6. Select a COBOL or JCL file, and then choose **View**. 7. Change the scope in **Flow diagram and functional rules** to review different levels of the flow charts,	App developer, App owner

Task	Description	Skills required
	including the supporting code. 8. (Optional) Review the generated business logic, which is stored in JSON format in your Amazon S3 bucket. For more information, see Extract business logic in the AWS Transform documenta tion.	

Task	Description	Skills required
Decompose the code.	1. In the left navigation pane, expand **Decompose code**, and then choose **Decompose into domains**. 2. In the **Actions** list, choose **Create domain**. 3. Enter a name for the new domain, such as `Bill payment`. 4. (Optional) Provide a description. 5. In the **Find files** search bar, search for `CB00`, and then select the file. 6. Choose **Mark as seed**. 7. Verify that the seed flag changes from **No** to **Yes**. 8. Choose **Create**. 9. Choose **Save**. 10. Choose **Decompose**. 11. Review the decomposition output by choosing the domain name. 12. When the decomposition is complete, choose **Send to AWS Transform**. For more information about decomposition and seeds, see Decomposition in the AWS Transform documentation.	App developer, App owner

Task	Description	Skills required
Plan the migration waves.	Plan the migration waves for the CardDemo application. Follow the instructions in [Migration wave planning](#) in the AWS Transform documentation to review and edit the wave plan.	App developer, App owner
Refactor the code.	1. Refactor the CardDemo mainframe application into Java code in all or selected domain files. Follow the instructions in [Refactor code](#) in the AWS Transform documentation. 2. After the refactoring process is complete, build and deploy the modernized Java application. Follow the instructions in [Build and deploy your modernized application post-refactoring](#) in the AWS Transform documentation.	App developer, App owner

Task	Description	Skills required
(Optional) Use Reforge to improve the Java code.	1. Create a new job, and then enter an objective of `Reforge the code`. 2. Enter the of a project that has been modernized through AWS Transform. This project should be in your Amazon S3 bucket. 3. Enter the Java class list that specifies which service classes to reforge. 4. Review the generated output in the Amazon S3 bucket. For more information, see Reforge in the AWS Transform documentation.	App developer, App owner

Task	Description	Skills required
Streamline the deployment.	AWS Transform can provide infrastructure as code (IaC) templates for CloudFormation, AWS CDK, or Terraform. These templates help you deploy core components, including compute, database, storage, and security resources. 1. In the AWS Transform chat interface, request the templates by entering `Share the mainframe refactor <service> templates` , where `<service>` is one of the supported IaC services. 2. Deploy the template by using your preferred service. For example, see the following resources: • Create a stack from the CloudFormation console (CloudFormation documentation) • Deploy AWS CDK applications (AWS CDK documentation) • Build infrastructure (Terraform documentation)	App developer, App owner

Task	Description	Skills required
	For more information, see Deployment capabilities in the AWS Transform documentation.	

Troubleshooting

Issue	Solution
You are unable to view the source code or generated document in the AWS Transform web application.	Add a policy to the CORS permission for the Amazon S3 bucket to allow AWS Transform as an origin. For more information, see S3 bucket CORS permissions in the AWS Transform documentation.

Related resources

AWS documentation

- Transformation of mainframe applications (AWS Transform documentation)

Other AWS resources

- Accelerate Your Mainframe Modernization Journey using AI Agents with AWS Transform (AWS blog post)
- AWS Transform FAQs
- AWS IAM Identity Center FAQs

Videos and tutorials

- Introduction to Amazon Q Developer: Transform (AWS Skill Builder)
- AWS re:Invent 2024 - Modernize mainframe applications faster using Amazon Q Developer (YouTube)

- [AWS re:Invent 2024 - Automating migration and modernization to accelerate transformation](#) (YouTube)
- [AWS re:Invent 2024 - Toyota drives innovation & enhances operational efficiency with gen AI](#) (YouTube)

> **Note**
>
> AWS Transform was previously known as *Amazon Q Developer transform for mainframe*.

Modernize mainframe output management on AWS by using Rocket Enterprise Server and LRS PageCenterX

Shubham Roy, Amazon Web Services

Abraham Rondon, Micro Focus

Guy Tucker, Levi, Ray and Shoup Inc

Summary

By modernizing your mainframe output management, you can achieve cost savings, mitigate the technical debt of maintaining legacy systems, and improve resiliency and agility through DevOps and Amazon Web Services (AWS) cloud-native technologies. This pattern shows you how to modernize your business-critical mainframe output-management workloads on the AWS Cloud. The pattern uses Rocket Enterprise Server as a runtime for a modernized mainframe application, with Levi, Ray & Shoup, Inc. (LRS) VPSX/MFI (Micro Focus Interface) as a print server and LRS PageCenterX as an archive server. LRS PageCenterX provides output-management solutions for viewing, indexing, searching, archiving, and securing access to business outputs.

The pattern is based on the replatform mainframe modernization approach. Mainframe applications are migrated by AWS Mainframe Modernization on Amazon Elastic Compute Cloud (Amazon EC2). Mainframe output-management workloads are migrated to Amazon EC2, and a mainframe database, such as IBM Db2 for z/OS, is migrated to Amazon Relational Database Service (Amazon RDS). The LRS Directory Integration Server (LRS/DIS) works with AWS Directory Service for Microsoft Active Directory for output-management workflow authentication and authorization.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A mainframe output-management workload.
- Basic knowledge of how to rebuild and deliver a mainframe application that runs on Rocket Enterprise Server. For more information, see the Rocket Enterprise Server data sheet in the Rocket Software documentation.
- Basic knowledge of LRS cloud printing solutions and concepts. For more information, see *Output Modernization* in the LRS documentation.

- Rocket Enterprise Server software and license. For more information, contact [Rocket Software](#).
- LRS VPSX/MFI, LRS PageCenterX, LRS/Queue, and LRS/DIS software and licenses. For more information, [contact LRS](#). You must provide the hostnames of the EC2 instances where the LRS products will be installed.

Note: For more information about configuration considerations for mainframe output-ma nagement workloads, see *Considerations* in the [Additional information](#) section of this pattern.

Product versions

- [Rocket Enterprise Server 10.0](#)
- [LRS VPSX/MFI](#)
- [LRS PageCenterX](#) V1R3 or later

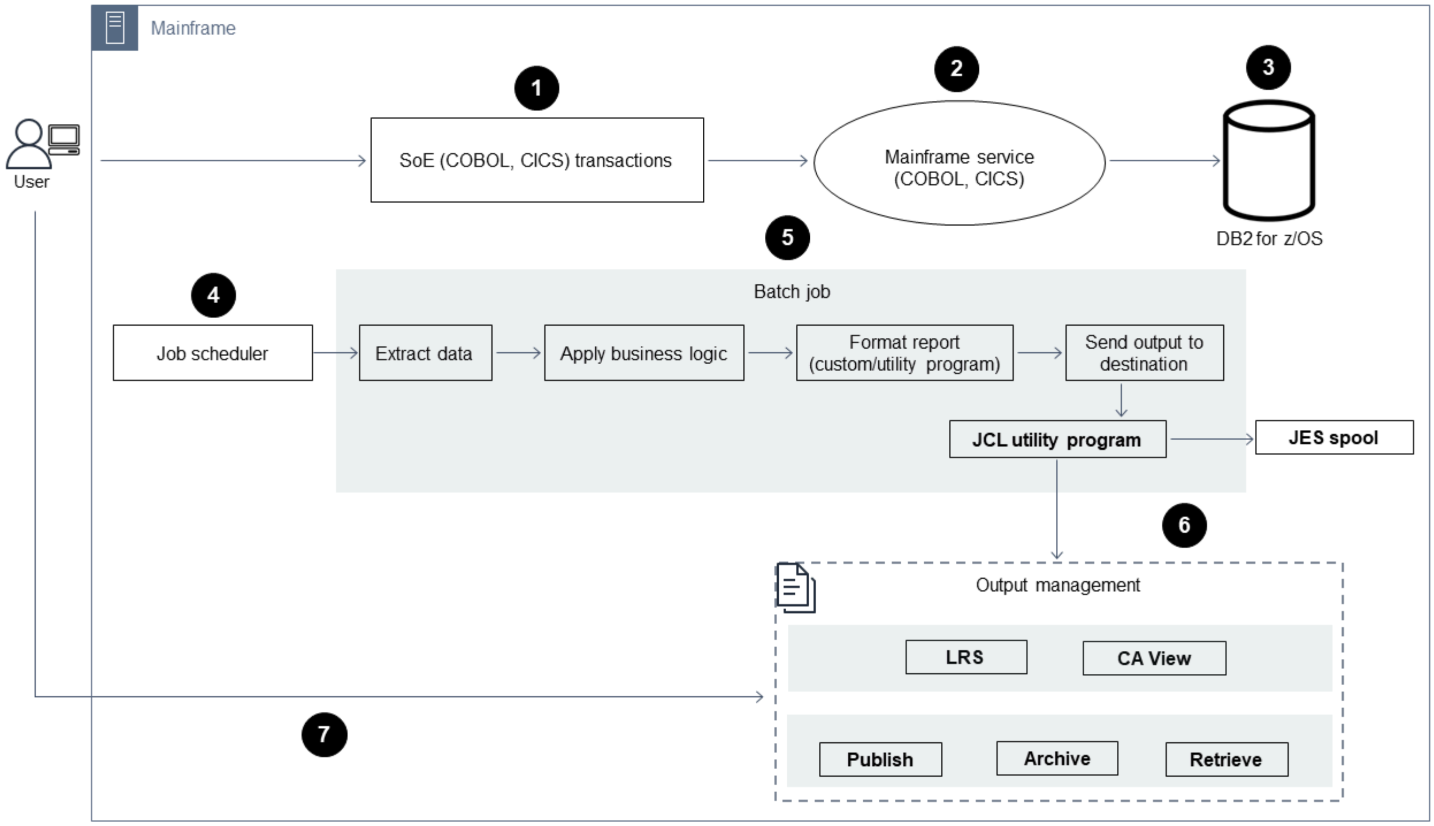
Architecture

Source technology stack

- Operating system – IBM z/OS
- Programming language – Common business-oriented language (COBOL), job control language (JCL), and Customer Information Control System (CICS)
- Database – IBM Db2 for z/OS, IBM Information Management System (IMS) database, and Virtual Storage Access Method (VSAM)
- Security – Resource Access Control Facility (RACF), CA Top Secret for z/OS, and Access Control Facility 2 (ACF2)
- Print and archive solutions – IBM mainframe z/OS output and printing products (IBM Infoprint Server for z/OS, LRS, and CA Deliver) and archiving solutions (CA Deliver, ASG Mobius, or CA Bundle)

Source architecture

The following diagram shows a typical current state architecture for a mainframe output-management workload.



The diagram shows the following workflow:

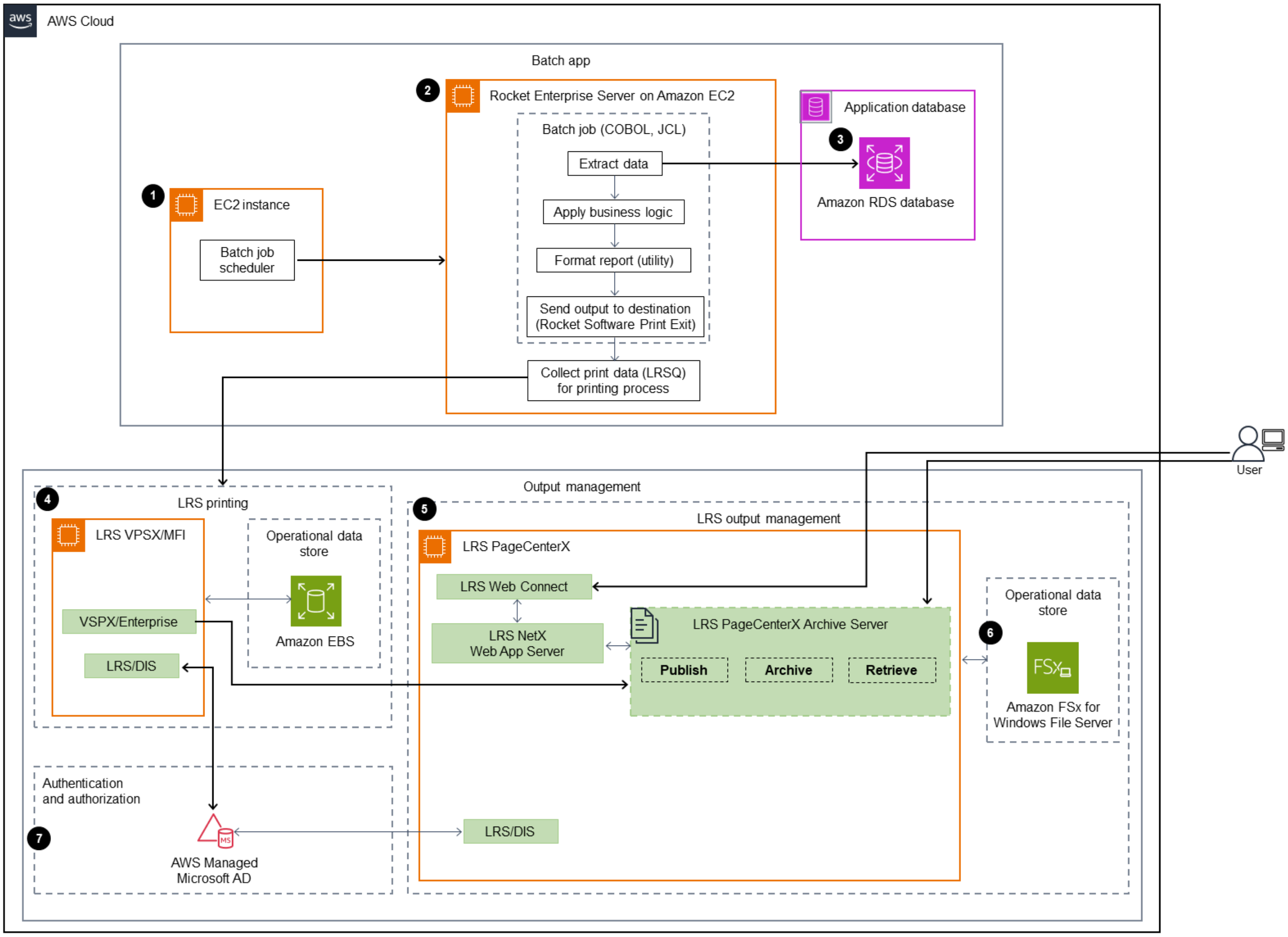
1. Users perform business transactions on a system of engagement (SoE) that's built on an IBM CICS application written in COBOL.
2. The SoE invokes the mainframe service, which records the business transaction data in a system-of-records (SoR) database such as IBM Db2 for z/OS.
3. The SoR persists the business data from the SoE.
4. The batch job scheduler initiates a batch job to generate print output.
5. The batch job extracts data from the database. It formats the data according to business requirements, and then it generates business output such as billing statements, ID cards, or loan statements. Finally, the batch job routes the output to output management for format, publish, and storage of the output based on the business requirements.
6. Output management receives output from the batch job. Output management indexes, arranges, and publishes the output to a specified destination in the output-management system, such as LRS PageCenterX solutions (as demonstrated in this pattern) or CA View.
7. Users can view, search, and retrieve the output.

Target technology stack

- Operating system – Windows Server running on Amazon EC2
- Compute – Amazon EC2
- Storage – Amazon Elastic Block Store (Amazon EBS) and Amazon FSx for Windows File Server
- Programming language – COBOL, JCL, and CICS
- Database – Amazon RDS
- Security – AWS Managed Microsoft AD
- Printing and archiving – LRS printing (VPSX) and archiving (PageCenterX) solution on AWS
- Mainframe runtime environment – Rocket Enterprise Server

Target architecture

The following diagram shows an architecture for a mainframe output-management workload that's deployed in the AWS Cloud.



The diagram shows the following workflow:

1. The batch job scheduler initiates a batch job to create output, such as billing statements, ID cards, or loan statements.
2. The mainframe batch job (replatformed to Amazon EC2) uses the Rocket Enterprise Server runtime to extract data from the application database, apply business logic to the data, and format the data. It then sends the data to an output destination by using the Rocket Software printer exit module (OpenText Micro Focus documentation).
3. The application database (an SoR that runs on Amazon RDS) persists data for print output.
4. The LRS VPSX/MFI printing solution is deployed on Amazon EC2, and its operational data is stored in Amazon EBS. LRS VPSX/MFI uses the TCP/IP-based LRS/Queue transmission agent to collect output data through the Rocket Software JES Print Exit API.

LRS VPSX/MFI does data preprocessing, such as EBCDIC to ASCII translation. It also does more complex tasks, including converting mainframe-exclusive data streams such as IBM Advanced Function Presentation (AFP) and Xerox Line Conditioned Data Stream (LCDS) into more common viewing and printing data streams such as Printer Command Language (PCL) and PDF.

During the maintenance window of LRS PageCenterX, LRS VPSX/MFI persists the output queue and serves as backup for the output queue. LRS VPSX/MFI connects and sends output to LRS PageCenterX by using the LRS/Queue protocol. LRS/Queue performs an exchange of both readiness and completion for the jobs to help ensure that the data transfer occurs.

Notes:

For more information on print data passed from Rocket Software Print Exit to LRS/Queue and LRS VPSX/MFI supported mainframe batch mechanisms, see *Print data capture* in the Additional information section.

LRS VPSX/MFI can perform health checks at the printer-fleet level. For more information, see *Printer-fleet health checks* in the Additional information section of this pattern.

5. The LRS PageCenterX output-management solution is deployed on Amazon EC2, and its operational data is stored in Amazon FSx for Windows File Server. LRS PageCenterX provides a central report management system of all files imported into LRS PageCenterX along with all users able to access the files. Users can view specific file content or perform searches across multiple files for matching criteria.

   The LRS/NetX component is a multi-threaded web application server that provides a common runtime environment for the LRS PageCenterX application and other LRS applications. The LRS/Web Connect component is installed on your web server and provides a connector from the web server to the LRS/NetX web application server.

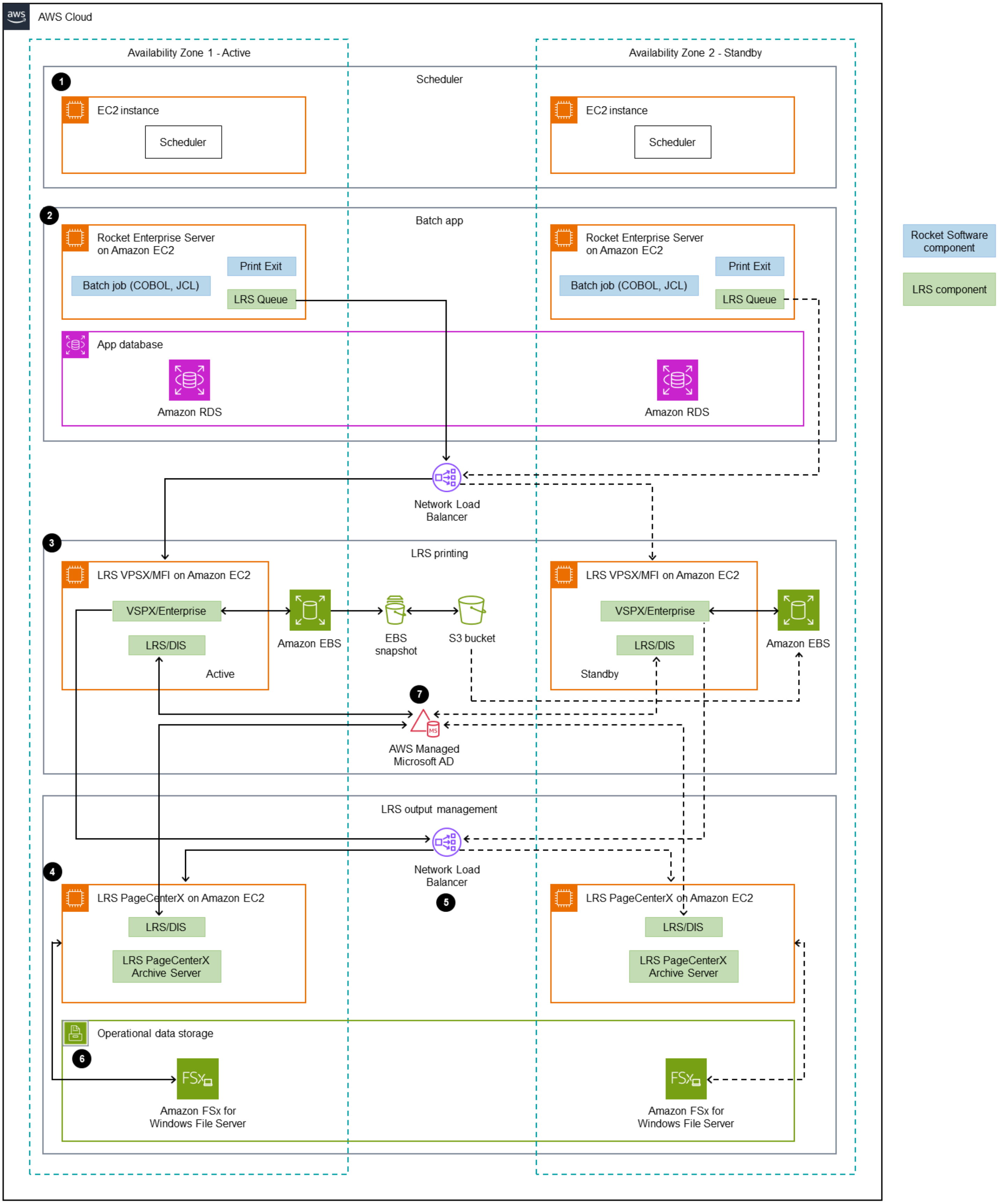
6. LRS PageCenterX provides storage for file system objects. The operational data of LRS PageCenterX is stored in Amazon FSx for Windows File Server.

7. Output-management authentication and authorization are performed by AWS Managed Microsoft AD with LRS/DIS.

> **Note**
>
> The target solution typically doesn't require application changes to accommodate mainframe formatting languages, such as IBM AFP or Xerox LCDS.

AWS infrastructure architecture

The following diagram shows a highly available and secure AWS infrastructure architecture for a mainframe output-management workload.


AWS Cloud
Availability Zone 1 - Active
Availability Zone 2 - Standby
1
Scheduler
EC2 instance
Scheduler
EC2 instance
Scheduler
2
Batch app
Rocket Enterprise Server on Amazon EC2
Print Exit
Batch job (COBOL, JCL)
LRS Queue
Rocket Enterprise Server on Amazon EC2
Print Exit
Batch job (COBOL, JCL)
LRS Queue
App database
Amazon RDS
Amazon RDS
Network Load Balancer
3
LRS printing
LRS VPSX/MFI on Amazon EC2
VSPX/Enterprise
LRS/DIS
Active
Amazon EBS
EBS snapshot
S3 bucket
LRS VPSX/MFI on Amazon EC2
VSPX/Enterprise
LRS/DIS
Standby
Amazon EBS
7
AWS Managed Microsoft AD
LRS output management
Network Load Balancer
5
4
LRS PageCenterX on Amazon EC2
LRS/DIS
LRS PageCenterX Archive Server
LRS PageCenterX on Amazon EC2
LRS/DIS
LRS PageCenterX Archive Server
Operational data storage
6
Amazon FSx for Windows File Server
Amazon FSx for Windows File Server
Rocket Software component
LRS component

The diagram shows the following workflow:

1. The batch scheduler initiates the batch process and is deployed on Amazon EC2 across multiple Availability Zones for high availability (HA).

> **Note**
>
> This pattern doesn't cover the implementation of the batch scheduler. For more information about implementation, see the software vendor documentation for your scheduler.

2. The mainframe batch job (written in a programming language such as JCL or COBOL) uses core business logic to process and generate print output, such as billing statements, ID cards, and loan statements. The batch job is deployed on Amazon EC2 across two Availability Zones for HA. It uses the Rocket Software Print Exit API to route print output to LRS VPSX/MFI for data preprocessing.
3. The LRS VPSX/MFI print server is deployed on Amazon EC2 across two Availability Zones for HA (active-standby redundant pair). It uses Amazon EBS as an operational data store. The Network Load Balancer performs a health check on the LRS VPSX/MFI EC2 instances. If an active instance is in an unhealthy state, the load balancer routes traffic to hot standby instances in the other Availability Zone. The print requests are persisted in the LRS Job Queue locally in each of the EC2 instances. In the event of a failure, a failed instance must be restarted before the LRS services can resume processing the print request.

> **Note**
>
> LRS VPSX/MFI can also perform health checks at the printer-fleet level. For more information, see *Printer-fleet health checks* in the Additional information section of this pattern.

4. LRS PageCenterX output management is deployed on Amazon EC2 across two Availability Zones for HA (active-standby redundant pair). It uses Amazon FSx for Windows File Server as an operational data store. If an active instance is in an unhealthy state, the load balancer performs a health check on the LRS PageCenterX EC2 instances and routes traffic to standby instances in the other Availability Zone.
5. A Network Load Balancer provides a DNS name to integrate the LRS VPSX/MFI Server with LRS PageCenterX.

> **Note**
>
> LRS PageCenterX supports a Layer 4 load balancer.

6. LRS PageCenterX uses Amazon FSx for Windows File Server as an operational data store deployed across two Availability Zones for HA. LRS PageCenterX understands only files that are in the file share, not in an external database.
7. AWS Managed Microsoft AD is used with LRS/DIS to perform output-management workflow authentication and authorization. For more information, see *Print output authentication and authorization* in the Additional information section.

Tools

AWS services

- AWS Directory Service for Microsoft Active Directory enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud.
- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Elastic Load Balancing (ELB) distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon EC2 instances, containers, and IP addresses in one or more Availability Zones. This pattern uses a Network Load Balancer.
- Amazon FSx provides file systems that support industry-standard connectivity protocols and offer high availability and replication across AWS Regions. This pattern uses Amazon FSx for Windows File Server.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.

Other tools

- LRS PageCenterX software provides a scalable document and report content management solution that helps users obtain maximum value from information through automated indexing, encryption, and advanced search features.

- [LRS VPSX/MFI (Micro Focus Interface)](), codeveloped by LRS and Rocket Software, captures output from a Rocket Software JES spool and reliably delivers it to a specified print destination.
- LRS/Queue is a transmission agent that's TCP/IP based. LRS VPSX/MFI uses LRS/Queue to collect or capture print data through the Rocket Software JES Print Exit programming interface.
- LRS Directory Integration Server (LRS/DIS) is used for authentication and authorization during the print workflow.
- [Rocket Enterprise Server]() is an application deployment environment for mainframe applications. It provides the runtime environment for mainframe applications that are migrated or created by using any version of Rocket Enterprise Developer.

Epics

Set up the Rocket runtime and deploy a mainframe batch application

Task	Description	Skills required
Set up the runtime and deploy a demo application.	To set up Rocket Enterprise Server on Amazon EC2 and deploy the Rocket Software BankDemo demonstra tion application, follow the instructions in AWS Mainframe Modernization [user-guide](). The BankDemo application is a mainframe batch applicati on that creates and then initiates print output.	Cloud architect

Set up an LRS print server on Amazon EC2

Task	Description	Skills required
Create an Amazon EC2 Windows instance.	To launch an Amazon EC2 Windows instance, follow	Cloud architect

Task	Description	Skills required
	the instructions in [Launch an Amazon EC2 instance](#) in the Amazon EC2 documentation. Use the same hostname that you used for your LRS product license. Your instance must meet the following hardware and software requirements for LRS VPSX/MFI: • CPU – Dual Core • RAM – 16 GB • Drive – 500 GB • Minimum EC2 instance – `m5.xlarge` • OS – Windows • Software – Internet Information Services (IIS) or Apache **Note** The preceding hardware and software requireme nts are intended for a small printer fleet (around 500-1000) . To get the full requirements, consult	

Task	Description	Skills required
	with your LRS and AWS contacts. 1. When you create your Windows instance, confirm that the EC2 hostname is the same hostname used for the LRS product license. 2. Connect to your EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 3. On the Windows **Start** menu, find and open **Server Manager**. 4. In Server Manager, choose **Dashboard**, **Quick Start**, **Add roles and features**, and then choose **Server roles**. 5. In Server roles, choose **WebServer (IIS)**, and then choose **Application Development**. 6. In Application Developme nt, select the **CGI** check box. 7. To install CGI, follow the instructions in the Windows Server Manger	

Task	Description	Skills required
	Add roles and features wizard. 8. Open port 5500 in the Windows firewall of the EC2 instance for LRS/ Queue communication.	

Task	Description	Skills required
Install LRS VPSX/MFI on the EC2 instance.	1. Connect to your EC2 instance. 2. Open the link to the product download page from the LRS email message that you should have received. **Note** LRS products are distributed by electronic file transfer (EFT). 3. Download LRS VPSX/MFI, and unzip the file (default folder: `c:\LRS`). 4. To install LRS VPSX/MFI, launch the LRS Product Installer from the unzipped folder. 5. On the **Select Features** menu, select **VPSX® Server**, and then choose **Next** to start the installat ion process. You will receive a success message when installation is complete.	Cloud architect

Task	Description	Skills required
Install LRS/Queue.	1. Connect to your Rocket Enterprise Server EC2 instance. 2. Open the link to the LRS product download page from the LRS email message that you should have received, download LRS/Queue, and then unzip the file. 3. Navigate to the location where you downloaded the files, and then launch the LRS Product Installer to install LRS/Queue. 4. Follow the instructions in the LRS Product Installer to complete the installation process.	Cloud architect

Task	Description	Skills required
Install LRS/DIS.	The LRS/DIS product often is included in LRS VPSX installation. However, if LRS/DIS wasn't installed along with LRS VPSX, use the following steps to install it: 1. Connect to your LRS VPSX/MFI EC2 instance. 2. Open the link to the LRS product download page from the LRS email message that you should have received, download LRS/DIS, and then unzip the file. 3. Navigate to the location where you downloaded the files, and then launch the LRS Product Installer. 4. In the LRS Product Installer, expand **LRS Misc Tools**, select **LRS DIS**, and then choose **Next**. 5. Follow the rest of the instructions in the LRS Product Installer to complete the installation process.	Cloud architect

Task	Description	Skills required
Create a target group.	Create a target group by following the instructions in [Create a target group for your Network Load Balancer](). When you create the target group, register the LRS VPSX/MFI EC2 instance as the target: 1. On the **Specify group details** page, for **Choose a Target Type**, choose **Instances**. 2. For **Protocol**, choose **TCP**. 3. For **Port**, choose **5500**. 4. On the **Register targets** page, in the **Available instances** section, select the LRS VPSX/MFI EC2 instance.	Cloud architect

Task	Description	Skills required
Create a Network Load Balancer.	To create the Network Load Balancer, follow the instructions in the [Elastic Load Balancing documentation](). Your Network Load Balancer routes traffic from Rocket Enterprise Server to the LRS VPSX/MFI EC2 instance. When you create the Network Load Balancer, choose the following values on the **Listeners and Routing** page: 1. For **Protocol**, choose **TCP**. 2. For **Port**, choose **5500**. 3. For **Default action**, choose **Forward to** for the target group that you created earlier.	Cloud architect

Integrate Rocket Enterprise Server with LRS/Queue and LRS VPSX/MFI

Task	Description	Skills required
Configure Rocket Enterprise Server for LRS/Queue integration.	1. Connect to your Rocket Enterprise Server EC2 instance by following the instructions in the [Amazon EC2 documentation](). 2. On the Windows **Start** menu, open the Rocket Enterprise Server Administration UI.	Cloud architect

Task	Description	Skills required
	3. In the menu bar, choose **NATIVE**. 4. In the navigation pane, choose **Directory Server**, and then choose **BANKDEMO** for your Enterprise Server region. 5. From **General** in the left navigation pane, scroll down to the **Additional** section to configure the environment variables (`LRSQ_ADDRESS` , `LRSQ_PORT` , `LRSQ_COMMAND`) to point to LRSQ. • For **LRSQ_ADDRESS**, enter the IP address or DNS name of the Network Load Balancer that you created earlier. • For **LRSQ_PORT**, enter **VPSX LRSQ Listener Port (5500)**. • For **LRSQ_COMMAND**, enter the path location of the LRSQ executable. **Note** LRS currently supports a maximum character limit of 50 for DNS names.	

Task	Description	Skills required
	If your DNS name is longer than 50 characters, you can use the IP address of the Network Load Balancer as an alternative.	

Task	Description	Skills required
Configure Rocket Enterprise Server for LRS VPSX/MFI integration.	1. Copy the `VPSX_MFI_R2` folder from the LRS VPSX/MFI installer to the Rocket Enterprise Server location at `C\BANKDEMO\print`. 2. Connect to your Rocket Enterprise Server EC2 instance by following the instructions in the [Amazon EC2 documentation](https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/connecting_to_windows_instance.html). 3. On the Windows **Start** menu, open the Rocket Enterprise Server Administration UI. 4. On the menu bar, choose **NATIVE**. 5. In the navigation pane, choose **Directory Server**, and then choose **BANKDEMO**. 6. Under BANKDEMO, choose **JES**. 7. Under **JES Program Path**, add the `DLL(VPSX_MFI_R2)` path from `C\BANKDEMO\print`.	Cloud architect

Set up the print queue and the print users

Task	Description	Skills required
Associate the Rocket Software Print Exit module	1. Connect to your Rocket Enterprise Server EC2	Cloud architect

Task	Description	Skills required
with the Rocket Enterprise Server batch printer Server Execution Process.	instance by following the instructions in the [Amazon EC2 documentation](). 2. On the Windows **Start** menu, open the Rocket Focus Enterprise Server Administration UI. 3. On the menu bar, choose **NATIVE**. 4. In the navigation pane, choose **Directory Server**, and then choose **BANKDEMO**. 5. Under BANKDEMO, choose **JES**, and scroll down to **Printers**. 6. In Printers, associate the Rocket Software Print Exit module (**LRSPRTE6 for Batch**) to the Rocket Enterprise Server batch printer **Server Execution Process (SEP)**. This enables print output routing to LRS VPSX/MFI. For more information about configuration, see [Using the Exit]() in the OpenText Micro Focus documentation.	

Task	Description	Skills required
Create a print output queue in LRS VPSX/MFI and integrate it with LRS PageCenterX.	1. Connect to your LRS VPSX/ MFI EC2 instance. 2. On the Windows **Start** menu, open **VPSX Web Interface**. 3. In the navigation pane, choose **Printers**. 4. Choose **Add**, and then choose **Add Printer**. 5. On the **Printer Configuration** page, for **Printer Name**, enter **Local**. 6. For **VPSX ID**, enter **VPS1**. 7. For **CommType**, select **TCPIP/LRSQ**. 8. For **Host/IP Address**, enter the IP address of the Network Load Balancer fronting the LRS PageCenterX EC2 instances. 9. For **Remote port**, enter 5800. 10. For **Remote queue,** enter the name of the LRS PageCenterX document folder where output will be stored. 11. Choose **Add**.	Cloud architect

Task	Description	Skills required
Create a print user in LRS VPSX/MFI.	1. Connect to your LRS VPSX/ MFI EC2 instance. 2. On the Windows **Start** menu, open **VPSX Web Interface**. 3. In the navigation pane, choose **Security**, and then choose **Users**. 4. In the **User Name** column, choose **admin**, and then choose **Copy**. 5. In the **User Profile Maintenance** window, for **User Name**, enter a username (for example, **PrintUser**). 6. For **Description**, enter a brief description (for example, **User for test print**). 7. Choose **Update**. This creates a print user (for example, **PrintUser**). 8. In the navigation pane, under **User**, choose the new user that you created. 9. On the **Command** menu, choose **Security**. 10. On the **Security Rules** page, choose all the applicable printer security and job security options, and then choose **Save**.	Cloud architect

Task	Description	Skills required
	11. To add your new print user to the **Administrator** group, on the navigation pane, choose **Security**, and then choose **Configure**. 12. In the **Security configuration** window, add your new print user to the **Administrator** column.	

Set up an LRS PageCenterX server on Amazon EC2

Task	Description	Skills required
Create an Amazon EC2 Windows instance.	Launch an Amazon EC2 Windows instance by following the instructions from [Step 1: Launch an instance](#) in the Amazon EC2 documentation. Use the same hostname that you used for your LRS product license. Your instance must meet the following hardware and software requirements for LRS PageCenterX: • CPU – Dual Core • RAM – 16 GB • Drive – 500 GB • Minimum EC2 instance – `m5.xlarge` • OS – Windows	Cloud architect

Task	Description	Skills required
	• Software –IIS or Apache **Note** The preceding hardware and software requirements are intended for a small printer fleet (around 500–1000). To get the full requirements, consult with your LRS and AWS contacts. 1. When you create your Windows instance, confirm that the EC2 hostname is the same hostname used for the LRS product license. 2. Connect to your EC2 instance by following the instructions in the [Amazon EC2 documentation](#). 3. On the Windows **Start** menu, find and open **Server Manager**. 4. In Server Manager, choose **Dashboard**, **Quick Start**, **Add roles and features**, then choose **Server roles**. 5. In Server roles, choose **WebServer (IIS)**, and	

Task	Description	Skills required
	then choose **Application Development**. 6. In Application Developme nt, select the **CGI** check box. 7. To install CGI, follow the instructions in the Windows Server Manger **Add roles and features** wizard. 8. Open port 5800 for inbound TCP/IP traffic in the Windows firewall of the EC2 instance. LRS VPSX uses the TCPIP/LRS Q protocol on 5800 port to communicate to LRS PageCenterX.	

Task	Description	Skills required
Install LRS PageCenterX on the EC2 instance.	1. Connect to your EC2 instance. 2. Open the link to the product download page from the LRS email message that you should have received. **Note** LRS products are distributed by electronic file transfer (EFT). 3. Download LRS PageCenterX, and unzip the file (default folder: `c:\LRS`). 4. To install LRS PageCenterX, launch the LRS Product Installer from the unzipped folder. 5. On the **Select Features** menu, select **PageCenterX**, and then choose **Next** to start the installation process. You will receive a success message when installation is complete.	Cloud architect

Task	Description	Skills required
Install LRS/DIS.	The LRS/DIS product often is included in LRS VPSX installat ion. However, if LRS/DIS wasn't installed along with LRS VPSX, use the following steps to install it: 1. Connect to your LRS PageCenterX EC2 instance. 2. Open the link to the LRS product download page from the LRS email that you should have received, download LRS/DIS, and then unzip the file. 3. Navigate to the location where you downloaded the files, and then launch the LRS Product Installer. 4. In the LRS Product Installer , expand **LRS Misc Tools**, select **LRS DIS**, and then choose **Next**. 5. Follow the rest of the instructions in the LRS Product Installer to complete the installation process.	Cloud architect

Task	Description	Skills required
Create a target group.	Create a target group by following the instructions in [Create a target group for your Network Load Balancer](). When you create the target group, register the LRS PageCenterX EC2 instance as the target: 1. On the **Specify group details** page, for **Choose a Target Type**, choose **Instances**. 2. For **Protocol**, choose **TCP**. 3. For **Port**, choose **5800**. 4. On the **Register targets** page, in the **Available instances** section, select the LRS PageCenterX EC2 instance.	Cloud architect

Task	Description	Skills required
Create a Network Load Balancer.	To create the Network Load Balancer, follow the instructions in the [Elastic Load Balancing documentation](). Your Network Load Balancer routes traffic from LRS VPSX/MFI to the LRS PageCenterX EC2 instance. When you create the Network Load Balancer, choose the following values on the **Listeners and Routing** page: 1. For **Protocol**, choose **TCP**. 2. For **Port**, choose **5800**. 3. For **Default action**, choose **Forward to** for the target group that you created earlier.	Cloud architect

Set up output-management features in LRS PageCenterX

Task	Description	Skills required
Enable the Import function in LRS PageCenterX.	You can use the LRS PageCenterX Import function to recognize the outputs landing on LRS PageCenterX by criteria such as Job name or Form ID. You can then route the outputs to specific folders in LRS PageCenterX.	Cloud architect

Task	Description	Skills required
	To enable the Import function, do the following: 1. Connect to your LRS PageCenterX EC2 instance by following the instructions in the [Amazon EC2 documentation](https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/connecting_to_windows_instance.html). 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**. 4. On the **Configuration** page, choose **Advanced**, **Import parameter**. 5. In the **Import parameter section**, select the **Advanced Import** check box. 6. To commit the changes, choose **Update**.	

Task	Description	Skills required
Configure the document retention policy.	LRS PageCenterX uses a document retention policy to decide how long to keep a document in LRS PageCente rX. To configure the document retention policy, do the following: 1. Connect to your LRS PageCenterX EC2 instance. 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**. 4. On the Admin page, choose **Archive Group List/General admin**, and then choose **Retention policy**. 5. In the Retention policy section, choose **Add** to create a retention policy. 6. On the **Retention Policy Information** page, enter a **Retention policy name**, **Description**, and **Document retention** period. 7. To save your changes and create the policy, choose **Ok**.	Cloud architect

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a rule to route the output document to a specific folder in LRS PageCenterX.</td><td>In LRS PageCenterX, Destination determines the folder path where output will be sent when this destination is invoked by Report Definition. For this example, create a folder based on the Form ID folder in the report definition, and save the output to that folder.

1. Connect to your LRS PageCenterX EC2 instance.
2. On the Windows Start menu, open PCX Web Interface.
3. In Folder Explorer, choose Admin, Advance Import, Destination.
4. In the Destination section, choose Add to open the Destination Maintenance form.
5. On the Destination Maintenance form, enter the following values:
• Destination name – Form
• Description – Description of the destination, such as Form-based folder structure</td><td>Cloud architect</td></tr>
</table>

Task	Description	Skills required
	Destination type – Folder **Folder Parameters** – Import folder path (the folder path that will be created in PageCenterX when the document arrives; for example, the path `/Test/&FORM/&IMPORTDATE/&IMPORTTIME` will create a base `Test` folder, a subfolder based on **Form-Id** named `STD`, a subfolder based on **Import date**, and then a subfolder based on **Import time**) **Document name** – Dynamic name assigned to a document when it's stored in the folder. 6. In the dropdown list, choose a retention policy. For example, choose **Year1** to retain the document for 1 year. 7. To save the changes, choose **Ok**.	

Task	Description	Skills required
Create a report definition.	1. Connect to your LRS PageCenterX EC2 instance. 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**, **Advance Import**, **Report Definition**, and then choose **Add**. 4. On the **Report definitio n maintenance** page, on the **General** tab, enter the **Report Definition name.** 5. On the **General** tab, under **Fields**, you can specify selection criteria such as **Job Name**, **Form**, **Class**, and **Author**. For example, you could enter a **Job Name** of **MFIDEMO**. The **Job Name** value will be the name of the batch job that will generate print output. 6. On the **Destination** tab, under **Available Destinati on**, choose the previously created destination (**Form**). 7. Choose **Add** to add the **Form** destination as **Assigned destination**.	Cloud architect

Task	Description	Skills required
	Note This example includes a report definition where an output generated by the MFIDEMO and routed to LRS PageCenterX is saved in the folder structure defined in the destination definition.	

Set up authentication and authorization for output management

Task	Description	Skills required
Create an AWS Managed Microsoft AD domain with users and groups.	1. To create a directory on AWS Managed Microsoft AD, follow the instructions in [Create your AWS Managed Microsoft AD directory](). 2. To deploy an EC2 instance (Active Directory manager) and install Active Directory tools to manage your AWS Managed Microsoft AD, follow the instructions in [Step 3: Deploy an EC2 instance to manage your]()	Cloud architect

Task	Description	Skills required
	AWS Managed Microsoft AD. 3. To connect to your EC2 instance, follow the instructions in the Amazon EC2 documentation. **Note** When you connect to the EC2 instance, in the **Windows Security** window, enter the administr ator credentials for the directory that you created in step 1. 4. On the Windows **Start** menu, under **Windows Administrative Tools**, choose **Active Directory Users and Computers**. 5. To create a print user in the Active Directory domain, follow the instructions in Create a user.	

Task	Description	Skills required
Join the EC2 instances to an AWS Managed Microsoft AD domain.	Join the LRS VPSX/MFI and LRS PageCenterX EC2 instances to your AWS Managed Microsoft AD domain automatically (AWS Knowledge Center documentation) or manually (AWS Directory Service documentation).	Cloud architect

Task	Description	Skills required
Configure and integrate LRS/DIS with AWS Managed Microsoft AD for the LRS PageCenterX EC2 instance.	1. Connect to your LRS PageCenterX EC2 instance. 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**. 4. On the **Configuration** page, in the **Security Parameters** section, for **Security Type**, select **LRS/DIS**. 5. Enter your preferences for the rest of the options in the Security Parameters section. 6. On the Windows **Start** menu, open the **PageCenterX** folder, choose **Server Start**, and then choose **Server Stop**. 7. Log in to LRS PageCenterX with your Active Directory username and password.	Cloud architect

Task	Description	Skills required
Configure an Import group to import output from LRS VPSX to LRS PageCenterX.	1. Connect to your LRS PageCenterX EC2 instance. 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**, **Security admin**, **Groups**. 4. In the Groups section, choose **Add** to open the **Group preference** form. 5. In the Group preferenc e form, enter values for **Group name** and **Descripti on**. 6. Expand **General options,** and then select the **Import** check box. 7. To save the changes, choose **Ok**.	Cloud architect
Add a security rule to the Import group.	1. Open the context (right-click) menu for **Import group**. 2. Choose **Advance**, and then choose **Security**. 3. In the Security section, choose **Import**, and select the **Subfolder** check box. 4. To save the changes, choose **Apply**.	Cloud architect

Task	Description	Skills required
Create a user in LRS PageCenterX to perform output import from LRS VPSX/MFI.	When you create a user in LRS PageCenterX to perform output import, the username should be the same as the **VPSX ID** of the print output queue in LRS VPSX/MFI. In this example, the VPSX ID is **VPS1**. 1. Connect to your LRS PageCenterX EC2 instance. 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**, **Security admin**, **User.** 4. Choose **Add** to open the **User profile maintenance** form. 5. In User profile maintenan ce, for **User name**, enter **VPS1**.	Cloud architect

Task	Description	Skills required
Add the LRS PageCenterX Import user to the Import only group.	To provide necessary permission for document import from LRS VPSX to LRS PageCenterX, do the following: 1. Connect to your LRS PageCenterX EC2 instance. 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**, **Security admin**, **Groups**. 4. In the Groups section, open the context (right-click) menu for the **Import only** group, and then choose **Advance**, **Security**. 5. On the **Folder Security Records (ImportOnly)** page, choose the **User** tab. 6. On the User tab, under **Name**, select user **VPS1** from the dropdown list, and choose **Apply**.	Cloud architect

Task	Description	Skills required
Configure LRS/DIS with AWS Managed Microsoft AD for the LRS VPSX/MFI EC2 instance.	1. Connect to your LRS VPSX/MFI EC2 instance. 2. On the Windows **Start** menu, open **VPSX Web Interface**. 3. In the navigation pane, choose **Security**, and then choose **Configure**. 4. On the **Security Configura tion** page, in the **Security Parameters** section, for **Security Type**, select **LRS/DIS (External)**. 5. Enter your preferences for the rest of the options in the Security Parameters section. 6. On the Windows **Start** menu, open the **LRS Output Management** folder, choose **Server Start**, and then choose **Server Stop**. 7. Log in to LRS VPSX/MFI with your Active Directory username and password.	Cloud architect

Configure Amazon FSx for Windows File Server as the operational data store for LRS PageCenterX

Task	Description	Skills required
Create a file system for LRS PageCenterX.	To use Amazon FSx for Windows File Server as an operational data store for LRS PageCenterX in a Multi-AZ environment, follow the instructions in [Step 1: Create your file system]().	Cloud architect
Map the file share to the LRS PageCenterX EC2 instance.	To map the file share created in previous step to the LRS PageCenterX EC2 instance, follow the instructions in [Step 2: Map your file share to an EC2 instance running Windows Server]().	Cloud architect
Map LRS PageCenterX Control Directory and Master Folder Directory to the Amazon FSx network share drive.	1. Connect to your LRS PageCenterX EC2 instance by following the instructi ons in the [Amazon EC2 documentation](). 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In **Folder Explorer**, choose **Admin**, **Configuration**. 4. On the Configuration page, choose **Directories**, and then choose **Control Directory**.	Cloud architect

Task	Description	Skills required
	5. In **Control Directories**, enter `\\FSx file share DNS name\share\cntl` . 6. In **Master Folder Directory**, enter `\\FSx file share DNS name\share\mstr` .	

Test an output-management workflow

Task	Description	Skills required
Initiate a batch print request from the Rocket Software BankDemo app.	1. Open the 3270-terminal emulator in your Rocket Enterprise Server EC2 instance. 2. Connect to the BankDemo app by running the command `connect 127.0.0.1:9278` . 3. On the BankDemo command line interface, for **User id**, enter **B0001**. For **Password**, enter a non-blank key. 4. For the **Request printed statement(s)** option, enter **X** on the blank line. 5. In the **Send statement by** section, for **Mail**, enter **Y**, and then press F10.	Test engineer
Check the print output in LRS PageCenterX.	1. Connect to your LRS PageCenterX EC2 instance	Test engineer

Task	Description	Skills required
	by following the instructions in the [Amazon EC2 documentation](). 2. On the Windows **Start** menu, open **PCX Web Interface**. 3. In the navigation pane, open the **Test** folder, open the **STD** folder, and then open the folder with the job run date, such as `08-03-2023` (MM-DD-YYYY). **Note** This is the same folder structure as defined in the story *Create a rule to route the output document to a specific folder in LRS PageCenterX*. 4. Open the `formtest-STD.txt` file. You can now see the print output of an account statement with columns for **Account No.**, **Description**, **Date**, **Amount**, and **Balance**. For an example, see the `batch_pri`	

Task	Description	Skills required
	nt_output attachment for this pattern.	

Related resources

- LRS
- Advanced Function Presentation data stream (IBM documentation)
- Line Conditioned Data Stream (LCDS) (Compart documentation)
- Empowering Enterprise Mainframe Workloads on AWS with Micro Focus (blog post)
- Modernize your mainframe online printing workloads on AWS (AWS Prescriptive Guidance)
- Modernize your mainframe batch printing workloads on AWS (AWS Prescriptive Guidance)

Additional information

Considerations

During your modernization journey, you might consider a wide variety of configurations for mainframe batch and online processes and the output they generate. The mainframe platform has been customized by every customer and vendor that uses it with particular requirements that directly affect print. For example, your current platform might incorporate the IBM AFP data stream or Xerox LCDS into the current workflow. Additionally, mainframe carriage control characters and channel command words can affect the look of the printed page and might need special handling. As part of the modernization planning process, we recommend that you assess and understand the configurations in your specific print environment.

Print data capture

Rocket Software Print Exit passes the necessary information for LRS VPSX/MFI to effectively process the spool file. The information consists of fields passed in the relevant control blocks, such as the following:

- JOBNAME
- OWNER (USERID)
- DESTINATION

- FORM
- FILENAME
- WRITER

LRS VPSX/MFI supports the following mainframe batch mechanisms for capturing data from Rocket Enterprise Server:

- BATCH COBOL print/spool processing using standard z/OS JCL SYSOUT DD/OUTPUT statements.
- BATCH COBOL print/spool processing using standard z/OS JCL CA-SPOOL SUBSYS DD statements.
- IMS/COBOL print/spool processing using the CBLTDLI interface. For a full list of supported methods and programming examples, see the LRS documentation that's included with your product license.

Printer-fleet health checks

LRS VPSX/MFI (LRS LoadX) can perform deep dive health checks, including device management and operational optimization. Device management can detect failure in a printer device and route the print request to a healthy printer. For more information about deep-dive health checks for printer fleets, see the LRS documentation that's included with your product license.

Print authentication and authorization

LRS/DIS enables LRS applications to authenticate user IDs and passwords by using Microsoft Active Directory or a Lightweight Directory Access Protocol (LDAP) server. In addition to basic print authorization, LRS/DIS can also apply granular-level print security controls in the following use cases:

- Manage who can browse the printer job.
- Manage the browsing level of other user's jobs.
- Manage operational tasks—for example, command-level security such as hold or release, purge, modify, copy, and reroute. Security can be set up by either the user-ID or the group, similar to an Active Directory security group or an LDAP group.

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Modernize mainframe batch printing workloads on AWS by using Rocket Enterprise Server and LRS VPSX/MFI

Shubham Roy and Kevin Yung, Amazon Web Services

Abraham Rondon, Micro Focus

Guy Tucker, Levi, Ray and Shoup Inc

Summary

This pattern shows you how to modernize your business-critical mainframe batch printing workloads on the Amazon Web Services (AWS) Cloud by using Rocket Enterprise Server as a runtime for a modernized mainframe application and LRS VPSX/MFI (Micro Focus Interface) as a print server. The pattern is based on the [replatform](#) mainframe modernization approach. In this approach, you migrate your mainframe batch jobs to Amazon Elastic Compute Cloud (Amazon EC2) and migrate your mainframe database, such as IBM DB2 for z/OS, to Amazon Relational Database Service (Amazon RDS). The authentication and authorization for the modernized print workflow is performed by AWS Directory Service for Microsoft Active Directory, also known as AWS Managed Microsoft AD. The LRS Directory Information Server (LRS/DIS) is integrated with AWS Managed Microsoft AD. By modernizing your batch printing workloads, you can reduce IT infrastructure costs, mitigate the technical debt of maintaining legacy systems, remove data silos, increase agility and efficiency with a DevOps model, and take advantage of on-demand resources and automation in the AWS Cloud.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A mainframe printing or output management workload
- Basic knowledge of how to rebuild and deliver a mainframe application that runs on Rocket Enterprise Server (For more information, see the Rocket Enterprise Server data sheet in the Rocket documentation.)
- Basic knowledge of LRS cloud printing solutions and concepts
- Rocket Enterprise Server software and license (For more information, contact Rocket sales.)
- LRS VPSX/MFI, LRS/Queue, and LRS/DIS software and licenses (For more information, contact LRS sales.)

Note

For more information about configuration considerations for mainframe batch printing workloads, see *Considerations* in the Additional information section of this pattern.

Product versions

- Rocket Enterprise Server 6.0 (product update 7)
- LRS VPSX/MFI V1R3 or higher

Architecture

Source technology stack

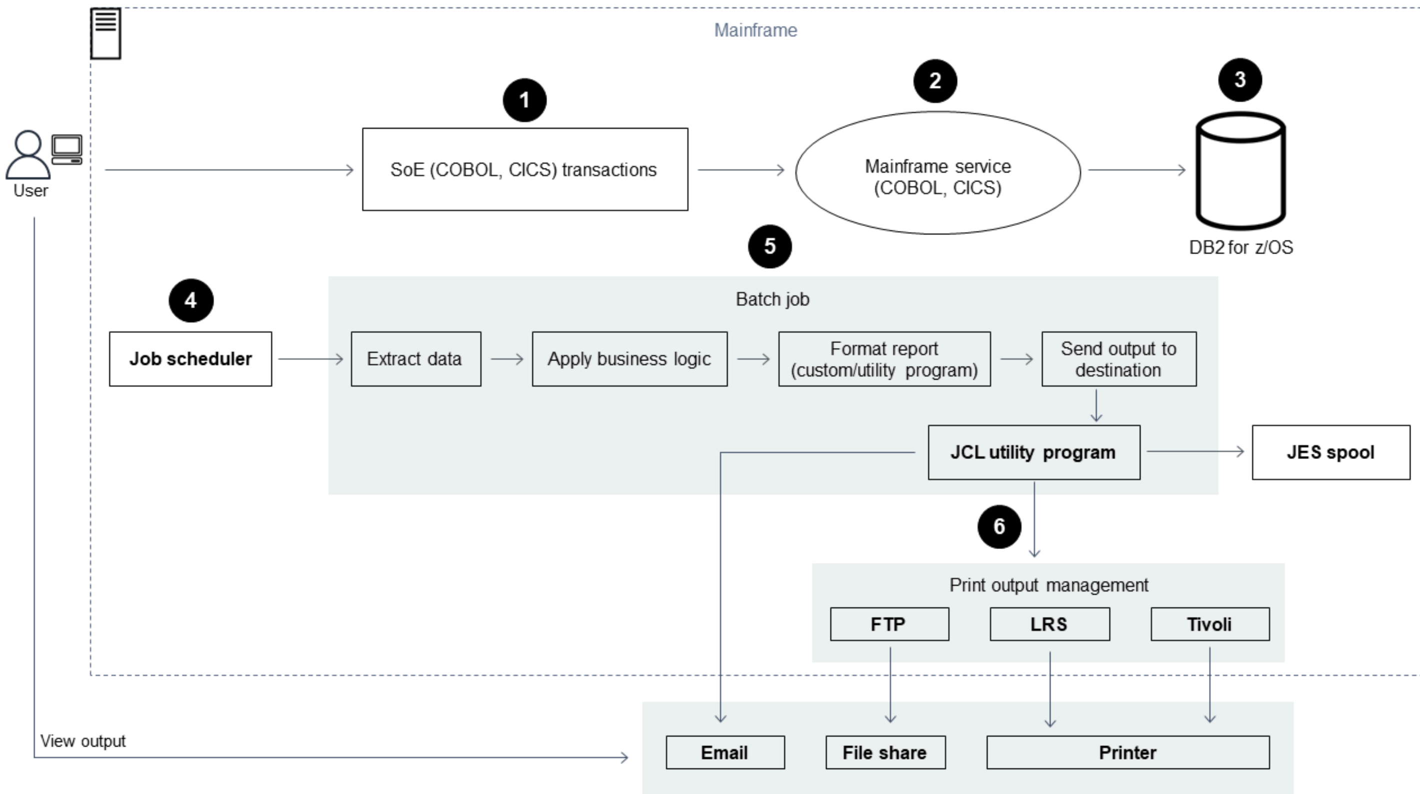
- Operating system – IBM z/OS
- Programming language – Common Business-Oriented Language (COBOL), job control language (JCL), and Customer Information Control System (CICS)
- Database – IBM DB2 for z/OS and Virtual Storage Access Method (VSAM)
- Security – Resource Access Control Facility (RACF), CA Top Secret for z/OS, and Access Control Facility 2 (ACF2)
- Printing and output management – IBM mainframe z/OS printing products (IBM Tivoli Output Manager for z/OS, LRS, and CA View)

Target technology stack

- Operating system – Microsoft Windows Server running on Amazon EC2
- Compute – Amazon EC2
- Programming language – COBOL, JCL, and CICS
- Database – Amazon RDS
- Security – AWS Managed Microsoft AD
- Printing and output management – LRS printing solution on AWS
- Mainframe runtime environment – Rocket Enterprise Server

Source architecture

The following diagram shows a typical current state architecture for a mainframe batch printing workload:

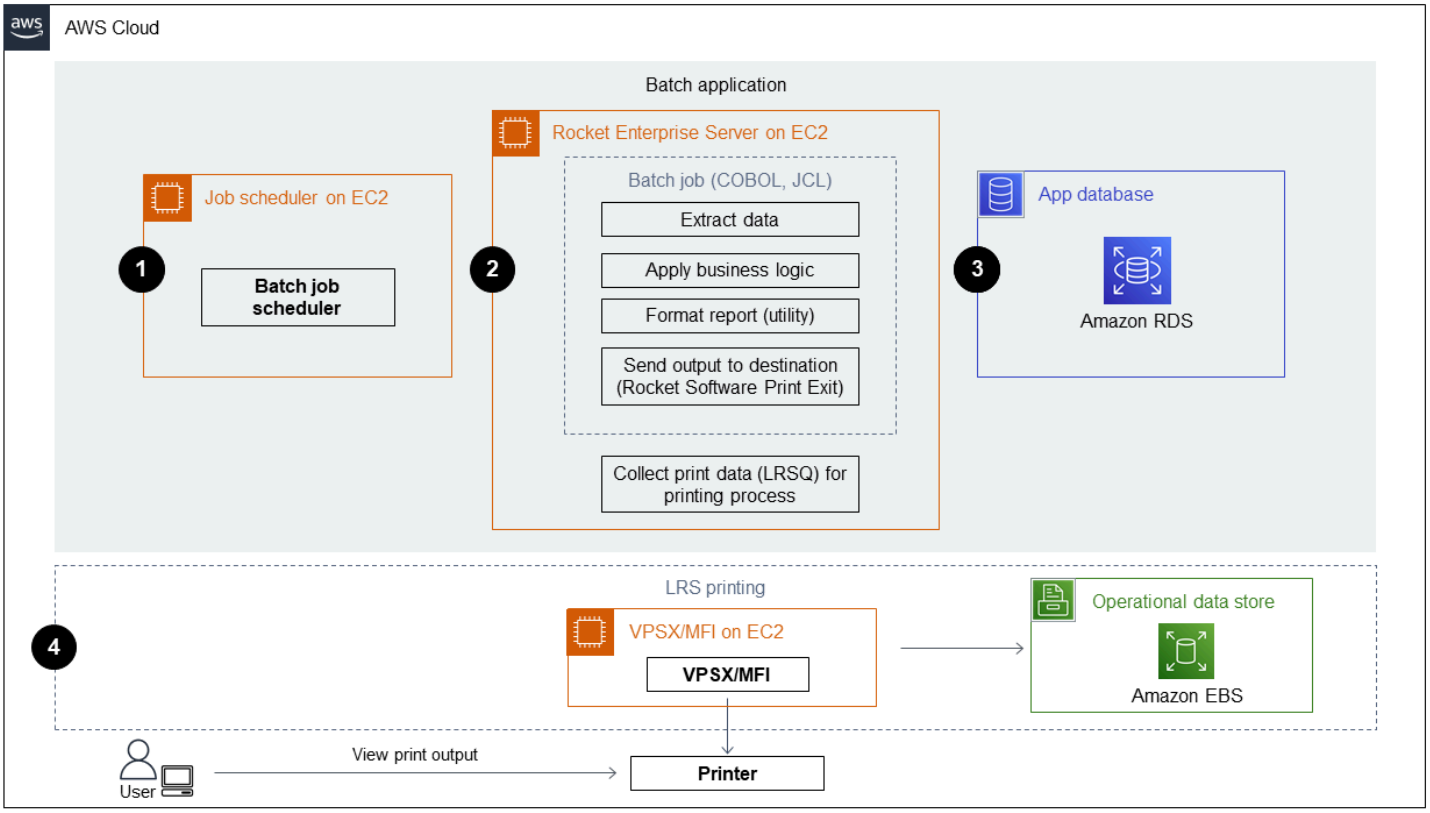


The diagram shows the following workflow:

1. Users perform business transactions on a system of engagement (SoE) that's built on an IBM CICS application written in COBOL.
2. The SoE invokes the mainframe service, which records the business transaction data in a system-of-records (SoR) database such as IBM DB2 for z/OS.
3. The SoR persists the business data from the SoE.
4. The batch job scheduler initiates a batch job to generate print output.
5. The batch job extracts data from the database, formats the data according to business requirements, and then generates business output such as billing statements, ID cards, or loan statements. Finally, the batch job routes the output to printing output management for processing and output delivery, based on the business requirements.
6. Printing output management receives print output from the batch job, and then delivers that output to a specified destination, such as email, a file share that uses secure FTP, a physical printer that uses LRS printing solutions (as demonstrated in this pattern), or IBM Tivoli.

Target architecture

The following diagram shows an architecture for a mainframe batch printing workload that's deployed in the AWS Cloud:



The diagram shows the following workflow:

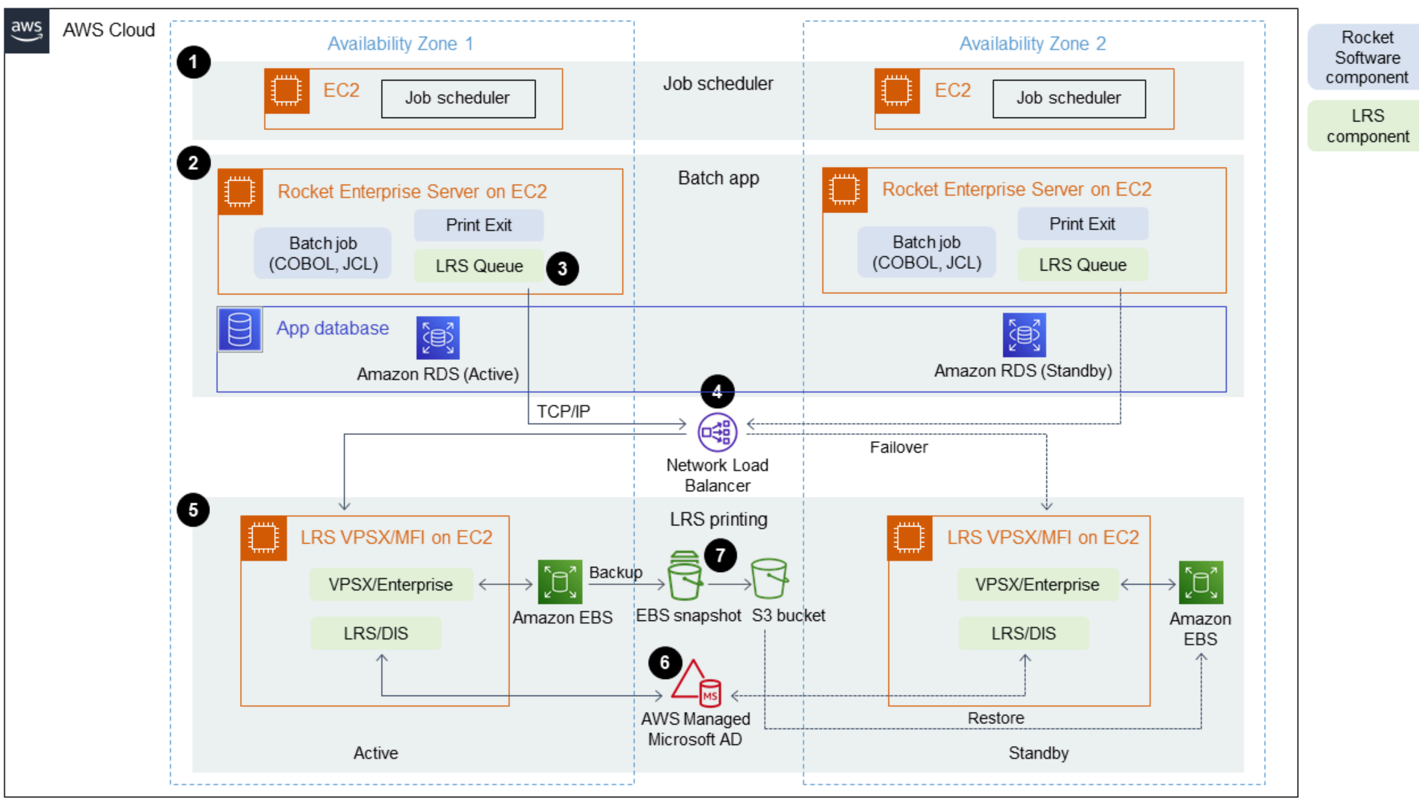
1. The batch job scheduler initiates a batch job to create print output, such as billing statements, ID cards, or loan statements.
2. The mainframe batch job (replatformed to Amazon EC2) uses the Rocket Enterprise Server runtime to extract data from the application database, apply business logic to the data, format the data, and then send the data to a print destination by using Rocket Software Print Exit (Micro Focus documentation).
3. The application database (an SoR that runs on Amazon RDS) persists data for print output.
4. The LRS VPSX/MFI printing solution is deployed on Amazon EC2 and its operational data is stored in Amazon Elastic Block Store (Amazon EBS). LRS VPSX/MFI uses the TCP/IP-based LRS/Queue transmission agent to collect print data through the Rocket Software JES Print Exit API and deliver the data to a specified printer destination.

Note

The target solution typically doesn't require application changes to accommodate mainframe formatting languages, such as IBM Advanced Function Presentation (AFP) or Xerox Line Condition Data Stream (LCDS). For more information about using Rocket Software for mainframe application migration and modernization on AWS, see the Empowering Enterprise Mainframe Workloads on AWS with Micro Focus blog post.

AWS infrastructure architecture

The following diagram shows a highly available and secure AWS infrastructure architecture for a mainframe batch printing workload:



The diagram shows the following workflow:

1. The batch scheduler initiates the batch process and is deployed on Amazon EC2 across multiple Availability Zones for high availability (HA).

> **Note**
>
> This pattern doesn't cover the implementation of the batch scheduler. For more information about implementation, see the software vendor documentation for your scheduler.

2. The mainframe batch job (written on a programming language such as JCL or COBOL) uses core business logic to process and generate print output, such as billing statements, ID cards, and loan statements. The job is deployed on Amazon EC2 across two Availability Zones for HA and uses Rocket Software Print Exit to route print output to LRS VPSX/MFI for end-user printing.
3. LRS VPSX/MFI uses a TCP/IP-based LRS/Queue transmission agent to collect or capture print data from the Rocket Software JES Print Exit programming interface. Print Exit passes the necessary information to enable LRS VPSX/MFI to effectively process the spool file and dynamically build LRS/Queue commands. The commands are then run using a standard built-in function from Rocket Software.

> **Note**
>
> For more information on print data passed from Rocket Software Print Exit to LRS/Queue and LRS VPSX/MFI supported mainframe batch mechanisms, see *Print data capture* in the Additional information section of this pattern.

4. > **Note**
   >
   > A Network Load Balancer provides a DNS name to integrate Rocket Enterprise Server with LRS VPSX/MFI. : LRS VPSX/MFI supports a Layer 4 load balancer. The Network Load Balancer also does a basic health check on LRS VPSX/MFI and routes traffic to the registered targets that are healthy.

5. > **Note**
   >
   > The LRS VPSX/MFI print server is deployed on Amazon EC2 across two Availability Zones for HA and uses Amazon EBS as an operational data store. LRS VPSX/MFI supports both the active-active and active-passive service modes. This architecture uses multiple AZs in an active-passive pair as an active and hot standby. The Network Load Balancer performs a health check on LRS VPSX/MFI EC2 instances and routes traffic to hot

standby instances in the other AZ if an active instance is in an unhealthy state. The print requests are persisted in the LRS Job Queue locally in each of the EC2 instances. In the event of recovery, a failed instance has to be restarted for the LRS services to resume processing the print request. : LRS VPSX/MFI can also perform health checks at the printer fleet level. For more information, see *Printer fleet health checks* in the Additional information section of this pattern.

6. AWS Managed Microsoft AD integrates with LRS/DIS to perform print workflow authentication and authorization. For more information, see *Print authentication and authorization* in the Additional information section of this pattern.
7. LRS VPSX/MFI uses Amazon EBS for block storage. You can back up Amazon EBS data from active EC2 instances to Amazon S3 as point-in-time snapshots and restore them to hot standby EBS volumes. To automate the creation, retention, and deletion of Amazon EBS volume snapshots, you can use Amazon Data Lifecycle Manager to set the frequency of automated snapshots and restore them based on your RTO/RPO requirements.

Tools

AWS services

- Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can use Amazon EC2 to launch as many or as few virtual servers as you need, and you can scale out or scale in.
- Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud. It provides cost-efficient, resizable capacity for a relational database and manages common database administration tasks.
- AWS Directory Service for Microsoft Active Directory, also known as AWS Managed Microsoft AD, enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud.

Other tools

- LRS VPSX/MFI (Micro Focus Interface), co-developed by LRS and Rocket Software, captures output from a Rocket Enterprise Server JES spool and reliably delivers it to a specified print destination.
- LRS Directory Information Server (LRS/DIS) is used for authentication and authorization during the print workflow.
- TCP/IP-based LRS/Queue transmission agent is used by LRS VPSX/MFI to collect or capture print data through the Rocket Software JES Print Exit programming interface.
- Rocket Enterprise Server is an application deployment environment for mainframe applications. It provides the execution environment for mainframe applications that are migrated or created by using any version of Rocket Software Enterprise Developer.

Epics

Set up Rocket Enterprise Server on Amazon EC2 and deploy a mainframe batch application

Task	Description	Skills required
Set up Rocket Enterprise Server and deploy a demo application.	Set up Rocket Enterpris e Server on Amazon EC2, and then deploy the Rocket Software BankDemo demonstration application on Amazon EC2. The BankDemo application is a mainframe batch applicati on that creates and then initiates print output.	Cloud architect

Set up an LRS print server on Amazon EC2

Task	Description	Skills required
Get an LRS product license for printing.	To get an LRS product license for LRS VPSX/MFI, LRS/ Queue, and LRS/DIS, contact	Build lead

Task	Description	Skills required
	the LRS Output Management team. You must provide the host names of the EC2 instances where the LRS products will be installed.	

Task	Description	Skills required
Create an Amazon EC2 Windows instance to install LRS VPSX/MFI.	Launch an Amazon EC2 Windows instance by following the instructions from [Launch an Amazon EC2 instance](#) in the Amazon EC2 documentation. Your instance must meet the following hardware and software requirements for LRS VPSX/MFI: • CPU – Dual Core • RAM – 16 GB • Drive – 500 GB • Minimum EC2 instance – m5.xlarge • OS – Windows/Linux • Software – Internet Information Service (IIS) or Apache **Note** The preceding hardware and software requirements are intended for a small printer fleet (around 500–1000). To get the full requirements, consult with your LRS and AWS contacts.	Cloud architect

Task	Description	Skills required
	When you create your Windows instance, do the following: 1. Confirm that the EC2 host name is the same host name used for the LRS product license. 2. Enable CGI in Amazon EC2 by completing the following: a. Connect to your EC2 instance by following the instructions from	

Task	Description	Skills required
Install LRS VPSX/MFI on the EC2 instance.	1. Connect to your EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. **Note** Open the link to the product download page from the LRS email that you should receive. : LRS products are distributed by electronic file transfer (EFT). 3. Download LRS VPSX/MFI and unzip the file (default folder: `c:\LRS`). 4. Launch the LRS Product Installer from the unzipped folder to install LRS VPSX/MFI. 5. In the **Select Features** menu, select **VPSX® Server (V1R3.022)**, and then choose **Next** to start the installation process. You will receive a success message when installation is complete.	Cloud architect

Task	Description	Skills required
Install LRS/Queue.	1. Connect to your Rocket Enterprise Server EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. Open the link to the LRS product download page from the LRS email that you should receive, download LRS/Queue, and then unzip the file. 3. Go to the location where you downloaded the files, and then launch the LRS product installer to install LRS/Queue.	Cloud architect

Task	Description	Skills required
Install LRS/DIS.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructi ons from [Step 2: Connect to your instance](#) in the Amazon EC2 documenta tion. 2. Open the link to the LRS product download page from the LRS email that you should receive, download LRS/DIS, and then unzip the file. 3. Go to the location where you downloaded the files, and then launch the LRS Product Installer. 4. In the LRS Product Installer , expand **LRS Misc Tools**, select **LRS DIS**, and then choose **Next**. 5. Follow the rest of the instructions in the LRS Product Installer to complete the installation process.	Cloud architect

Task	Description	Skills required
Create a target group and register LRS VPSX/MFI EC2 as the target.	Create a target group by following the instructions from [Create a target group for your Network Load Balancer](#) in the Elastic Load Balancing documentation. When you create the target group, do the following: 1. On the **Specify group details** page, for **Choose a Target Type**, choose **Instances**. 2. For **Protocol**, choose **TCP**. 3. For **Port**, choose **5500**. 4. On the **Register targets** page, in the **Available instances** section, select the LRS VPSX/MFI EC2 instances.	Cloud architect

Task	Description	Skills required
Create a Network Load Balancer.	Follow the instructions from [Create a Network Load Balancer](#) in the Elastic Load Balancing documentation. Your Network Load Balancer routes traffic from Rocket Enterprise Server to LRS VPSX/MFI EC2. When you create the Network Load Balancer, do the following on the **Listeners and Routing** page: 1. For **Protocol**, choose **TCP**. 2. For **Port**, choose **5500**. 3. For **Default action**, choose **Forward to** for the target group that you created earlier.	Cloud architect

Integrate Rocket Enterprise Server with LRS VPSX/MFI and LRS/Queue

Task	Description	Skills required
Configure Rocket Enterpris e Server for LRS/Queue integration.	1. Connect to your Rocket Enterprise Server EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. In the Windows Start menu, open the Rocket	Cloud architect

Task	Description	Skills required
	Enterprise Server Administration UI. 3. In the menu bar, choose **NATIVE**. 4. In the navigation pane, choose **Directory Server**, and then choose **BANKDEMO**. 5. From **General** in the left navigation pane, scroll down to the **Additional** section to configure the environment variables (LRSQ_ADDRESS, LRSQ_PORT, LRSQ_COMMAND) to point to LRSQ. 6. For **LRSQ_ADDRESS**, enter the IP address or DNS name of the Network Load Balancer that you created earlier. 7. For **LRSQ_PORT**, enter **VPSX LRSQ Listener Port (5500)**. 8. For **LRSQ_COMMAND**, enter the path location of the LRSQ executable. **Note** LRS currently supports a maximum character limit of	

Task	Description	Skills required
	50 for DNS names, but this is subject to change in the future. If your DNS name is greater than 50, then you can use the IP address of the Network Load Balancer as an alternative.	

Task	Description	Skills required
Configure Rocket Enterpris e Server for LRS VPSX/MFI integration.	1. Copy the `VPSX_MFI_R2` folder from the LRS VPSX/MFI installer to the Rocket Enterprise Server location at `C\BANKDEMO\print` . 2. Connect to your Rocket Enterprise Server EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 3. In the Windows Start menu, open the Rocket Enterprise Server Administr ation UI. 4. In the menu bar, choose **NATIVE**. 5. In the navigation pane, choose **Directory Server**, and then choose **BANKDEMO**. 6. Under **BANKDEMO**, choose **JES**. 7. Under **JES Program Path**, add the `DLL(VPSX_ MFI_R2)` path from the `C\BANKDEMO\print` location.	Cloud architect

Set up printers and print users in Rocket Enterprise Server and LRS VPSX/MFI

Task	Description	Skills required
Associate the Rocket Software Print Exit module to the Rocket Enterprise Server batch printer Server Execution Process.	1. Connect to your Rocket Enterprise Server EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. In the Windows Start menu, open the Rocket Enterprise Server Administration UI. 3. In the menu bar, choose **NATIVE**. 4. In the navigation pane, choose **Directory Server**, and then choose **BANKDEMO**. 5. Under **BANKDEMO**, choose **JES**, and scroll down to **Printers**. 6. In **Printers**, associate the Rocket Software Print Exit module (LRSPRTE6 for Batch) to the Rocket Enterprise Server batch printer Server Execution Process (SEP). This allows print output routing to LRS VPSX/MFI. 7. Sign in to the Enterprise Server Administration UI.	Cloud architect

Task	Description	Skills required
	For more information about configuration, see Using the Exit in the Micro Focus documentation.	

Task	Description	Skills required
Add a printer in LRS VPSX/MFI.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. Open the VPSX Web Interface from the Windows Start menu. 3. In the navigation pane, choose **Printers**. 4. Choose **Add**, and then choose **Add Printer**. 5. On the **Printer Configuration** page, for **Printer Name**, enter **Local**. 6. For **VPSX ID**, enter **VPS1**. 7. For **CommType**, select **TCPIP/LRSQ**. 8. For **Host/IP Address**, enter the IP address of the physical printer that you want to add. 9. For **Device**, enter the name of your device. 10. Choose either **Windows Driver** or **Linux/Mac Driver**. 11. Choose **Add**.	Cloud architect

Task	Description	Skills required
Create a print user in LRS VPSX/MFI.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. Open the VPSX Web Interface from the Windows Start menu. 3. In the navigation pane, choose **Security**, and then choose **Users**. 4. In the **User Name** column, choose **admin**, and then choose **Copy**. 5. In the **User Profile Maintenance** window, for **User Name**, enter a user name (for example, **PrintUser**). 6. For **Description**, enter a brief description (for example, **User for test print**). 7. Choose **Update**. This creates a print user (for example, **PrintUser**). 8. In the navigation pane, under **User**, choose the new user that you created. 9. From the **Command** menu, choose **Security**.	Cloud architect

Task	Description	Skills required
	10. On the **Security Rules** page, choose all the applicable printer security and job security options, and then choose **Save**. 11. To add your new print user to the **Administrator** group, go to the navigation pane, choose **Security**, and then choose **Configure**. 12. In the **Security configuration** window, add your new print user to the **Administrator** column.	

Set up print authentication and authorization

Task	Description	Skills required
Create an AWS Managed Microsoft AD domain with users and groups.	1. Create an Active Directory on AWS Managed Microsoft AD by following the instructions from [Create your AWS Managed Microsoft AD directory](#) in the AWS Directory Service documentation. 2. Deploy an EC2 instance (Active Directory manager) and install Active Directory tools to manage your AWS Managed Microsoft AD by following the instructions from [Step 3: Deploy an EC2](#)	Cloud architect

Task	Description	Skills required
	[instance to manage your AWS Managed Microsoft AD](#) in the AWS Directory Service documentation. 3. **Note** Connect to your EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documenta tion. : When you connect to the EC2 instance, enter your administrator credentials (for the directory that you created in step one) in the Windows Security window. 4. In the Windows Start menu, under **Windows Administrative Tools**, choose **Active Directory Users and Computers**. 5. Create a print user in the Active Directory domain by following the steps from [Create a user](#) in the	

Task	Description	Skills required
	AWS Directory service documentation.	
Join LRS VPSX/MFI EC2 to an AWS Managed Microsoft AD domain.	Join LRS VPSX/MFI EC2 to your AWS Managed Microsoft AD domain automatically (AWS Knowledge Center documentation) or manually (AWS Directory Service documentation).	Cloud architect

Task	Description	Skills required
Configure and integrate LRS/DIS with AWS Managed Microsoft AD.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. In the Windows Start menu, open the VPSX Web Interface. 3. In the navigation pane, choose **Security**, and then choose **Configure**. 4. On the **Security Configuration** page, in the **Security Parameters** section, for **Security Type**, select **Internal**. 5. Enter your preferences for the rest of the options in the **Security Parameters** section. 6. Open the **LRS Output Management** folder from the Microsoft Windows Start menu, choose **Server Start**, and then choose **Server Stop**. 7. Log in to LRS VPSX/MFI with your Active Directory user name and password.	Cloud architect

Test a print workflow

Task	Description	Skills required
Initiate a batch print request from the Rocket Software BankDemo app.	1. Open the 3270 terminal emulator in your Rocket Enterprise Server EC2 instance. 2. Connect to the BankDemo app by running the following command: `connect 127.0.0.1:9278` 3. On the BankDemo command line interface, for **User id**, enter **B0001**. For **Password**, enter a non-blank key. 4. For the **Request printed statement(s)** option, enter **X** on the blank line. 5. In the **Send statement by** section, for **Mail**, enter **Y**, and then press F10.	Test engineer
Check the print output in LRS VPSX/MFI.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. In Windows Start menu, open the VPSX Web Interface.	Test engineer

Task	Description	Skills required
	3. In the navigation pane, choose **Printers**, and then choose **Output Queue**. 4. In the **Spool ID** column, choose the spool ID for the request in the printer queue. 5. On the **Actions** tab, in the **COMMAND** column, choose **Browse**. You can now see the print output of an account statement with columns for **Account No.**, **Descripti on**, **Date**, **Amount**, and **Balance**. For an example, see the **batch_print_output** attachment for this pattern.	

Related resources

- LRS Output Modernization (LRS documentation)
- ANSI and machine carriage controls (IBM documentation)
- Channel command words (IBM documentation)
- Empowering Enterprise Mainframe Workloads on AWS with Micro Focus (AWS Partner Network Blog)
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager (AWS Prescriptive Guidance documentation)
- Advanced Function Presentation (AFP) data stream (IBM documentation)
- Line Conditioned Data Stream (LCDS) (Compart documentation)

Additional information

Considerations

During your modernization journey, you may consider a wide variety of configurations for both mainframe batch processes and the output they generate. The mainframe platform has been customized by every customer and vendor that uses it with particular requirements that directly affect print. For example, your current platform may incorporate the IBM Advanced Function Presentation (AFP) or the Xerox Line Condition Data Stream (LCDS) into the current workflow. Additionally, mainframe carriage control characters and channel command words can affect the look of the printed page and may need special handling. As part of the modernization planning process, we recommend that you assess and understand the configurations in your specific print environment.

Print data capture

Rocket Software Print Exit passes the necessary information to enable LRS VPSX/MFI to effectively process the spool file. The information consists of fields passed in the relevant control blocks, such as:

- JOBNAME
- OWNER (USERID)
- DESTINATION
- FORM
- FILENAME
- WRITER

LRS VPSX/MFI supports the following mainframe batch mechanisms for capturing data from Rocket Enterprise Server.

- BATCH COBOL print/spool processing using standard z/OS JCL SYSOUT DD/OUTPUT statements
- BATCH COBOL print/spool processing using standard z/OS JCL CA-SPOOL SUBSYS DD statements
- IMS/COBOL print/spool processing using the CBLTDLI interface (For a full list of supported methods and programming examples, see the LRS documentation that's included with your product license.)

Printer fleet health checks

LRS VPSX/MFI (LRS LoadX) can perform deep dive health checks, including device management and operational optimization. Device management can detect failure in a printer device and route the print request to a healthy printer. For more information about deep dive health checks for printer fleets, see the LRS documentation that's included with your product license.

Print authentication and authorization

LRS/DIS enables LRS applications to authenticate user IDs and passwords by using Microsoft Active Directory or an LDAP server. In addition to basic print authorization, LRS/DIS can also apply granular-level print security controls in the following use cases:

- Manage who can browse the printer job.
- Manage the browsing level of other user's jobs.
- Manage operational tasks. For example, command-level security such as hold/release, purge, modify, copy, and reroute. Security can be set up by either the User-ID or Group (similar to AD group or LDAP group).

Attachments

To access additional content that is associated with this document, unzip the following file: [attachment.zip]

Mainframe modernization: DevOps on AWS with Rocket Software Enterprise Suite

Kevin Yung, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

Customer challenges

Organizations that run core applications on mainframe hardware usually encounter a few challenges when the hardware needs to scale up to meet the demands of digital innovations. These challenges include the following constraints.

- Mainframe development and test environments are unable to scale due to the inflexibility of mainframe hardware components and the high cost of changing.
- Mainframe development is facing skill shortages, because new developers are not familiar and not interested in the traditional mainframe development tools. Modern technology such as containers, continuous integration/continuous delivery (CI/CD) pipelines, and modern test frameworks are not available in mainframe development.

Pattern outcomes

To address these challenges, Amazon Web Services (AWS) and Rocket Software Micro Focus, an AWS Partner Network (APN) Partner, have collaborated to create this pattern. The solution is designed to help you achieve the following outcomes.

- Improved developer productivity. Developers can be given new mainframe development instances within minutes.
- Use of the AWS Cloud to create new mainframe test environments with virtually unlimited capacity.
- Rapid provisioning of new mainframe CI/CD infrastructure. Provisioning on AWS can be completed within an hour by using AWS CloudFormation and AWS Systems Manager.
- Native use of AWS DevOps tools for mainframe development, including AWS CodeBuild, AWS CodeCommit, AWS CodePipeline, AWS CodeDeploy, and Amazon Elastic Container Registry (Amazon ECR).

- Transform traditional waterfall development to agile development in mainframe projects.

Technologies summary

In this pattern, the target stack contains the following components.

Logical components	Implementation solutions	Description
Source code repositories	Rocket Software AccuRev Server, CodeCommit, Amazon ECR	Source code management – The solution uses two types of source code: • Mainframe source code, for example, COBOL and JCL. • AWS infrastructure templates and automation scripts Both types of source code need version control, but they are managed in different SCMs. Source code deployed into mainframe or Rocket Software Enterprise Servers is managed in Rocket Software Micro Focus AccuRev Server. AWS templates and automation scripts are managed in CodeCommit. Amazon ECR is used for the Docker image repositories.
Enterprise developer instances	Amazon Elastic Compute Cloud (Amazon EC2), Rocket Software Enterprise Developer for Eclipse	Mainframe developers can develop code in Amazon EC2 by using Rocket Software Enterprise Developer for Eclipse. This eliminates the

		need to rely on mainframe hardware to write and test code.
Rocket Software Enterprise Suite license management	Rocket Software Enterprise Suite License Manager	For centralized Rocket Software Enterprise Suite license management and governance, the solution uses Rocket Software Enterprise Suite License Manager to host the required license.
CI/CD pipelines	CodePipeline, CodeBuild, CodeDeploy, Rocket Software Enterprise Developer in a container, Rocket Software Enterprise Test Server in a container, Rocket Software Micro Focus Enterprise Server	Mainframe developme nt teams need CI/CD pipelines to perform code compilation, integration tests, and regression tests. In AWS, CodePipeline and CodeBuild can work with Rocket Software Enterprise Developer and Enterprise Test Server in a container natively.

Prerequisites and limitations

Prerequisites

Name	Description
py3270	py3270 is a Python interface to x3270, an IBM 3270 terminal emulator. It provides an API to a x3270 or s3270 subprocess.
x3270	x3270 is an IBM 3270 terminal emulator for the X Window System and Windows. This can be used by developer for unit testing locally.

Robot-Framework-Mainframe-3270-Library	Mainframe3270 is a library for Robot Framework based on py3270 project.
Rocket Software Verastream	Rocket Software Verastream is an integrati on platform that enables testing mainframe assets the way that mobile apps, web applicati ons, and SOA web services are tested.
Rocket Software Unified Functional Testing (UFT) installer and license	Rocket Software Unified Functional Testing is software that provides functional and regression test automation for software applications and environments.
Rocket Software Enterprise Server installer and license	Enterprise Server provides the runtime environment for mainframe applications.
Rocket Software Enterprise Test Server installer and license	Rocket Software Enterprise Test Server is an IBM mainframe application test environment.
Rocket Software AccuRev installer and license for Server, and Rocket Software Micro Focus AccuRev installer and license for Windows and Linux operating systems	AccuRev provides source code management (SCM). The AccuRev system is designed for use by a team of people who are developing a set of files.
Rocket Software Enterprise Developer for Eclipse installer, patch and license	Enterprise Developer provide mainframe developer a platform to develop and maintain the core mainframe online and batch applicati ons.

Limitations

- Building a Windows Docker image is not supported in CodeBuild. This reported issue needs support from Windows Kernel/HCS and Docker teams. The work-around is to create a Docker image build runbook by using Systems Manager. This pattern uses the work-around to build Rocket Software Enterprise Developer for Eclipse and Rocket Software Micro Focus Enterprise Test Server Container images.

- Virtual private cloud (VPC) connectivity from CodeBuild is not supported in Windows yet, so the pattern does not use Rocket Software License Manager to manage licenses in OpenText Rocket Software Enterprise Developer and Rocket Software Enterprise Test Server containers.

Product versions

- Rocket Software Enterprise Developer 5.5 or later
- Rocket Software Enterprise Test Server 5.5 or later
- Rocket Software Enterprise Server 5.5 or later
- Rocket Software AccuRev 7.x or later
- Windows Docker base image for Rocket Software Enterprise Developer and Enterprise Test Server: **microsoft/dotnet-framework-4.7.2-runtime**
- Linux Docker base image for AccuRev client: **amazonlinux:2**

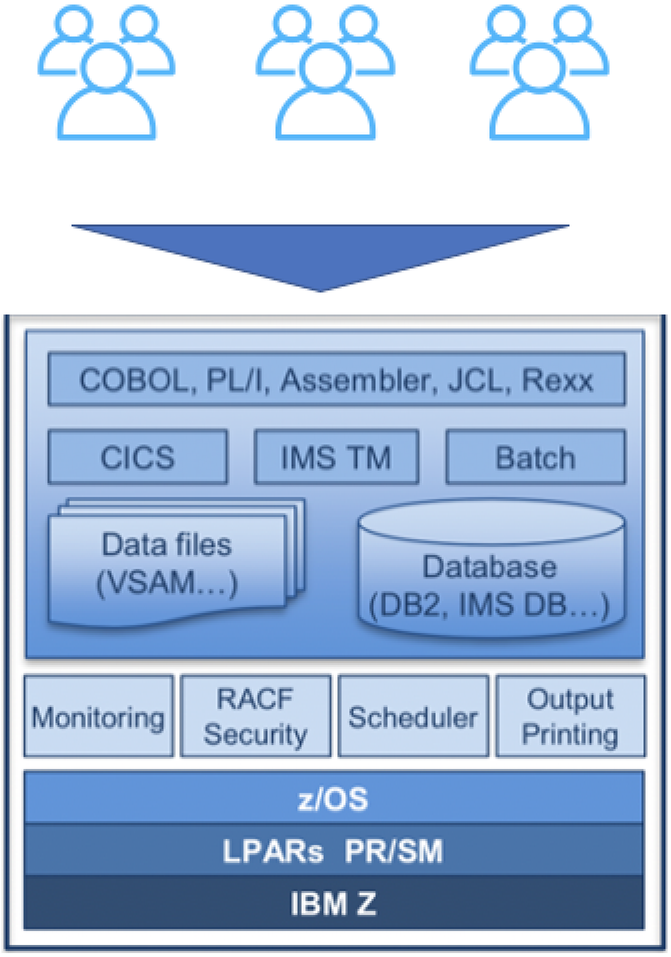
Architecture

Mainframe environment

In conventional mainframe development, the developers need to use mainframe hardware to develop and test programs. They face capacity limitations, for example restricted million instructions per second (MIPS) for the dev/test environment, and they must rely on the tools that are available on the mainframe computers.

In many organizations, mainframe development follows the waterfall development methodology, with teams relying on long cycles to release changes. These release cycles are usually longer than digital product development.

The following diagram shows multiple mainframe projects sharing mainframe hardware for their development. In mainframe hardware, it is expensive to scale out a development and test environment for more projects.



AWS **architecture**

This pattern extends mainframe development to the AWS Cloud. First, it uses AccuRev SCM to host the mainframe source code on AWS. Then it makes Enterprise Developer and Enterprise Test Server available for building and testing the mainframe code on AWS.

The following sections describe the pattern's three major components.

1. SCM

In AWS, the pattern uses AccuRev to create a set of SCM workspaces and version control for the mainframe source code. Its stream-based architecture enables parallel mainframe development for multiple teams. To merge a change, AccuRev uses the promote concept. To add that change to other workspaces, AccuRev uses the update concept.

At the project level, each team can create one or more streams in AccuRev to track project level changes. These are called project streams. These project streams are inherited from the same parent stream. The parent stream is used to merge the changes from different project streams.

Each project stream can promote code to AccuRev, and a promote post trigger is set up to initiate the AWS CI/CD pipeline. The successful build for a project stream change can be promoted to its parent stream for more regression tests.

Usually, the parent stream is called the system integration stream. When there is a promotion from a project stream to a system integration stream, a post promotion trigger initiates another CI/CD pipeline to run regression tests.

In addition to mainframe code, this pattern includes AWS CloudFormation templates, Systems Manager Automation documents, and scripts. Following infrastructure-as-code best practices, they are version-controlled in CodeCommit.

If you need to synchronize mainframe code back to a mainframe environment for deployment, Rocket Software provides the Enterprise Sync solution, which synchronizes code from the AccuRev SCM back to the mainframe SCM.

2. Developer and test environments

In a large organization, scaling more than a hundred or even more than a thousand mainframe developers is challenging. To address this constraint, the pattern uses Amazon EC2 Windows instances for development. On the instances, Enterprise Developer for Eclipse tools are installed. The developer can perform all mainframe code test and debugging locally on the instance.

AWS Systems Manager State Manager and Automation documents are used to automate the developer instance provisioning. The average time to create a developer instance is within 15 minutes. The following software and configurations are prepared:

- AccuRev Windows client for checking out and committing source code into AccuRev
- Enterprise Developers for Eclipse tool, for writing, testing, and debugging mainframe code locally
- Open source testing frameworks Python behavior-driven development (BDD) test framework Behave, py3270, and the x3270 emulator for creating scripts to test applications
- A Docker developer tool for building the Enterprise Test Server Docker image and testing the application in the Enterprise Test Server Docker container

In the development cycle, developers use the EC2 instance to develop and test mainframe code locally. When the local changes are tested successfully, developers promote the change into the AccuRev server.

3. CI/CD pipelines

In the pattern, CI/CD pipelines are used for integration tests and regression tests before deployment to the production environment.

As explained in the SCM section, AccuRev uses two types of streams: a project stream and an integration stream. Each stream is hooked up with CI/CD pipelines. To perform the integration between the AccuRev server and AWS CodePipeline, the pattern uses AccuRev post promotion script to create an event to initiate CI/CD.

For example, when a developer promotes a change to a project stream in AccuRev, it initiates a post promotion script to run in AccuRev Server. Then the script uploads the metadata of the change into an Amazon Simple Storage Service (Amazon S3) bucket to create an Amazon S3 event. This event will initiate a CodePipeline configured pipeline to run.

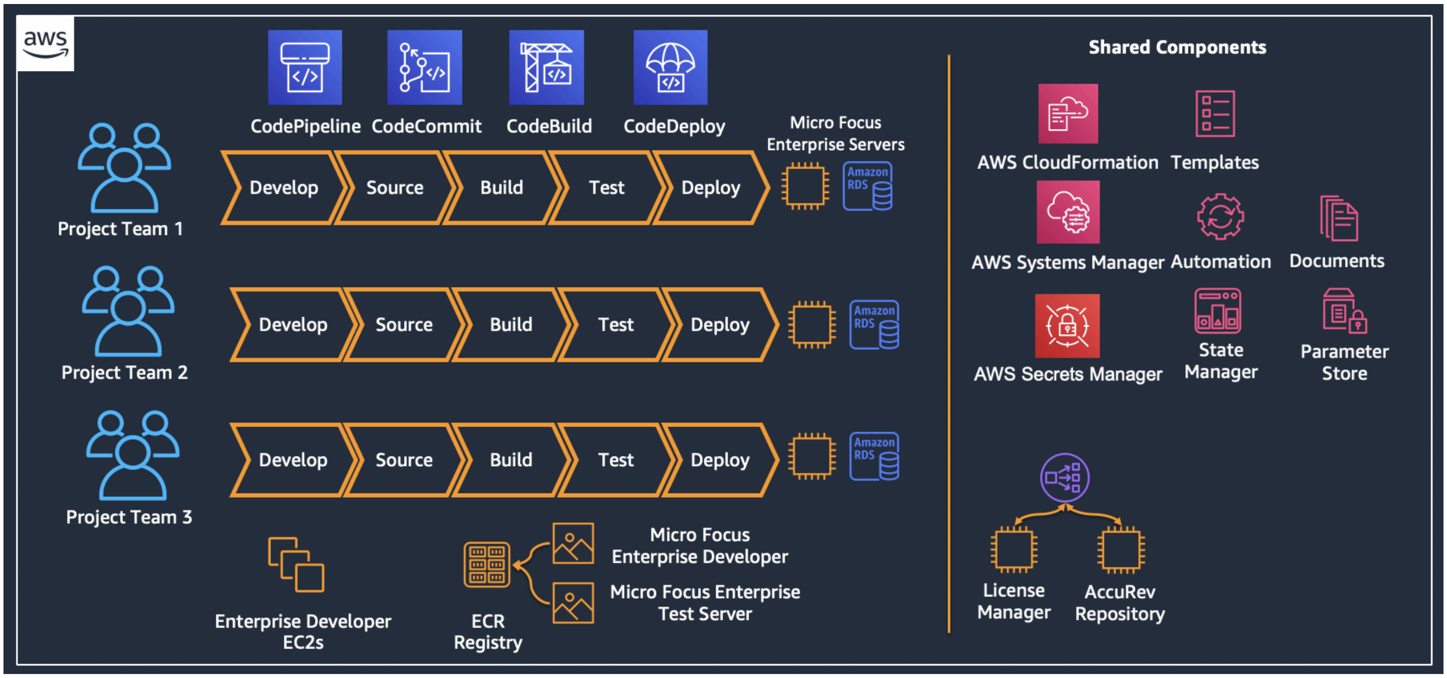
The same event-initiating mechanism is used for the integration stream and its associated pipelines.

In the CI/CD pipeline, CodePipeline uses CodeBuild with the AccuRev Linux client container to check out the latest code from the AccuRev streams. Then the pipeline starts CodeBuild to use the Enterprise Developer Windows container to compile the source code, and to use the Enterprise Test Server Windows container in CodeBuild to test mainframe applications.

The CI/CD pipelines are built using CloudFormation templates, and the blueprint will be used for new projects. By using the templates, it takes less than an hour for a project to create a new CI/CD pipeline in AWS.

To scale your mainframe test capability on AWS, the pattern builds out the Rocket Software DevOps test suite, Verastream and UFT server. By using the modern DevOps tools, you can run as many tests on AWS as you need.

An example mainframe development environment with Rocket Software on AWS is shown in the following diagram.



Target technology stack

This section provides a closer look at the architecture of each component in the pattern.

1. Source code repository – AccuRev SCM

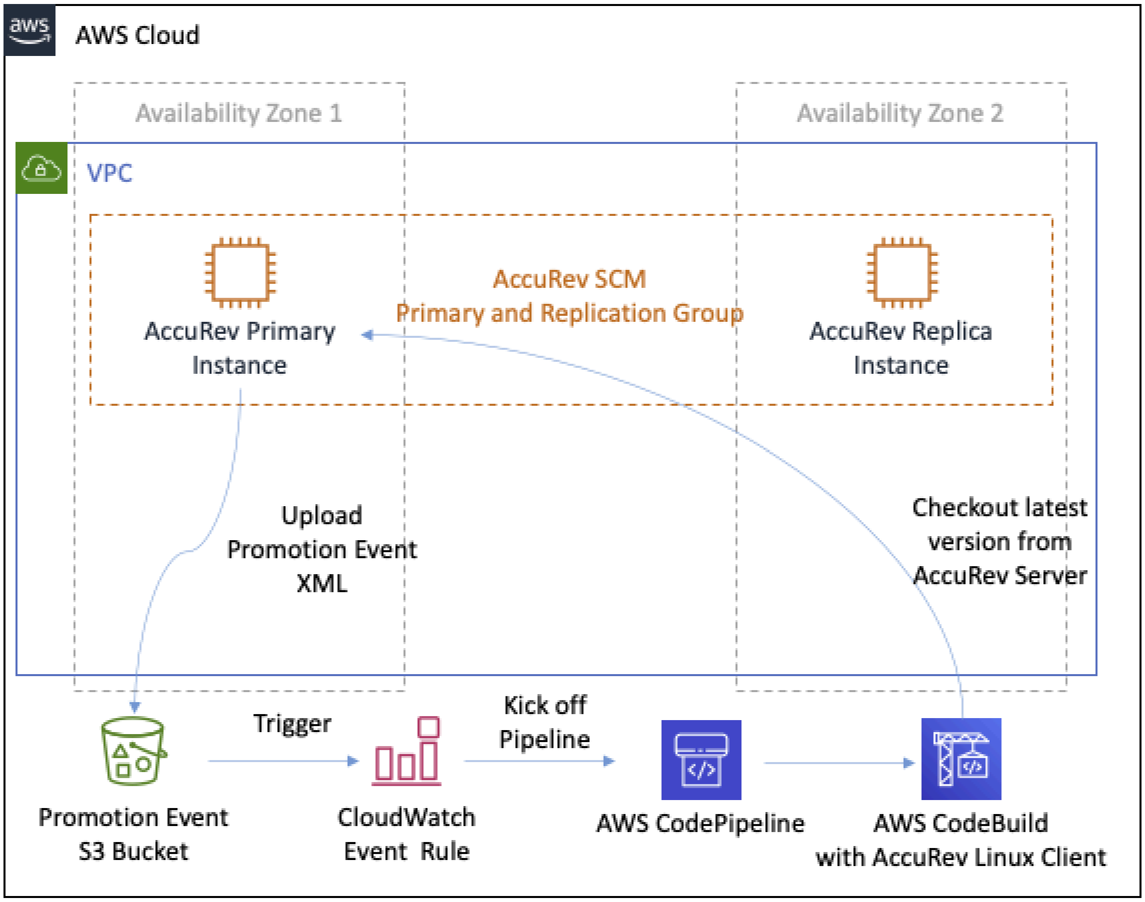
AccuRev SCM is set up to manage mainframe source code versions. For high availability, AccuRev supports primary and replica modes. Operators can fail over to the replica when performing maintenance on the primary node.

To speed up the response of the CI/CD pipeline, the pattern uses Amazon CloudWatch Events to detect source code changes and initiate the start of the pipeline.

1. The pipeline is set up to use an Amazon S3 source.
2. A CloudWatch Events rule is set up to capture S3 events from a source S3 bucket.
3. The CloudWatch Events rule sets a target to the pipeline.
4. AccuRev SCM is configured to run a post promotion script locally after promotion is complete.
5. AccuRev SCM generates an XML file that contains the metadata of the promotion, and the script uploads the XML file to the source S3 bucket.
6. After the upload, the source S3 bucket sends events to match the CloudWatch Events rule, and the CloudWatch Events rule initiates the pipeline to run.

When the pipeline runs, it kicks off a CodeBuild project to use an AccuRev Linux client container to check out the latest mainframe code from an associated AccuRev stream.

The following diagram shows an AccuRev Server setup.



2. Enterprise Developer template

The pattern uses Amazon EC2 templates to simplify creation of the developer instance. By using State Manager, it can apply software and license settings to EC2 instances consistently.

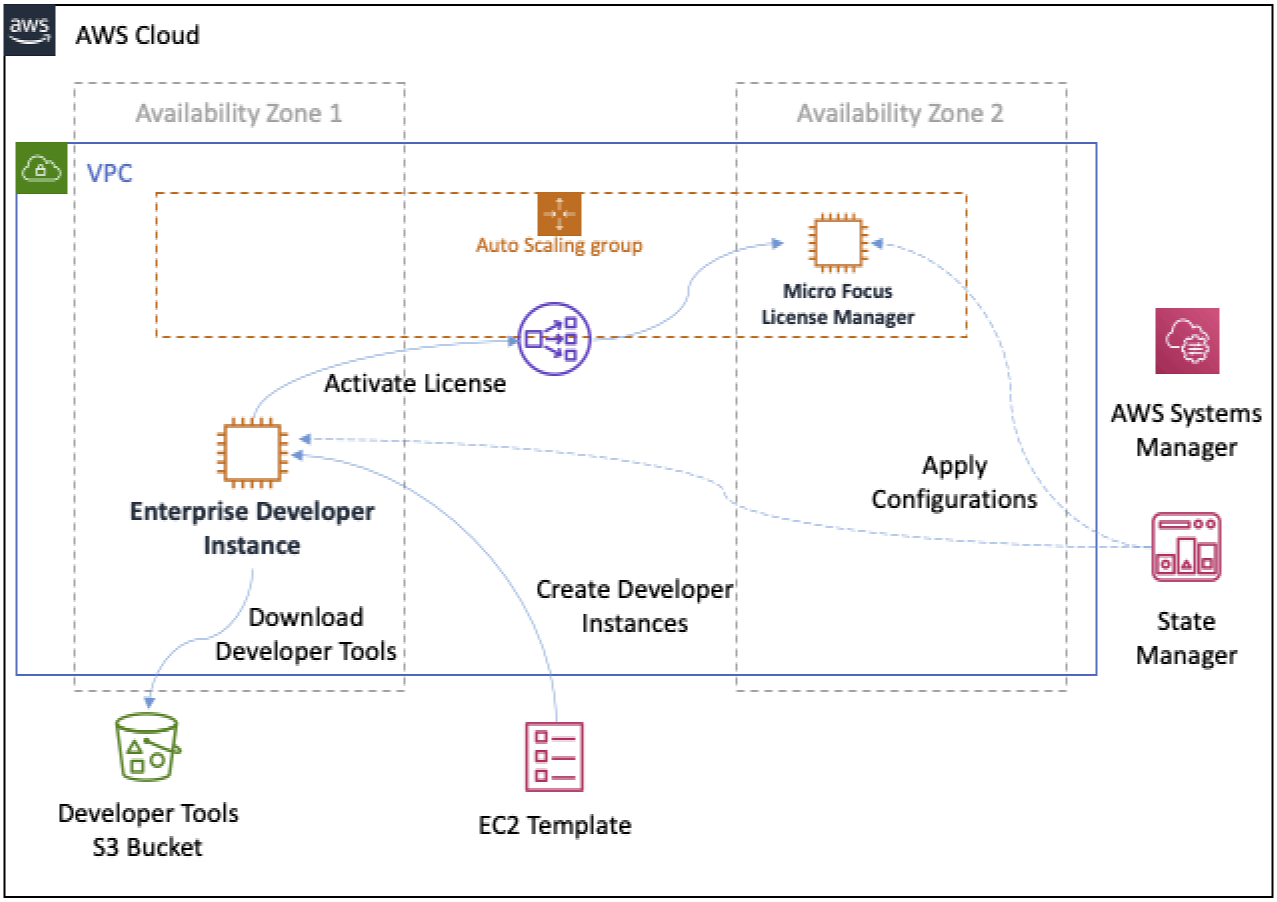
The Amazon EC2 template builds in its VPC context settings and default instance settings, and it follows enterprise tagging requirements. By using a template, a team can create their own new development instances.

When a developer instance starts, by associating with tags, Systems Manager uses State Manager to apply automation. The automation includes the following general steps.

1. Install Enterprise Developer software and install patches.
2. Install the AccuRev client for Windows.

3. Install the pre-configured script for developers to join the AccuRev stream. Initialize Eclipse workspaces.
4. Install development tools, including x3270, py3270, and Docker.
5. Configure license settings to point to a License Manager load balancer.

The following diagram shows an Enterprise developer instance created by the Amazon EC2 template, with software and configuration applied to the instance by State Manager. Enterprise developer instances connect to AWS License Manager to activate their license.



3. CI/CD pipelines

As explained in the AWS architecture section, in the pattern, there are project-level CI/CD pipelines and system integration pipelines. Each mainframe project team creates a pipeline or multiple CI/CD pipelines for building the programs that they are developing in a project. These project CI/CD pipelines check out source code from an associated AccuRev stream.

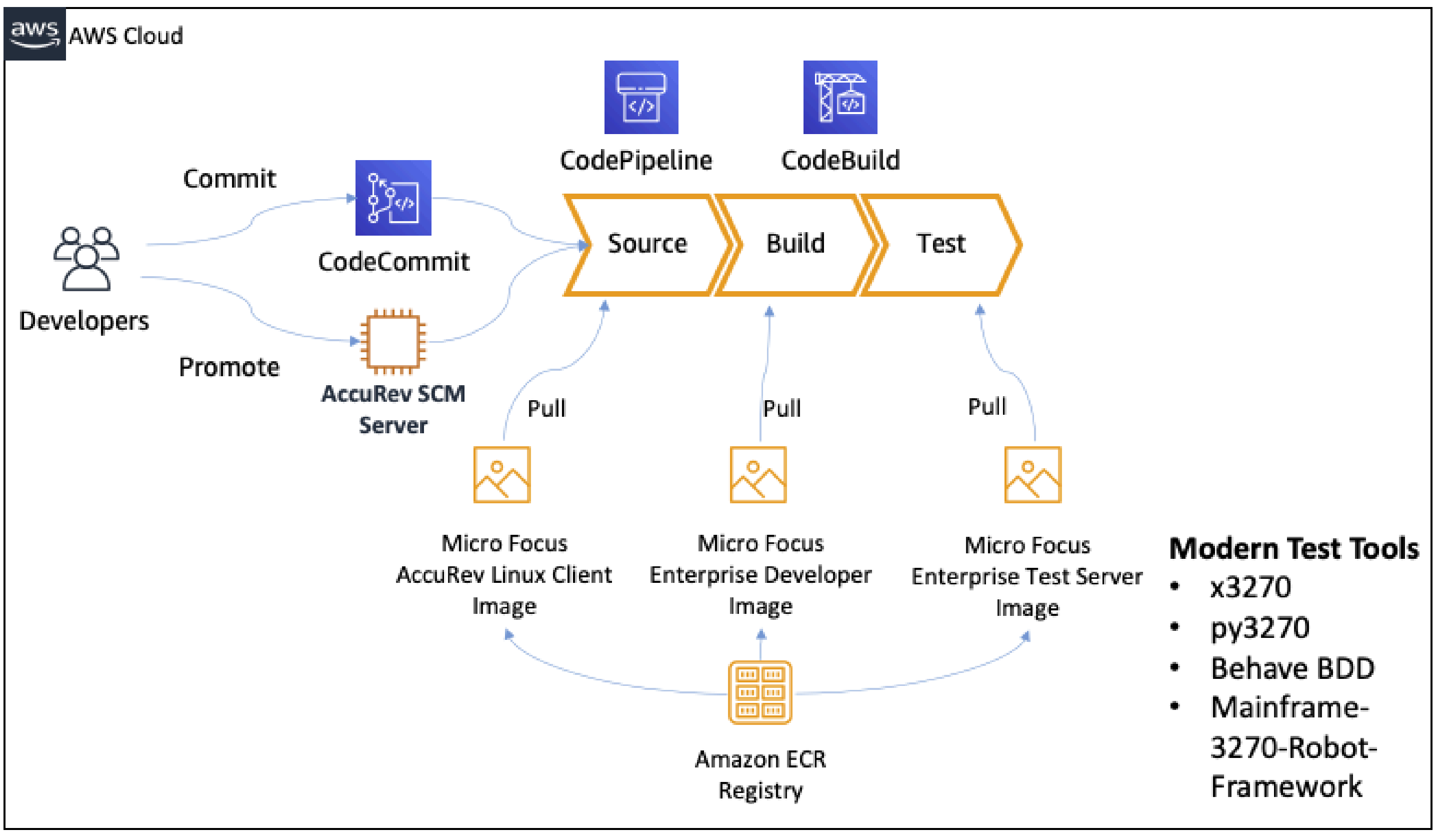
In a project team, developers promote their code in the associated AccuRev stream. Then the promotion initiates the project pipeline to build the code and run integration tests.

Each project CI/CD pipeline uses CodeBuild projects with the Enterprise Developer tool Amazon ECR image and Enterprise Test Server tool Amazon ECR image.

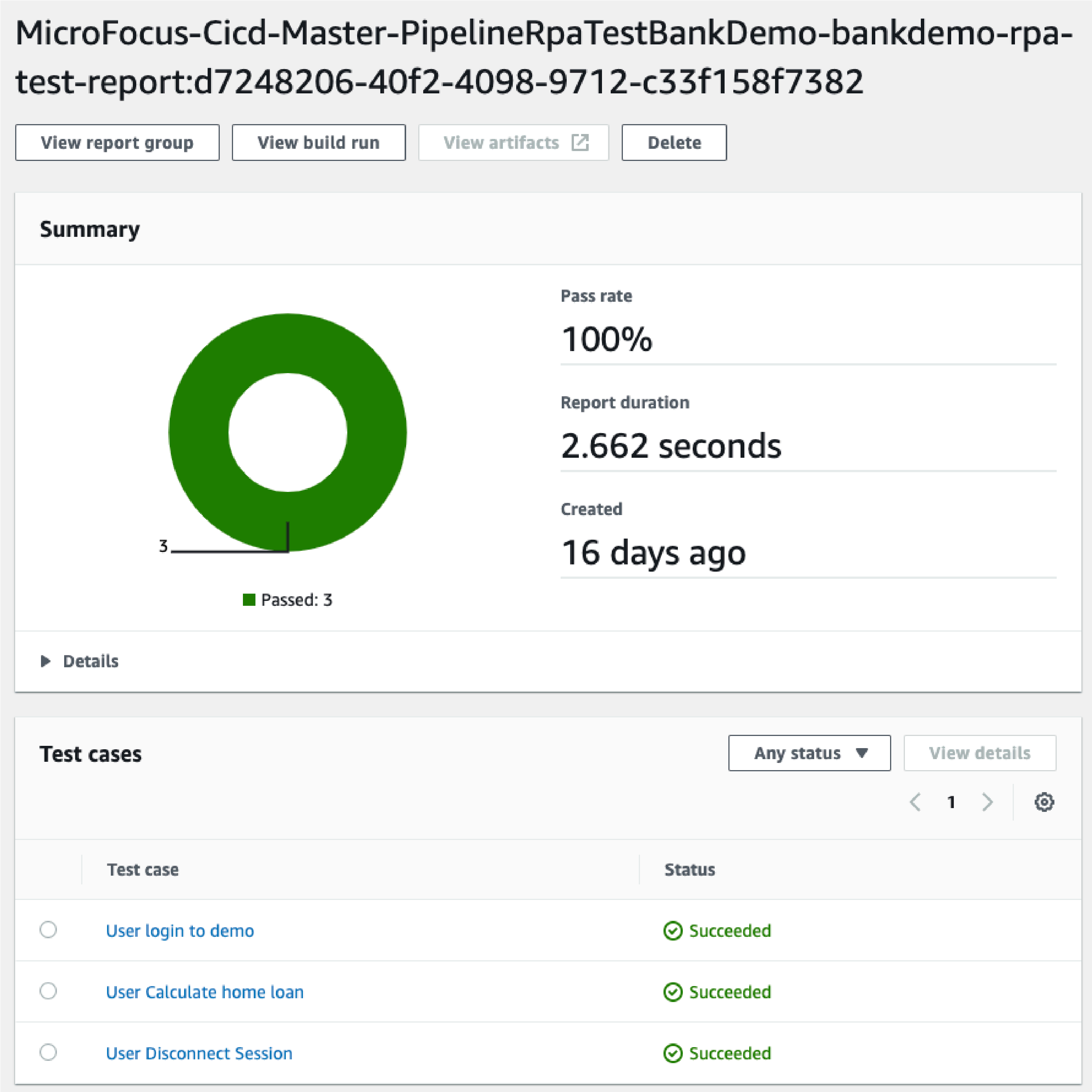
CodePipeline and CodeBuild are used to create the CI/CD pipelines. Because CodeBuild and CodePipeline have no upfront fees or commitments, you pay only for what you use. Compared to mainframe hardware, the AWS solution greatly reduces hardware provisioning lead time and lowers the cost of your testing environment.

In modern development, multiple test methodologies are used. For example, test-driven development (TDD), BDD, and Robot Framework. With this pattern, developers can use these modern tools for mainframe testing. For example, by using x3270, py3270 and the Behave python test tool, you can define an online application's behavior. You can also use build mainframe 3270 robot framework in these CI/CD pipelines.

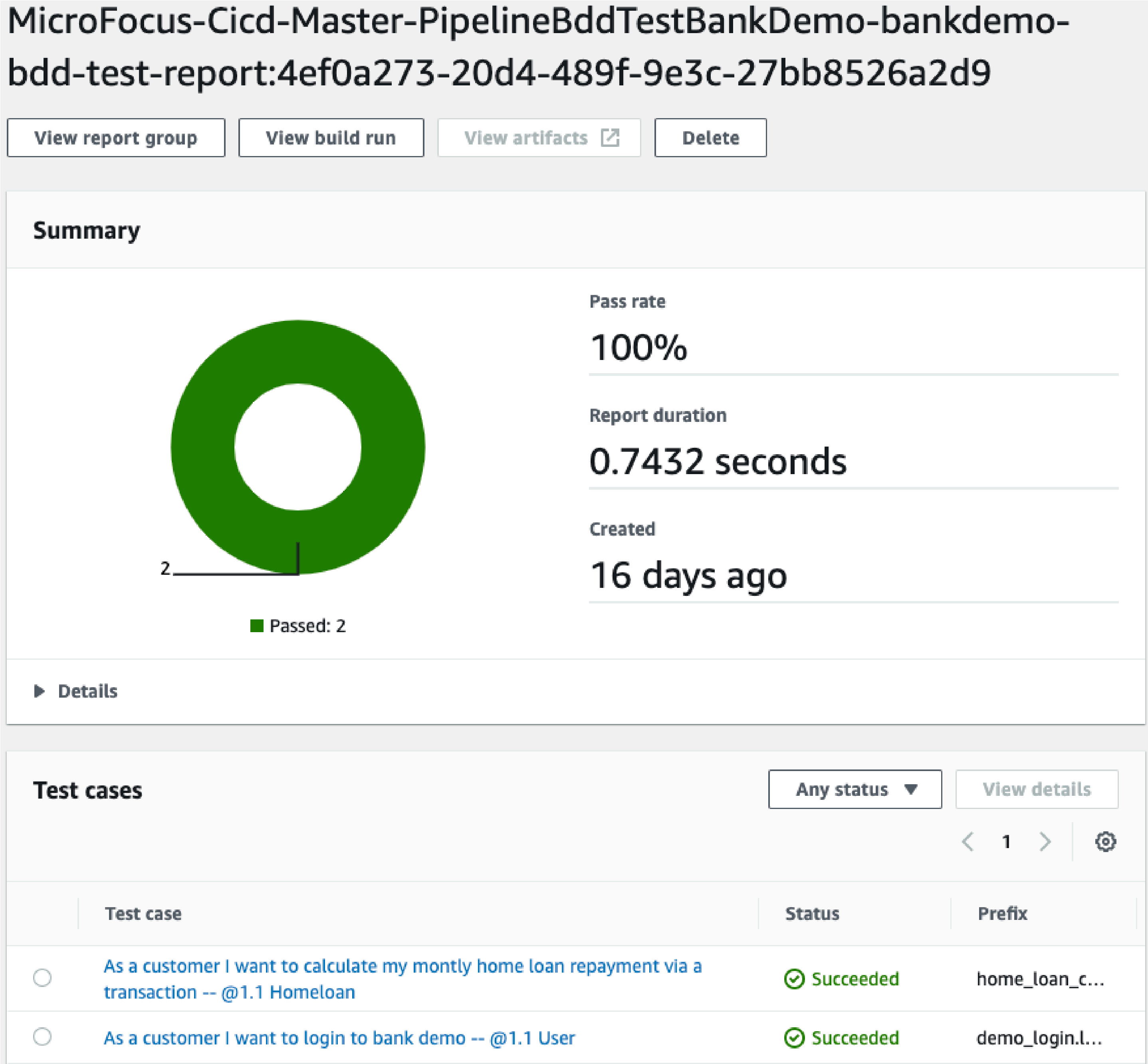
The following diagram shows the team stream CI/CD pipeline.



The following diagram shows the project CI/CD test report produced by CodePipeline in Mainframe3270 Robot Framework.



The following diagram shows the project CI/CD test report produced by CodePipeline in Py3270 and Behave BDD.



After project level tests are passed successfully, the tested code is manually promoted to the integration stream in AccuRev SCM. You can automate this step after the teams have a confidence on the tests coverage of their project pipeline.

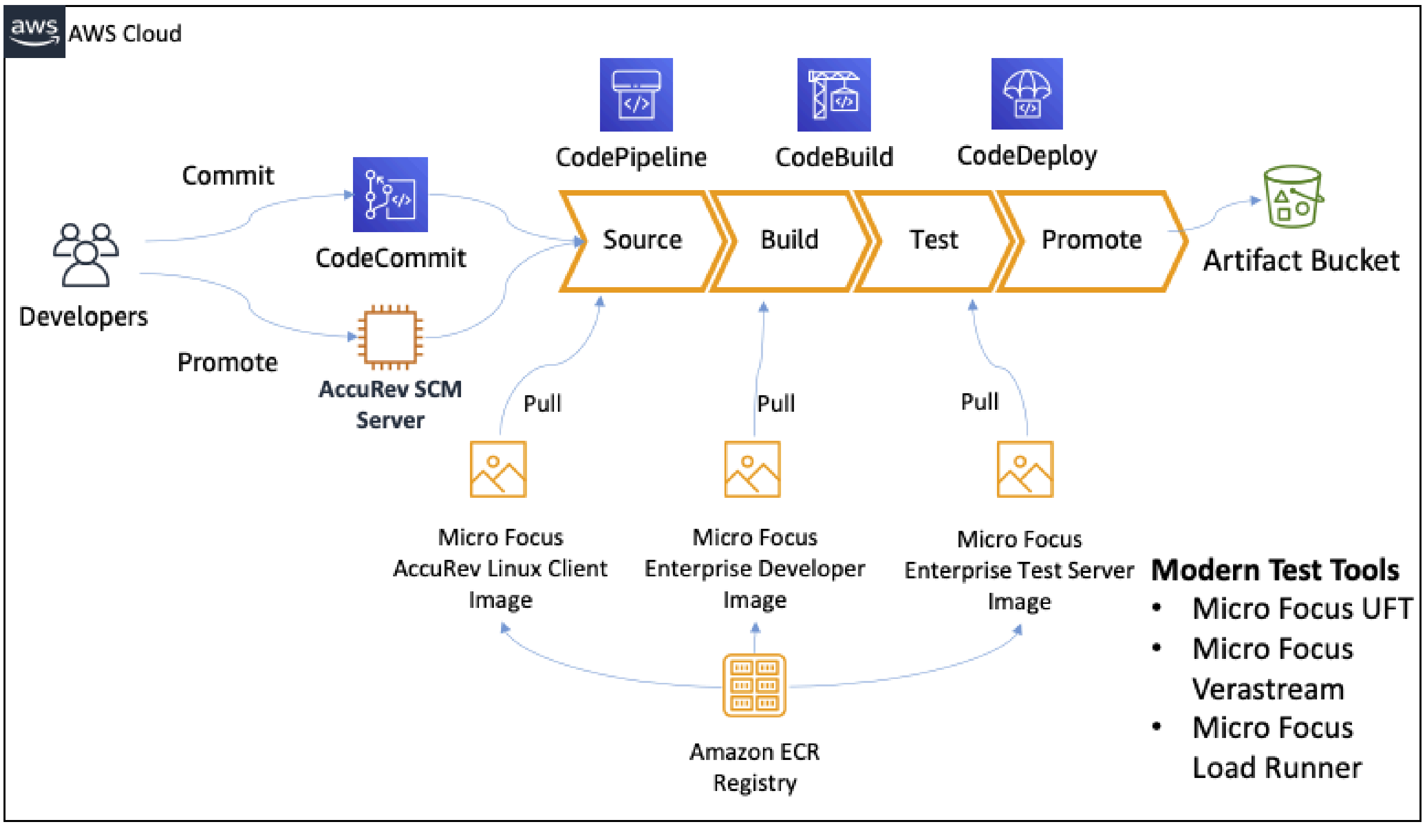
When code is promoted, the system integration CI/CD pipeline checks out the merged code and performs regression tests. The merged code is promoted from all parallel project streams.

Depending on how fine grain the test environment is required, customers can have more system integration CI/CD pipelines in a different environment, for example UAT, Pre-Production.

In the pattern, the tools used in the system integration pipeline are Enterprise Test Server, UFT Server, and Verastream. All these tools can be deployed into the Docker container and used with CodeBuild.

After successfully testing of the mainframe programs, the artifact is stored, with version control, in an S3 bucket.

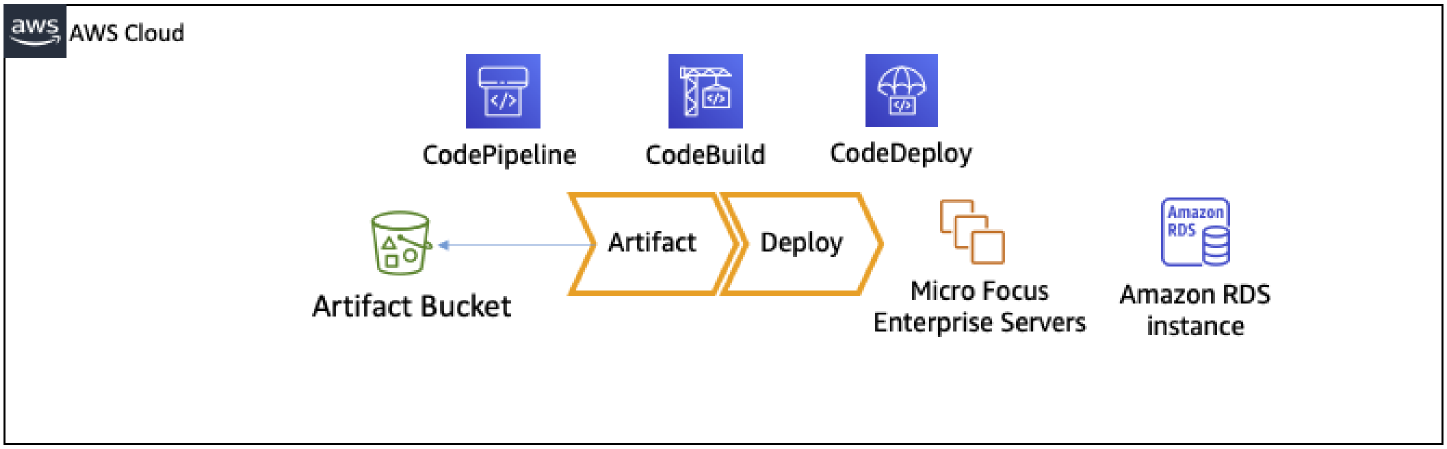
The following diagram shows a system integration CI/CD pipeline.



After the artifact has been successfully tested in the system integration CI/CD pipelines, it can be promoted for production deployment.

If you need to deploy source code back to the mainframe, Rocket Software offers the Enterprise Sync solution to synchronize source code from AccuRev back to Mainframe Endeavour.

The following diagram shows a production CI/CD pipeline deploying the artifact into Enterprise Servers. In this example, CodeDeploy orchestrates the deployment of the tested mainframe artifact into Enterprise Server.



In addition to the architecture walkthrough of the CI/CD pipeline, see the AWS DevOps blog post Automate thousands of mainframe tests on AWS with the Micro Focus Enterprise Suite for more information on testing mainframe applications in CodeBuild and CodePipeline. (Micro Focus is now Rocket Software.) Refer to the blog post for the best practices and details of doing mainframe tests on AWS.

Tools

AWS automation tools

- AWS CloudFormation
- Amazon CloudWatch Events
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline
- Amazon ECR
- Amazon S3
- AWS Secrets Manager
- AWS Systems Manager

Rocket Software tools

- Rocket Enterprise Developer for Eclipse

- Rocket Enterprise Test Server
- Rocket Enterprise Server (production deployment)
- Rocket Software AccuRev
- Rocket Software Enterprise Suite License Manager
- Rocket Software Verastream Host Integrator
- Rocket Software UFT One

Other tools

- x3270
- py3270
- Robot-Framework-Mainframe-3270-Library

Epics

Create the AccuRev SCM infrastructure

Task	Description	Skills required
Deploy a primary AccuRev SCM server by using CloudFormation.		AWS CloudFormation
Create the AccuRev Administr ator user.	Log in to AccuRev SCM Server, and run the CLI command to create an Administrator user.	AccuRev SCM Server Administrator
Create AccuRev streams.	Create AccuRev streams that inherit from upper streams in sequence: Production, System Integration, Team streams.	AccuRev SCM Administrator
Create the developer AccuRev login accounts.	Use AccuRev SCM CLI commands to create AccuRev users login accounts for mainframe developers.	AccuRev SCM Administrator

Create the Enterprise Developer Amazon EC2 launch template

Task	Description	Skills required
Deploy the Amazon EC2 launch template by using CloudFormation.	Use CloudFormation to deploy an Amazon EC2 launch template for Enterprise Developer instances. The template includes a Systems Manager Automation document for the Rocket Enterprise Developer instance.	AWS CloudFormation
Create the Enterprise Developer instance from the Amazon EC2 template.		AWS Console Login and Mainframe Developer Skills

Create the Enterprise Developer tool Docker image

Task	Description	Skills required
Create the Enterprise Developer tool Docker image.	Use the Docker command and the Enterprise Developer tool Dockerfile to create the Docker image.	Docker
Create the Docker repository in Amazon ECR.	On the Amazon ECR console, create the repository for the Enterprise Developer Docker image.	Amazon ECR
Push the Enterprise Developer tool Docker image to Amazon ECR.	Run the Docker push command to push the Enterprise Developer tool Docker image to save it in the	Docker

Task	Description	Skills required
	Docker repository in Amazon ECR.	

Create the Enterprise Test Server Docker image

Task	Description	Skills required
Create the Enterprise Test Server Docker image.	Use the Docker command and the Enterprise Test Server Dockerfile to create the Docker image.	Docker
Create the Docker repository in Amazon ECR.	On the Amazon ECR console, create the Amazon ECR repository for the Enterprise Test Server Docker image.	Amazon ECR
Push the Enterprise Test Server Docker image to Amazon ECR.	Run the Docker push command to push and save the Enterprise Test Server Docker image in Amazon ECR.	Docker

Create the team stream CI/CD pipeline

Task	Description	Skills required
Create the CodeCommit repository.	On the CodeCommit console, create a Git-based repositor y for infrastructure and CloudFormation code.	AWS CodeCommit
Upload the CloudFormation template and the automatio	Run the Git push command to upload CloudFormation	Git

Task	Description	Skills required
n code into the CodeCommit repository.	template and automation code into the repository.	
Deploy the team stream CI/CD pipeline by using CloudFormation.	Use the prepared CloudFormation template to deploy a team stream CI/CD pipeline.	AWS CloudFormation

Create the system integration CI/CD pipeline

Task	Description	Skills required
Create the UFT Docker image.	Use the Docker command and the UFT Dockerfile to create the Docker image.	Docker
Create the Docker repository in Amazon ECR for the UFT image.	On the Amazon ECR console, create the Docker repository for the UFT image.	Amazon ECR
Push the UFT Docker image to Amazon ECR.	Run the Docker push command to push and save the Enterprise Test Server Docker image in Amazon ECR.	Docker
Create the Verastream Docker image.	Use the Docker command and the Verastream Dockerfile to create the Docker image.	Docker
Create the Docker repository in Amazon ECR for the Verastream image.	On the Amazon ECR console, create the Docker repository for the Verastream image.	Amazon ECR
Deploy the system integration CI/CD pipeline by using CloudFormation.	Use the prepared CloudFormation template to deploy a system integration CI/CD pipeline.	AWS CloudFormation

Create production deployment CI/CD pipeline

Task	Description	Skills required
Deploy Enterprise Server by using the AWS Quick Start.	To deploy Enterprise Server by using CloudFormation, launch the Enterprise Server on AWS Quick Start.	AWS CloudFormation
Deploy a production deployment CI/CD pipeline.	On the CloudFormation console, use the CloudFormation template to deploy a production deployment CI/CD pipeline.	AWS CloudFormation

Related resources

References

- AWS DevOps Blog - Automate thousands of mainframe tests on AWS with the Micro Focus Enterprise Suite (Micro Focus is now Rocket Software.)
- py3270/py3270 GitHub repository
- Altran-PT-GDC/Robot-Framework-Mainframe-3270-Library GitHub repository
- Welcome to behave!
- APN Partner Blog - Tag: Micro Focus (Micro Focus is now Rocket Software.)
- Launching an instance from a launch template

AWS Marketplace

- Rocket Software UFT One

AWS Quick Start

- Rocket Enterprise Server on AWS

Modernize mainframe online printing workloads on AWS by using Micro Focus Enterprise Server and LRS VPSX/MFI

Shubham Roy and Kevin Yung, Amazon Web Services

Abraham Rondon, Micro Focus

Guy Tucker, Levi, Ray and Shoup Inc

Summary

This pattern shows you how to modernize your business-critical mainframe online printing workloads on the Amazon Web Services (AWS) Cloud by using Micro Focus Enterprise Server as a runtime for a modernized mainframe application and LRS VPSX/MFI (Micro Focus Interface) as a print server. The pattern is based on the replatform mainframe modernization approach. In this approach, you migrate your mainframe online application to Amazon Elastic Compute Cloud (Amazon EC2) and migrate your mainframe database, such as IBM DB2 for z/OS, to Amazon Relational Database Service (Amazon RDS). The authentication and authorization for the modernized print workflow is performed by AWS Directory Service for Microsoft Active Directory, also known as AWS Managed Microsoft AD. The LRS Directory Information Server (LRS/DIS) is integrated with AWS Managed Microsoft AD for print workflow authentication and authorization. By modernizing your online printing workloads, you can reduce IT infrastructure costs, mitigate the technical debt of maintaining legacy systems, remove data silos, increase agility and efficiency with a DevOps model, and take advantage of on-demand resources and automation in the AWS Cloud.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A mainframe online printing or output management workload
- Basic knowledge of how to rebuild and deliver a mainframe application that runs on Micro Focus Enterprise Server (For more information, see the Enterprise Server data sheet in the Micro Focus documentation.)
- Basic knowledge of LRS cloud printing solutions and concepts (For more information, see Output Modernization in the LRS documentation.)
- Micro Focus Enterprise Server software and license (For more information, contact Micro Focus sales.)

- LRS VPSX/MFI, LRS/Queue, and LRS/DIS software and licenses (For more information, contact LRS sales.)

> **Note**
>
> For more information about configuration considerations for mainframe online printing workloads, see *Considerations* in the *Additional information* section of this pattern.

Product versions

- Micro Focus Enterprise Server 8.0 or later
- LRS VPSX/MFI V1R3 or later

Architecture

Source technology stack

- Operating system – IBM z/OS
- Programming language – Common Business-Oriented Language (COBOL) and Customer Information Control System (CICS)
- Database – IBM DB2 for z/OS IBM Information Management System (IMS) and Virtual Storage Access Method (VSAM)
- Security – Resource Access Control Facility (RACF), CA Top Secret for z/OS, and Access Control Facility 2 (ACF2)
- Printing and output management – IBM mainframe z/OS printing products (IBM Infoprint Server for z/OS, LRS, and CA View)

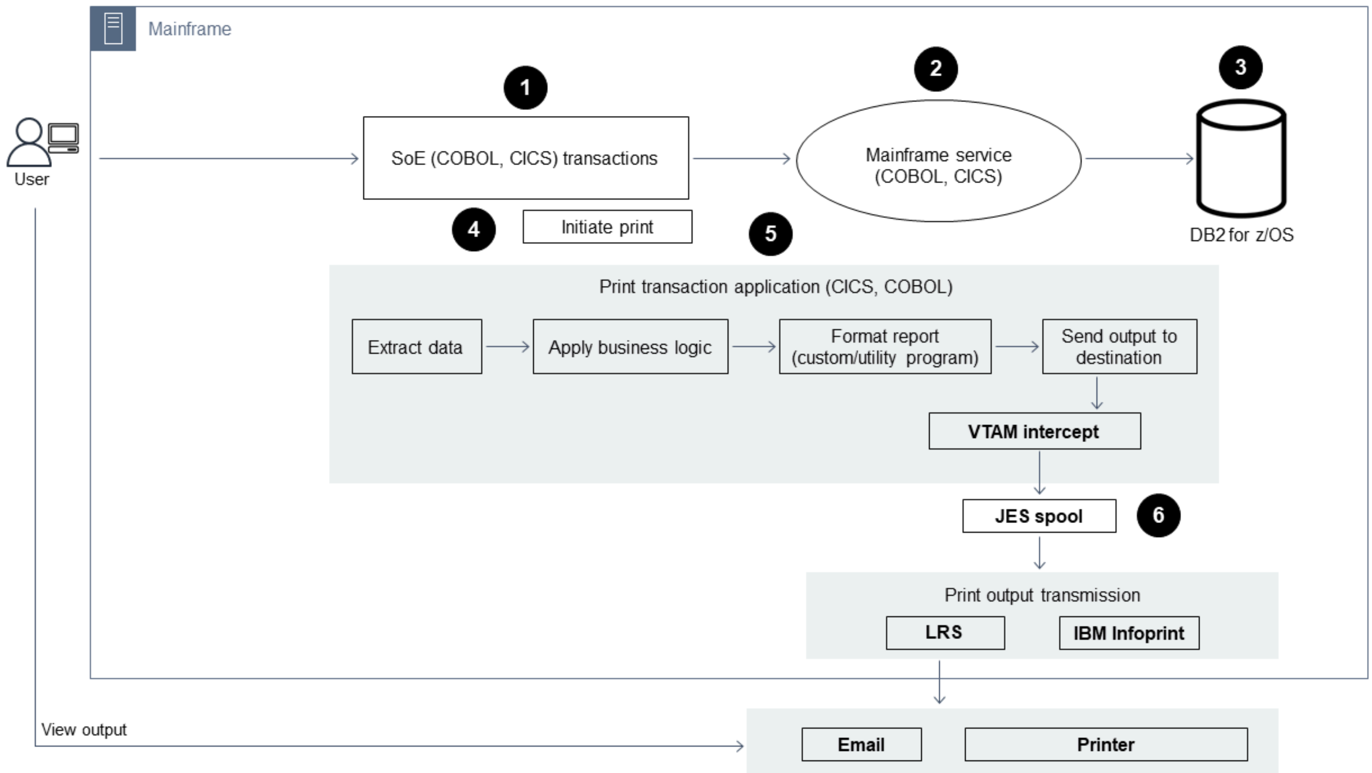
Target technology stack

- Operating system – Microsoft Windows Server running on Amazon EC2
- Compute – Amazon EC2
- Programming language – COBOL and CICS
- Database – Amazon RDS
- Security – AWS Managed Microsoft AD

- Printing and output management – LRS printing solution on AWS
- Mainframe runtime environment – Micro Focus Enterprise Server

Source architecture

The following diagram shows a typical current state architecture for a mainframe online printing workload.



The diagram shows the following workflow:

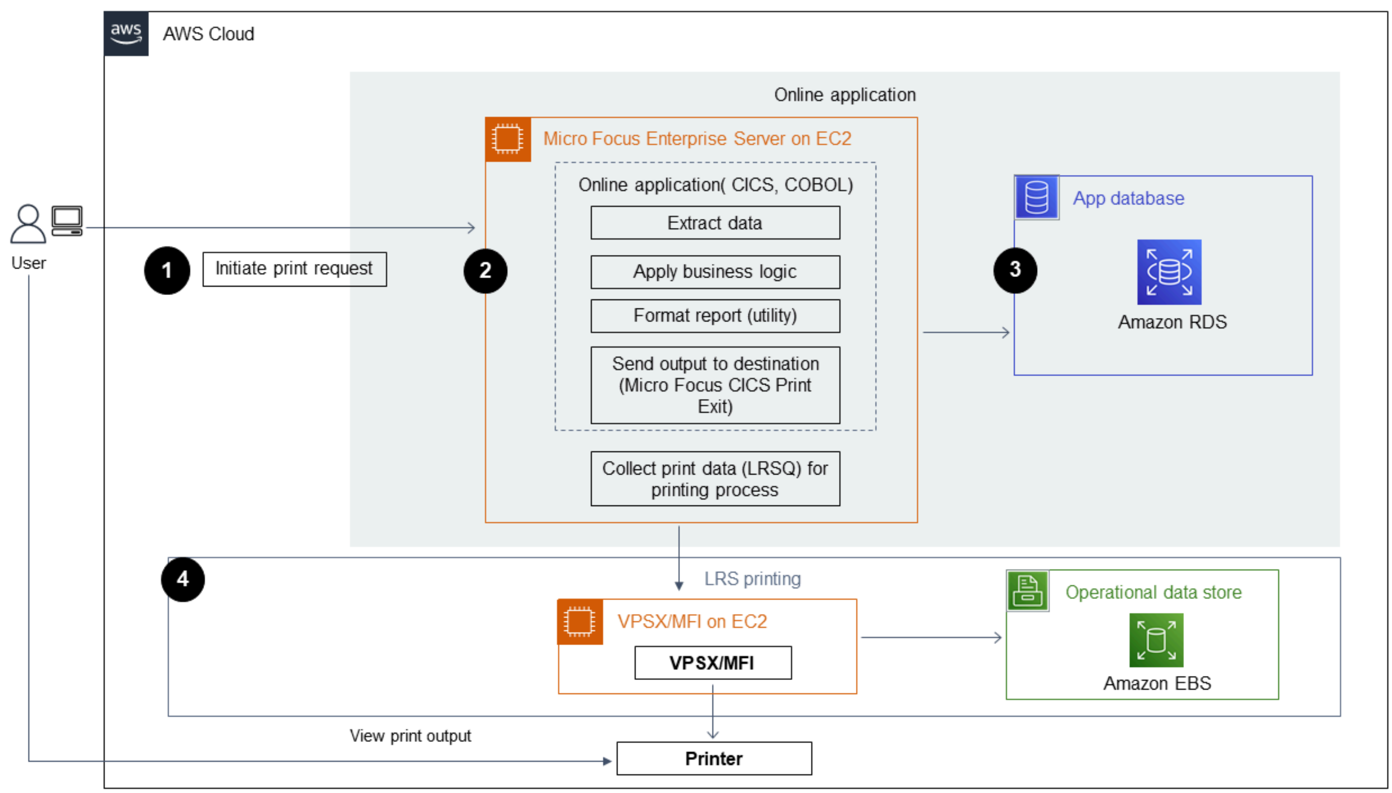
1. Users perform business transactions on a system of engagement (SoE) that's built on an IBM CICS application written in COBOL.
2. The SoE invokes the mainframe service, which records the business transaction data in a system-of-records (SoR) database such as IBM DB2 for z/OS.
3. The SoR persists the business data from the SoE.
4. A user initiates a request to generate print output from the CICS SoE, which initiates a print transaction application to process the print request.
5. The print transaction application (such as a CICS and COBOL program) extracts data from the database, formats the data according to business requirements, and generates business output

(print data) such as billing statements, ID cards, or loan statements. Then, the application sends a print request by using Virtual Telecommunications Access Method (VTAM). A z/OS print server (such as IBM Infoprint Server) uses NetSpool or a similar VTAM component to intercept the print requests, and then creates print output datasets on the JES spool by using JES output parameters. The JES output parameters specify routing information that the print server uses to transmit the output to a particular network printer. The term *VTAM* refers to the z/OS Communications Server and the System Network Architecture (SNA) services element of z/OS.

6. The printing output transmission component transmits the output print datasets from the JES spool to remote printers or print servers, such as LRS (as demonstrated in this pattern), IBM Infoprint Server, or email destinations.

Target architecture

The following diagram shows an architecture for a mainframe online printing workload that's deployed in the AWS Cloud:



The diagram shows the following workflow:

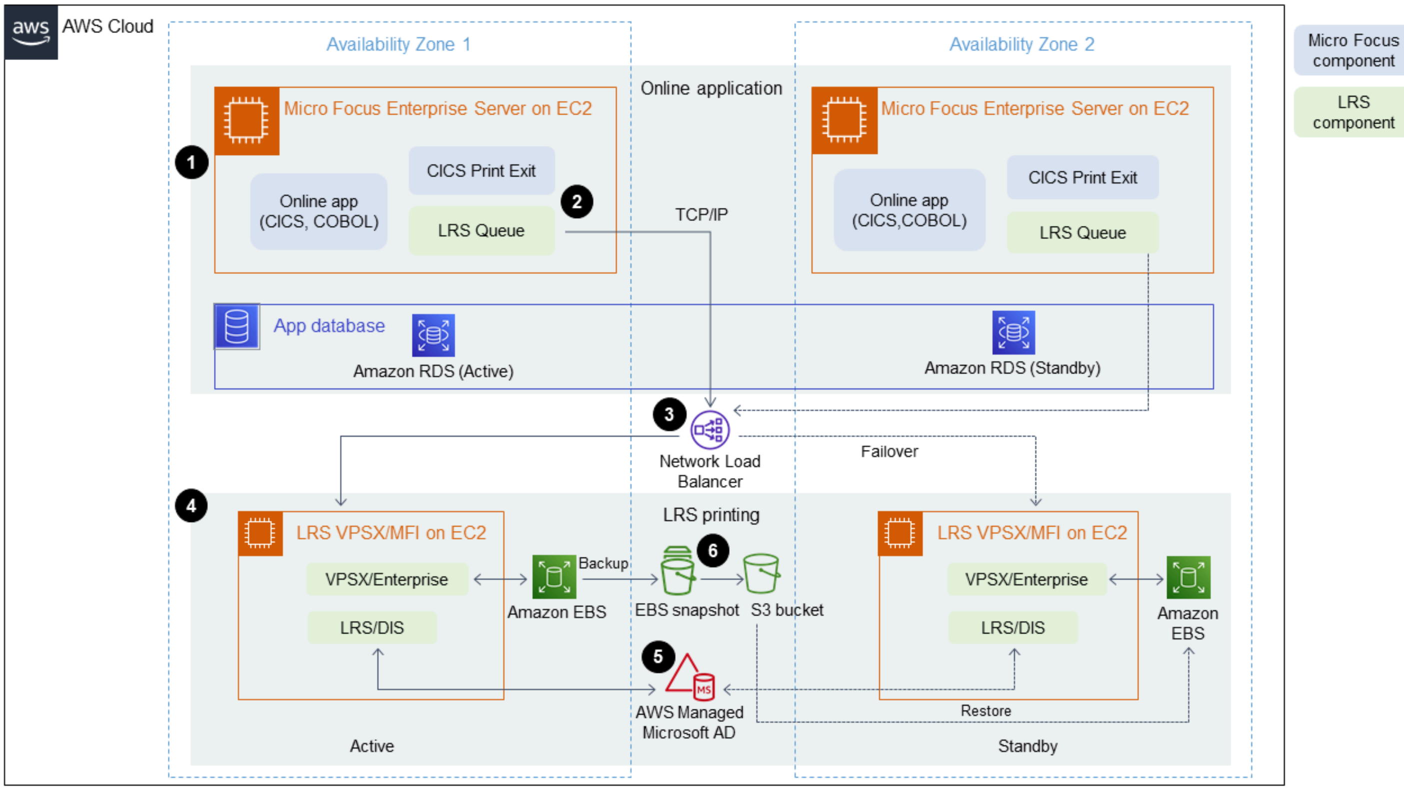
1. A user initiates a print request from an online (CICS) user-interface to create print output, such as billing statements, ID cards, or loan statements.

2. The mainframe online application (replatformed to Amazon EC2) uses the Micro Focus Enterprise Server runtime to extract data from the application database, apply business logic to the data, format the data, and then send the data to a print destination by using Micro Focus CICS Print Exit (DFHUPRNT).
3. The application database (an SoR that runs on Amazon RDS) persists data for print output.
4. The LRS VPSX/MFI printing solution is deployed on Amazon EC2, and its operational data is stored in Amazon Elastic Block Store (Amazon EBS). LRS VPSX/MFI uses a TCP/IP-based LRS/Queue transmission agent to collect print data through the Micro Focus CICS Print Exit API (DFHUPRNT) and deliver the data to a specified printer destination. The original TERMID (TERM) that's used in the modernized CICS application is used as the VPSX/MFI Queue name.

> **Note**
>
> The target solution typically doesn't require application changes to accommodate mainframe formatting languages, such as IBM Advanced Function Presentation (AFP) or Xerox Line Condition Data Stream (LCDS). For more information about using Micro Focus for mainframe application migration and modernization on AWS, see Empowering Enterprise Mainframe Workloads on AWS with Micro Focus in the AWS documentation.

AWS infrastructure architecture

The following diagram shows a highly available and secure AWS infrastructure architecture for a mainframe online printing workload:



The diagram shows the following workflow:

1. The mainframe online application (written on a programming language such as CICS or COBOL) uses core business logic to process and generate print output, such as billing statements, ID cards, and loan statements. The online application is deployed on Amazon EC2 across two Availability Zones (AZ) for high availability (HA) and uses Micro Focus CICS Print Exit to route print output to LRS VPSX/MFI for end-user printing.
2. LRS VPSX/MFI uses a TCP/IP-based LRS/Queue transmission agent to collect or capture print data from the Micro Focus online Print Exit programming interface. Online Print Exit passes the necessary information to enable LRS VPSX/MFI to effectively process the print file and dynamically build LRS/Queue commands.

> **Note**
>
> For more information on various CICS application programing methods for print and how they are supported in Micro Focus Enterprise server and LRS VPSX/MFI, see *Print data capture* in the *Additional information* section of this pattern.

3. > **Note**
   >
   > A Network Load Balancer provides a DNS name to integrate Micro Focus Enterprise Server with LRS VPSX/MFI. : LRS VPSX/MFI supports a Layer 4 load balancer. The Network Load Balancer also does a basic health check on LRS VPSX/MFI and routes traffic to the registered targets that are healthy.

4. The LRS VPSX/MFI print server is deployed on Amazon EC2 across two Availability Zones for HA and uses Amazon EBS as an operational data store. LRS VPSX/MFI supports both the active-active and active-passive service modes. This architecture uses multiple Availability Zones in an active-passive pair as an active and hot standby. The Network Load Balancer performs a health check on LRS VPSX/MFI EC2 instances and routes traffic to hot standby instances in another Availability Zone if an active instance is in an unhealthy state. The print requests are persisted in the LRS Job Queue locally in each of the EC2 instances. In the event of recovery, a failed instance has to be restarted for the LRS services to resume processing the print request.

   > **Note**
   >
   > LRS VPSX/MFI can also perform health checks at the printer fleet level. For more information, see *Printer fleet health checks* in the *Additional information* section of this pattern.

5. AWS Managed Microsoft AD integrates with LRS/DIS to perform print workflow authentication and authorization. For more information, see *Print authentication and authorization* in the *Additional information* section of this pattern.
6. LRS VPSX/MFI uses Amazon EBS for block storage. You can back up Amazon EBS data from active EC2 instances to Amazon S3 as point-in-time snapshots and restore them to hot standby EBS volumes. To automate the creation, retention, and deletion of Amazon EBS volume snapshots, you can use Amazon Data Lifecycle Manager to set the frequency of automated snapshots and restore them based on your RTO/RPO requirements.

Tools

AWS services

- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with Amazon EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- AWS Directory Service for Microsoft Active Directory (AD), also known as AWS Managed Microsoft Active Directory, enables your directory-aware workloads and AWS resources to use managed Active Directory in AWS.

Other tools

- LRS VPSX/MFI (Micro Focus Interface), co-developed by LRS and Micro Focus, captures output from a Micro Focus Enterprise Server JES spool and reliably delivers it to a specified print destination.
- LRS Directory Information Server (LRS/DIS) is used for authentication and authorization during the print workflow.
- LRS/Queue is a TCP/IP-based LRS/Queue transmission agent, used by LRS VPSX/MFI, to collect or capture print data through the Micro Focus online Print Exit programming interface.
- Micro Focus Enterprise Server is an application deployment environment for mainframe applications. It provides the execution environment for mainframe applications that are migrated or created by using any version of Micro Focus Enterprise Developer.

Epics

Set up Micro Focus Enterprise Server on Amazon EC2 and deploy a mainframe online application

Task	Description	Skills required
Set up Micro Focus Enterprise Server and deploy a demo online application.	Set up Micro Focus Enterprise Server on Amazon EC2, and then deploy the Micro Focus Account Demo application	Cloud architect

Task	Description	Skills required
	(ACCT Demo) on Amazon EC2 by following the instructions from Tutorial: CICS Support in the Micro Focus documenta tion. The ACCT Demo application is a mainframe online (CICS) application that creates and then initiates print output.	

Set up an LRS print server on Amazon EC2

Task	Description	Skills required
Get an LRS product license for printing.	To get an LRS product license for LRS VPSX/MFI, LRS/Queue, and LRS/DIS, contact the LRS Output Managemen t team. You must provide the host names of the EC2 instances where the LRS products will be installed.	Build lead
Create an Amazon EC2 Windows instance to install LRS VPSX/MFI.	Launch an Amazon EC2 Windows instance by following the instructions from Step 1: Launch an instance in the Amazon EC2 documentation. Your instance must meet the following hardware and software requirements for LRS VPSX/MFI:	Cloud architect

Task	Description	Skills required
	• CPU – Dual Core • RAM – 16 GB • Drive – 500 GB • Minimum EC2 instance – m5.xlarge • OS – Windows/Linux • Software – Internet Information Service (IIS) or Apache **Note** The preceding hardware and software requirements are intended for a small printer fleet (around 500–1000). To get the full requirements, consult with your LRS and AWS contacts. When you create your Windows instance, do the following: 1. Confirm that the EC2 host name is the same host name used for the LRS product license.	

Task	Description	Skills required
	2. Enable CGI in Amazon EC2 by completing the following: a. Connect to your EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. b. In the Windows Start menu, find and open Server Manager. c. In Server Manager, choose **Dashboard**, **Quick Start**, **Add roles and features**. Then, choose **Server roles**. d. In **Server roles**, choose **WebServer (IIS)**, and then choose **Application Development**. e. In **Application Development**, select the **CGI** check box. f. Follow the instructions on the Windows Server Manager **Add roles and features** wizard to install CGI. g. Open port 5500 in the Windows firewall of the EC2 instance for LRS/ Queue communication.	

Task	Description	Skills required
Install LRS VPSX/MFI on the EC2 instance.	1. Connect to your EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. **Note** Open the link to the product download page from the LRS email that you should receive. : LRS products are distributed by electronic file transfer (EFT). 3. Download LRS VPSX/MFI and unzip the file (default folder: `c:\LRS`). 4. Launch the LRS Product Installer from the unzipped folder to install LRS VPSX/MFI. 5. In the **Select Features** menu, select **VPSX® Server (V1R3.022)**, and then choose **Next** to start the installation process. You will receive a success message when installation is complete.	Cloud architect

Task	Description	Skills required
Install LRS/Queue.	1. Connect to your Micro Focus Enterprise Server EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. Open the link to the LRS product download page from the LRS email that you should receive, download LRS/Queue, and then unzip the file. 3. Go to the location where you downloaded the files, and then launch the LRS product installer to install LRS/Queue.	Cloud architect

Task	Description	Skills required
Install LRS/DIS.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructi ons from [Step 2: Connect to your instance](#) in the Amazon EC2 documenta tion. 2. Open the link to the LRS product download page from the LRS email that you should receive, download LRS/DIS, and then unzip the file. 3. Go to the location where you downloaded the files, and then launch the LRS Product Installer. 4. In the LRS Product Installer , expand **LRS Misc Tools**, select **LRS DIS**, and then choose **Next**. 5. Follow the rest of the instructions in the LRS Product Installer to complete the installation process.	Cloud architect

Task	Description	Skills required
Create a target group and register LRS VPSX/MFI EC2 as the target.	Create a target group by following the instructions from [Create a target group for your Network Load Balancer](#) in the Elastic Load Balancing documentation. When you create the target group, do the following: 1. On the **Specify group details** page, for **Choose a Target Type**, choose **Instances**. 2. For **Protocol**, choose **TCP**. 3. For **Port**, choose **5500**. 4. On the **Register targets** page, in the **Available instances** section, select the LRS VPSX/MFI EC2 instances.	Cloud architect

Task	Description	Skills required
Create a Network Load Balancer.	Follow the instructions from [Create a Network Load Balancer](#) in the Elastic Load Balancing documentation. Your Network Load Balancer routes traffic from Micro Focus Enterprise Server to LRS VPSX/MFI EC2. When you create the Network Load Balancer, do the following on the **Listeners and Routing** page: 1. For **Protocol**, choose **TCP**. 2. For **Port**, choose **5500**. 3. For **Default action**, choose **Forward to** for the target group that you created earlier.	Cloud architect

Integrate Micro Focus Enterprise Server with LRS VPSX/MFI and LRS/Queue

Task	Description	Skills required
Configure Micro Focus Enterprise Server for LRS/Queue integration.	1. Connect to your Micro Focus Enterprise Server EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. In the Windows Start menu, open the Micro	Cloud architect

Task	Description	Skills required
	Focus Enterprise Server Administration UI. 3. In the menu bar, choose **NATIVE**. 4. In the navigation pane, choose **Directory Server**, and then choose **BANKDEMO** or your Enterprise server region. 5. From **General** in the left navigation pane, scroll down to the **Additional** section to configure the environment variables (LRSQ_ADDRESS, LRSQ_PORT, LRSQ_COMMAND) to point to LRSQ. 6. For **LRSQ_ADDRESS**, enter the IP address or DNS name of the Network Load Balancer that you created earlier. 7. For **LRSQ_PORT**, enter **VPSX LRSQ Listener Port (5500)**. 8. For **LRSQ_COMMAND**, enter the path location of the LRSQ executable. 9. **Note** LRS currently supports a maximum	

Task	Description	Skills required
	character limit of 50 for DNS names, but this is subject to change in the future. If your DNS name is greater than 50, then you can use the IP address of the Network Load Balancer as an alternative.	

Task	Description	Skills required
Make CICS Print Exit (DFHUPRNT) available to Micro Focus Enterprise Server initialization.	1. Connect to your Micro Focus Enterprise Server EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. **Note** Copy CICS Print Exit (DFHUPRNT) from the LRS VPSX/MFI executable folder (named `VPSX_MFI_R2`) to the Micro Focus Enterprise Server EC2 instance location. For 32 bit systems, the location is `C:\Program Files (x86)\Micro Focus\Enterprise Server\bin` . For 64 bit systems, the location is `C:\Program Files (x86)\Micro Focus\Enterprise Server\bin64` .: The DFHUPRNT_	Cloud architect

Task	Description	Skills required
	`64.dll` file must be renamed to `DFHUPRNT.dll` when copied. **Validate that Micro Focus Enterprise Server has detected CICS Print Exit (DFHUPRNT)** 1. Stop and start Micro Focus Enterprise Server. 2. In the Administration panel of Micro Focus Enterpris e Server, open **Monitor**, **Logs**, **Console logs**. 3. Check the console logs for the following message: "3270 printer user exit DFHUPRNT installed successfully."	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Define the CICS printer's terminal ID (TERMIDs) as Micro Focus Enterprise Server.</td><td>Enable 3270 printing in Micro Focus Enterprise Server

1. In the Administration panel of Micro Focus Enterpris e Server, open CICS, Resources, By Group.
2. From the left navigation panel, choose SIT (System Initialization Table), and then choose BNKCICV.
3. In the General section, scroll down to 3270, and then select the 3270 Print check box.

Define the CICS printer's terminal in Micro Focus Enterprise Server

1. In the Administration panel of Micro Focus Enterpris e Server, open CICS, Resources, By Type.
2. From the left navigation panel, choose Term, and then choose New. The Create Terminal Resource form opens.
3. For Name, enter the name of the LRS Print Queue. (Note: This pattern uses "P275" as the CICS</td><td>Cloud architect</td></tr>
</table>

Task	Description	Skills required
	printer's terminal ID and LRS VPSX Print Queue.) 4. For **Group**, enter **BANKTERM**. 5. For **Auto Install – Model**, enter **NO**. 6. For **Terminal Identifiers - Terminal type**, enter **DFHPRT32**. 7. For **Net name**, enter **VTAMP275**. 8. For **Terminal Usage**, select the **In Service** check box. 9. Scroll at the top of the page, and then chose **Save**. 10. Choose **Install**. A pop-up message displays a successful installation message.	

Set up printers and print users in Micro Focus Enterprise Server and LRS VPSX/MFI

Task	Description	Skills required
Create a print queue in the LRS VPSX.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation.	Cloud architect

Task	Description	Skills required
	2. Open the VPSX Web Interface from the Windows Start menu. 3. In the navigation pane, choose **Printers**. 4. Choose **Add**, and then choose **Add Printer**. 5. On the **Printer Configuration** page, for **Printer Name**, enter **P275**. 6. For **VPSX ID**, enter **VPS1**. 7. For **CommType**, select **TCPIP/LRSQ**. 8. For **Host/IP Address**, enter the IP address of the physical printer that you want to add. 9. For **Device**, enter the name of your device. 10. Choose either **Windows Driver** or **Linux/Mac Driver**. 11. Choose **Add**. **Note** The print queue must be equivalent to the Print TERMIDs created in Micro Focus Enterprise Server.	

Task	Description	Skills required
Create a print user in LRS VPSX/MFI.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructions from [Step 2: Connect to your instance](#) in the Amazon EC2 documentation. 2. Open the VPSX Web Interface from the Windows Start menu. 3. In the navigation pane, choose **Security**, and then choose **Users**. 4. In the **User Name** column, choose **admin**, and then choose **Copy**. 5. In the **User Profile Maintenance** window, for **User Name**, enter a user name (for example, **PrintUser**). 6. For **Description**, enter a brief description (for example, **User for test print**). 7. Choose **Update**. This creates a print user (for example, **PrintUser**). 8. In the navigation pane, under **User**, choose the new user that you created. 9. From the **Command** menu, choose **Security**.	Cloud architect

Task	Description	Skills required
	10. On the **Security Rules** page, choose all the applicable printer security and job security options, and then choose **Save**. 11. To add your new print user to the **Administrator** group, go to the navigation pane, choose **Security**, and then choose **Configure**. 12. In the **Security configura tion** window, add your new print user to the **Administr ator** column.	

Set up print authentication and authorization

Task	Description	Skills required
Create an AWS Managed Microsoft AD domain with users and groups.	1. Create an Active Directory on AWS Managed Microsoft AD by following the instructions from [Create your AWS Managed Microsoft AD directory](#) in the AWS Directory Service documentation. 2. Deploy an EC2 instance (Active Directory manager) and install Active Directory tools to manage your AWS Managed Microsoft AD by following the instructions from [Step 3: Deploy an EC2](#)	Cloud architect

Task	Description	Skills required
	instance to manage your AWS Managed Microsoft AD in the AWS Directory Service documentation. 3. **Note** Connect to your EC2 instance by following the instructions from Step 2: Connect to your instance in the Amazon EC2 documenta tion. : When you connect to the EC2 instance, enter your administrator credentials (for the directory that you created in step one) in the Windows Security window. 4. In the Windows Start menu, under **Windows Administrative Tools**, choose **Active Directory Users and Computers**. 5. Create a print user in the Active Directory domain by following the steps from Create a user in the	

Task	Description	Skills required
	AWS Directory service documentation.	
Join LRS VPSX/MFI EC2 to an AWS Managed Microsoft AD domain.	Join LRS VPSX/MFI EC2 to your AWS Managed Microsoft AD domain automatically (AWS Knowledge Center documentation) or manually (AWS Directory Service documentation).	Cloud architect

Task	Description	Skills required
Configure and integrate LRS/DIS with AWS Managed Microsoft AD.	1. Connect to your LRS VPSX/MFI EC2 instance by following the instructions from [Step 2: Connect to your instance] in the Amazon EC2 documentation. 2. In the Windows Start menu, open the **VPSX Web Interface**. 3. In the navigation pane, choose **Security**, and then choose **Configure**. 4. On the **Security Configuration** page, in the **Security Parameters** section, for **Security Type**, select **Internal**. 5. Enter your preferences for the rest of the options in the **Security Parameters** section. 6. Open the **LRS Output Management** folder from the Microsoft Windows Start menu, choose **Server Start**, and then choose **Server Stop**. 7. Log in to LRS VPSX/MFI with your Active Directory user name and password.	Cloud architect

Test an online print workflow

Task	Description	Skills required
Initiate an online print request from the Micro Focus ACCT Demo app.	1. **Note** Open the TN3270 terminal emulator in your Micro Focus Enterprise Server EC2 instance. (: This pattern uses 3270 terminal emulators .) 2. Connect to the TN3270 terminal emulator (Rumba). For **Host name address**, use **127.0.0.1**. For **Telnet Port**, use **9270**. 3. After connecting to the 3270 screen, press CTL +SHIFT+Z to clear the screen. 4. To start the ACCT Demo application, in clear screen, enter **ACCT**. This opens the ACCT Demo online (CICS) application main screen. **Note**: The main screen includes menu options such as **Account file**, **To search by name**, **enter**, **Request type**, **Account**, and **Printer**.	Cloud architect

Task	Description	Skills required
	5. To submit a print request from the ACCT Demo online (CICS) application, enter **P** in the **request type** field, **11111** in the **account** field, and **P275** in the **printer** field. Be sure to set the value in the **printer** field to the value of the CICS printer's terminal ID. 6. Press **Enter**. The "Print Request Scheduled " message appears at the bottom of the screen. This confirms that an online print request was generated from the ACCT Demo application and sent to LRS VPS/MFI for print processing.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Check the print output in LRS VPSX/MFI.</td><td>Connect to your LRS VPSX/MFI EC2 instance by following the instructions from Step 2: Connect to your instance in the Amazon EC2 documentation.
In the Windows Start menu, open the VPSX Web Interface.
In the navigation pane, choose Printers, and then choose Output Queue. Find the P275 print queue that you created for online printing earlier.
For the print queue (P275), in the Spool ID column, choose the spool ID for the request in the printer queue.
On the Actions tab, in the COMMAND column, choose Browse.
You can now see the print output of an account statement with columns for Account No., SURNAME, FIRST, ADDRESS, TELEPHONE, No. Cards Issued, Date issued, Amount, and Balance.</td><td>Test engineer</td></tr>
</table>

Task	Description	Skills required
	For an example, see the **online_print_output** attachment for this pattern.	

Related resources

- LRS Output Modernization (LRS documentation)
- VTAM networking concepts (IBM documentation)
- Summary of logical unit (LU) types (IBM documentation)
- ANSI and machine carriage controls (IBM documentation)
- Empowering Enterprise Mainframe Workloads on AWS with Micro Focus (AWS Partner Network Blog)
- Build a Micro Focus Enterprise Server PAC with Amazon EC2 Auto Scaling and Systems Manager (AWS Prescriptive Guidance documentation)
- Advanced Function Presentation (AFP) data stream (IBM documentation)
- Line Conditioned Data Stream (LCDS) (Compart documentation)

Additional information

Considerations

During your modernization journey, you may consider a wide variety of configurations for mainframe online processes and the output they generate. The mainframe platform has been customized by every customer and vendor that uses it with particular requirements that directly affect print. For example, your current platform may incorporate the IBM Advanced Function Presentation (AFP) or the Xerox Line Condition Data Stream (LCDS) into the current workflow. Additionally, mainframe carriage control characters and channel command words can affect the look of the printed page and may need special handling. As part of the modernization planning process, we recommend that you assess and understand the configurations in your specific print environment.

Print data capture

This section summarizes the CICS application programming methods that you can use in an IBM mainframe environment for printing. LRS VPSX/MFI components provide techniques to allow the same application programs to create data in the same way. The following table describes how each application programming method is supported in a modernized CICS application running in AWS and Micro Focus Enterprise Server with an LRS VPSX/MFI print server.

Method	Description	Support for the method in a modernized environment
EXEC CICS SEND TEXT.. or EXEC CICS SEND MAP..	These CICS and VTAM methods are responsible for creating and delivering 3270/SCS print data streams to LUTYPE0, LUTYPE1, and LUTYPE3 print devices.	A Micro Focus online Print Exit (DFHUPRNT) applicati on program interface (API) enables print data to be processed by VPSX/MFI when 3270/SCS print data streams are created by using either of these methods.
EXEC CICS SEND TEXT.. or EXEC CICS SEND MAP.. (with third-party IBM mainframe software)	The CICS and VTAM methods are responsible for creating and delivering 3270/SCS print data streams to LUTYPE0, LUTYPE1, and LUTYPE3 print devices. Third-party software products intercept the print data, convert the data to standard print format data with an ASA/MCH control character, and place the data on the JES spool to be processed by mainframe-based printing systems that use JES.	A Micro Focus online Print Exit (DFHUPRNT) API enables print data to be processed by VPSX/MFI when 3270/SCS print data streams are created by using either of these methods.
EXEC CICS SPOOLOPEN	This method is used by CICS application programs to write data directly to the JES	Micro Focus Enterprise Server spools the data to the Enterprise Server spool where

	spool. The data then becomes available to be processed by mainframe-based printing systems that use JES.	it can be processed by the VPSX/MFI Batch Print Exit (LRSPRTE6) that spools the data to VPSX.
DRS/API	An LRS-supplied programma tic interface is used for writing print data to JES.	VPSX/MFI supplies a replacement interface that spools the print data directly to VPSX.

Printer fleet health checks

LRS VPSX/MFI (LRS LoadX) can perform deep dive health checks, including device management and operational optimization. Device management can detect failure in a printer device and route the print request to a healthy printer. For more information about deep dive health checks for printer fleets, see the LRS documentation that's included with your product license.

Print authentication and authorization

LRS/DIS enables LRS applications to authenticate user IDs and passwords by using Microsoft Active Directory or an LDAP server. In addition to basic print authorization, LRS/DIS can also apply granular-level print security controls in the following use cases:

- Manage who can browse the printer job.
- Manage the browsing level of other user's jobs.
- Manage operational tasks. For example, command-level security such as hold/release, purge, modify, copy, and reroute. Security can be set up by either the User-ID or Group (similar to AD group or LDAP group).

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Move mainframe files directly to Amazon S3 using Transfer Family

Luis Gustavo Dantas, Amazon Web Services

Summary

As part of the modernization journey, you can face the challenge of transferring files between your on-premises servers and the Amazon Web Services (AWS) Cloud. Transferring data from mainframes can be a significant challenge because mainframes typically can't access modern data stores like Amazon Simple Storage Service (Amazon S3), Amazon Elastic Block Store (Amazon EBS), or Amazon Elastic File System (Amazon EFS).

Many customers use intermediate staging resources, such as on-premises Linux, Unix, or Windows servers, to transfer files to the AWS Cloud. You can avoid this indirect method by using AWS Transfer Family with the Secure Shell (SSH) File Transfer Protocol (SFTP) to upload mainframe files directly to Amazon S3.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC) with a subnet that's reachable by your legacy platform
- A Transfer Family endpoint for your VPC
- Mainframe Virtual Storage Access Method (VSAM) files converted to sequential, fixed-length files (IBM documentation)

Limitations

- SFTP transfers files in binary mode by default, which means that files are uploaded to Amazon S3 with EBCDIC encoding preserved. If your file doesn't contain binary or packed data, then you can use the **sftp** ascii subcommand (IBM documentation) to convert your files to text during the transfer.
- You must unpack mainframe files (AWS Prescriptive Guidance) that contain packed and binary content to use these files in your target environment.
- Amazon S3 objects can range in size from a minimum of 0 bytes to a maximum of 5 TB. For more information about Amazon S3 capabilities, see Amazon S3 FAQs.

Architecture

Source technology stack

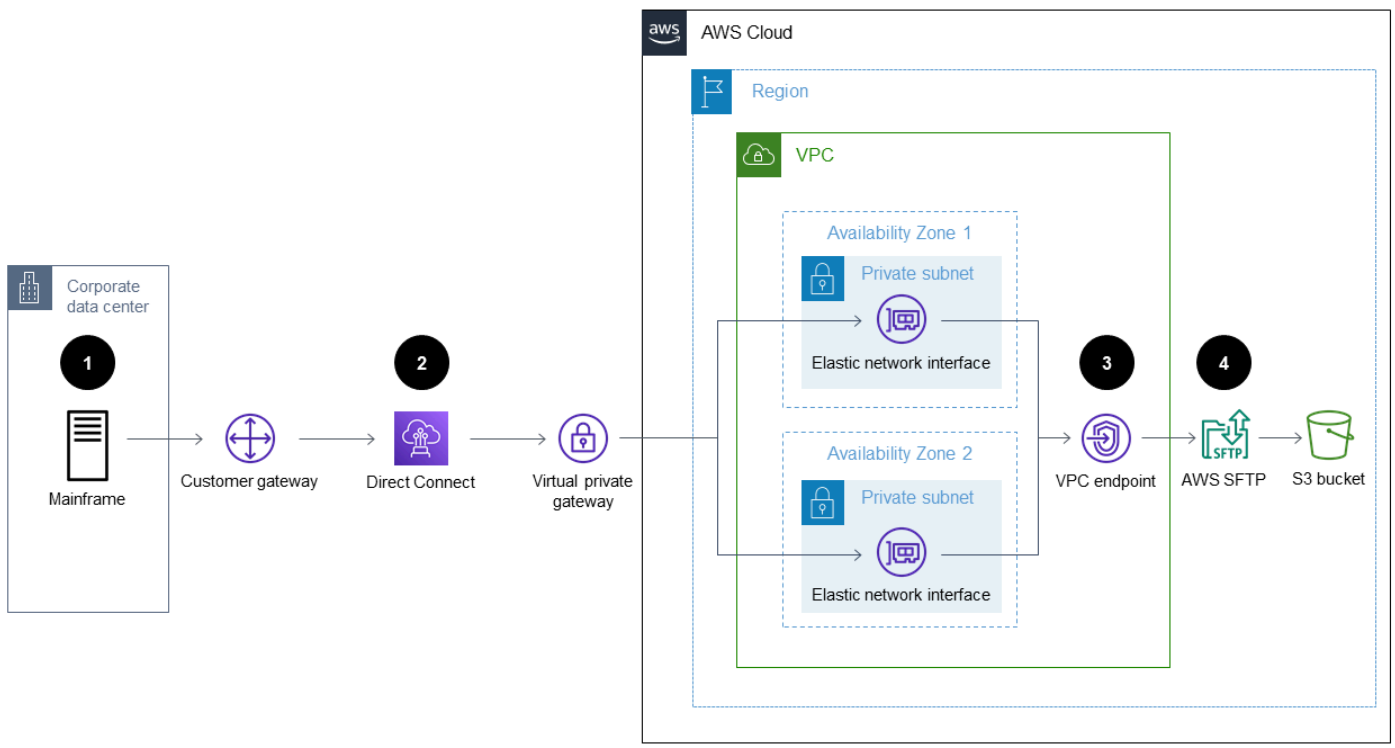
- Job control language (JCL)
- z/OS Unix shell and ISPF
- SFTP
- VSAM and flat files

Target technology stack

- Transfer Family
- Amazon S3
- Amazon Virtual Private Cloud (Amazon VPC)

Target architecture

The following diagram shows a reference architecture for using Transfer Family with SFTP to upload mainframe files directly to an S3 bucket.



The diagram shows the following workflow:

1. You use a JCL job to transfer your mainframe files from the legacy mainframe to the AWS Cloud through Direct Connect.
2. Direct Connect enables your network traffic to remain on the AWS global network and bypass the public internet. Direct Connect also enhances the network speed, starting at 50 Mbps and scaling up to 100 Gbps.
3. The VPC endpoint enables connections between your VPC resources and the supported services without using the public internet. Access to Transfer Family and Amazon S3 achieves high availability by taking place through the elastic network interfaces located in two private subnets and Availability Zones.
4. Transfer Family authenticates users and uses SFTP to receive your files from the legacy environment and move them to an S3 bucket.

Automation and scale

After the Transfer Family service is in place, you can transfer an unlimited number of files from the mainframe to Amazon S3 by using a JCL job as the SFTP client. You can also automate the file transfer by using a mainframe batch job scheduler to run the SFTP jobs when you're ready to transfer the mainframe files.

Tools

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.
- AWS Transfer Family enables you to securely scale your recurring business-to-business file transfers to Amazon S3 and Amazon EFS by using SFTP, FTPS, and FTP protocols.

Epics

Create the S3 bucket and the access policy

Task	Description	Skills required
Create the S3 bucket.	Create an S3 bucket to host the files that you transfer from your legacy environme nt.	General AWS
Create the IAM role and policy.	Transfer Family uses your AWS Identity and Access Management (IAM) role to grant access to the S3 bucket that you created earlier. Create an IAM role that includes the following IAM policy: `{ "Version": "2012-10-17", "Statement": [ { "Sid": "UserFolderListing", "Action": [ "s3:ListBucket", "s3:GetBucketLocat ion" ], "Effect": "Allow", "Resource": [`	General AWS

Task	Description	Skills required

	``
	 "arn:aws:s3:::<your-
	bucket-name>"
	            ]
	        },
	        {
	            "Sid":
	 "HomeDirObjectAcce
	ss",
	            "Effect":
	 "Allow",
	            "Action": [
	
	 "s3:PutObject",
	
	 "s3:GetObjectAcl",
	
	 "s3:GetObject",
	
	 "s3:DeleteObjectVe
	rsion",
	
	 "s3:DeleteObject",
	
	 "s3:PutObjectAcl",
	
	 "s3:GetObjectVersion"
	            ],
	            "Resource":
	 "arn:aws:s3:::<your-
	bucket-name>/*"
	        }
	    ]
	}
	``
	
	Note
	
	You must choose the Transfer use case	

Task	Description	Skills required
	when you create the IAM role.	

Define the transfer service

Task	Description	Skills required
Create the SFTP server.	1. Sign in to the AWS Management console, open the [Transfer Family console](), and then choose **Create server**. 2. Choose only **SFTP (SSH File Transfer Protocol) - file transfer over Secure Shell protocol** and then choose **Next**. 3. For **Identity provider**, choose **Service managed** and then choose **Next**. 4. For **Endpoint type**, choose **VPC hosted**. 5. For **Access**, choose **Internal**. 6. For **VPC**, choose your VPC. 7. In the **Availability Zones** section, choose your Availability Zones and subnets. 8. In the **Security Groups** section, choose your	General AWS

Task	Description	Skills required
	security group, and then choose **Next**. 9. For **Domain**, choose **Amazon S3** and then choose **Next**. 10. Leave the default options on the **Configure additional details** page and then choose **Next**. 11. Choose **Create server**. **Note** For more information about how to set up an SFTP server, see Create an SFTP-enabled server (AWS Transfer Family User Guide).	

Task	Description	Skills required
Get the server address.	1. Open the Transfer Family console and choose your server ID in the **Server ID** column. 2. In the **Endpoint details** section, for **Endpoint type**, choose the endpoint ID. This takes you to the Amazon VPC console. 3. On the **Details** tab of the Amazon VPC console, find the DNS names next to **DNS names**.	General AWS
Create the SFTP client key pair.	Create an SSH key pair for either Microsoft Windows or macOS/Linux/UNIX.	General AWS, SSH

Task	Description	Skills required
Create the SFTP user.	1. Open the [Transfer Family console](), choose **Servers** from the navigation pane, and then select your server. 2. In the **Server ID** column, choose the server ID for your server and then choose **Add user**. 3. For **Username**, enter a user name that matches your SSH key pair user name. 4. For **Role**, choose the IAM role that you created earlier. 5. For **Home directory** , choose the S3 bucket that you created earlier. 6. For **SSH public keys**, enter the key pair that you created earlier. 7. Choose **Add**.	General AWS

Transfer the mainframe file

Task	Description	Skills required
Send the SSH private key to the mainframe.	Use SFTP or SCP to send the SSH private key to the legacy environment. SFTP example:	Mainframe, z/OS Unix shell, FTP, SCP

Task	Description	Skills required

	<pre>sftp [USERNAME@mainframeIP]
	[password]
	cd [/u/USERNAME]
	put [your-key-pair-file]</pre> SCP example: <pre>scp [your-key-pair-file] [USERNAME@MainframeIP]:/[u/USERNAME]</pre> Next, store the SSH key in the z/OS Unix file system under the user name that will later run the file transfer batch job (for example, `/u/CONTROLM`). **Note** For more information about z/OS Unix shell, see An introduction to the z/OS shells (IBM documentation).	

Task	Description	Skills required

Create the JCL SFTP client.	Because mainframes don't have a native SFTP client, you must use the BPXBATCH utility to run the SFTP client from the z/OS Unix shell.
	
	In the ISPF editor, create the JCL SFTP client. For example:
	
	`//JOBNAM JOB ...`
	`//*************************************************************`
	`//SFTP EXEC PGM=BPXBATCH,REGION=0M`
	`//STDPARM DD *`
	`SH cp "//'MAINFRAME.FILE.NAME'" filename.txt;`
	`echo 'put filename.txt' > uplcmd;`
	`sftp -b uplcmd -i ssh_private_key_file ssh_username@<transfer service ip or DNS>;`
	`//SYSPRINT DD SYSOUT=*`
	`//STDOUT DD SYSOUT=*`
	`//STDENV DD *`
	`//STDERR DD SYSOUT=*`
	
	Note
	For more information about how to run a command in the z/OS Unix shell, see
		JCL, Mainframe, z/OS Unix shell

Task	Description	Skills required
	The BPXBATCH utility (IBM documentation). For more information about how to create or edit JCL jobs in z/OS, see What is ISPF? and The ISPF editor (IBM documentation).	
Run the JCL SFTP client.	1. In the ISPF editor, enter **SUB**, and then press the ENTER key after the JCL job is created. 2. Monitor the mainframe's file transfer batch job activity in SDSF. **Note** For more information about how to check the activity of batch jobs, see z/OS SDSF User's Guide (IBM documentation).	Mainframe, JCL, ISPF

Task	Description	Skills required
Validate the file transfer.	1. Sign in to the AWS Management console, open the Amazon S3 console, and then choose **Buckets** from the navigation pane. 2. Choose the bucket that's associated with your Transfer Family. 3. In the **Objects** section of the **Objects** tab, find the file that you transferred from the mainframe.	General AWS
Automate the JCL SFTP client.	Use job scheduler to automatically trigger the JCL SFTP client. **Note** You can use mainframe job schedulers, such as BMC Control-M or CA Workload Automation, to automate batch jobs for file transfers based on time and other batch job dependencies.	Job scheduler

Related resources

- How AWS Transfer Family works

- [Mainframe Modernization with AWS]

Secure and streamline user access in a Db2 federation database on AWS by using trusted contexts

Sai Parthasaradhi, Amazon Web Services

Summary

Many companies are migrating their legacy mainframe workloads to Amazon Web Services (AWS). This migration includes shifting IBM Db2 for z/OS databases to Db2 for Linux, Unix, and Windows (LUW) on Amazon Elastic Compute Cloud (Amazon EC2). During a phased migration from on premises to AWS, users might need to access data in IBM Db2 z/OS and in Db2 LUW on Amazon EC2 until all applications and databases are fully migrated to Db2 LUW. In such remote data-access scenarios, user authentication can be challenging because different platforms use different authentication mechanisms.

This pattern covers how to set up a federation server on Db2 for LUW with Db2 for z/OS as a remote database. The pattern uses a trusted context to propagate a user's identity from Db2 LUW to Db2 z/OS without re-authenticating on the remote database. For more information about trusted contexts, see the Additional information section.

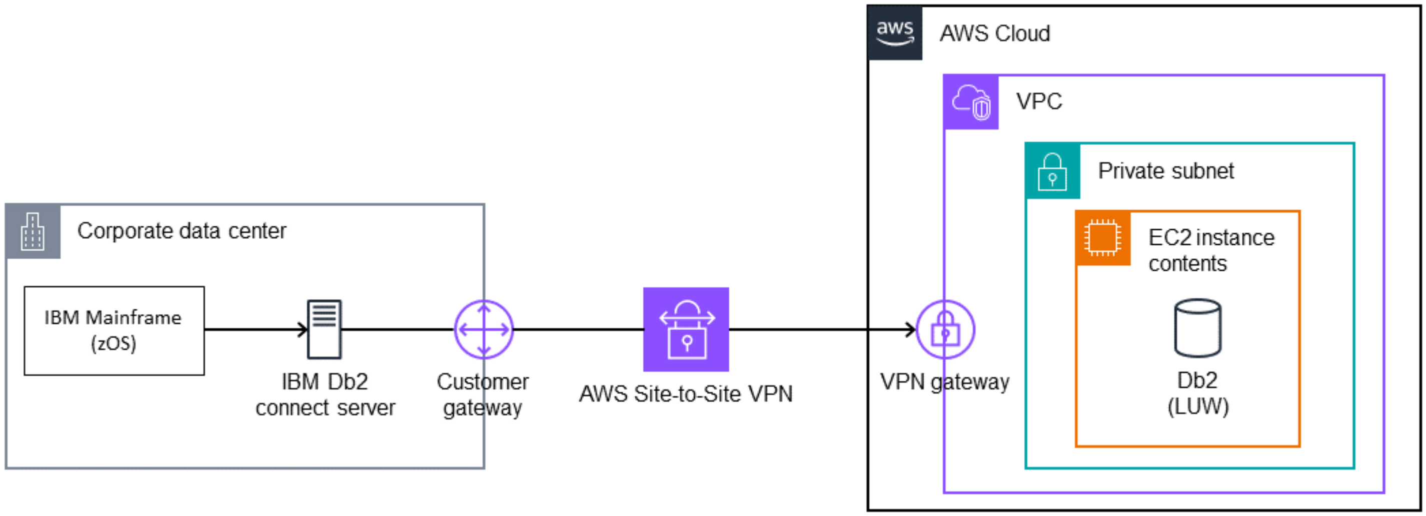
Prerequisites and limitations

Prerequisites

- An active AWS account
- A Db2 instance running on an Amazon EC2 instance
- A remote Db2 for z/OS database running on premises
- The on-premises network connected to AWS through AWS Site-to-Site VPN or AWS Direct Connect

Architecture

Target architecture



Tools

AWS services

- [Amazon Elastic Compute Cloud (Amazon EC2)](#) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- [AWS Site-to-Site VPN](#) helps you pass traffic between instances that you launch on AWS and your own remote network.

Other tools

- [db2cli](#) is the Db2 interactive command line interface (CLI) command.

Epics

Enable federation on the Db2 LUW database running on AWS

Task	Description	Skills required
Enable federation on the DB2 LUW DB.	To enable federation on DB2 LUW, run the following command. `update dbm cfg using federated YES`	DBA

Task	Description	Skills required

Restart the database.	To restart the database, run the following command.
	`db2stop force;`
	`db2start;`
		DBA

Catalog the remote database

Task	Description	Skills required
Catalog the remote Db2 z/OS subsystem.	To catalog the remote Db2 z/OS database on Db2 LUW running on AWS, use the following example command. `catalog TCPIP NODE tcpnode REMOTE mainframehost SERVER mainframeport`	DBA
Catalog the remote database.	To catalog the remote database, use the following example command. `catalog db dbnam1 as ndbnam1 at node tcpnode`	DBA

Create the remote server definition

Task	Description	Skills required
Collect user credentials for the remote Db2 z/OS database.	Before proceeding with the next steps, gather the following information:	DBA

Task	Description	Skills required
	- Db2 z/OS subsystem name – The cataloged Db2 z/OS name on LUW from the previous step (for example, `ndbnam1`) - Db2 z/OS version – The Db2 z/OS subsystem version (for example, `12`) - Db2 z/OS user ID – The user with the BIND privilege, which is needed to create only the server definition (for example, `dbuser1`) - Db2 z/OS password – The password for `dbuser1` (for example, `dbpasswd`) - Db2 z/OS proxy user – The ID of the proxy user, which will be used to establish trusted connection (for example, `zproxy`) - Db2 z/OS proxy password – The password for the `zproxy` user (for example, `zproxy`)	
Create the DRDA wrapper.	To create the DRDA wrapper, run the following command. `CREATE WRAPPER DRDA;`	DBA

Task	Description	Skills required
Create the server definition.	To create the server definitio n, run the following example command. `CREATE SERVER ndbserver TYPE DB2/ZOS VERSION 12 WRAPPER DRDA AUTHORIZATION "dbuser1" PASSWORD "dbpasswd " OPTIONS ( DBNAME 'ndbnam1',FED_PROX Y_USER 'ZPROXY' );` In this definition, `FED_PROXY _USER` specifies the proxy user that will be used for establishing trusted connectio ns to the Db2 z/OS database. The authorization user ID and password are required only for creating the remote server object in the Db2 LUW database. They won't be used later during runtime.	DBA

Create user mappings

Task	Description	Skills required
Create a user mapping for the proxy user.	To create a user mapping for proxy user, run the following command. `CREATE USER MAPPING FOR ZPROXY SERVER ndbserver OPTIONS (REMOTE_AUTHID`	DBA

Task	Description	Skills required
	``` 'ZPROXY', REMOTE_PA SSWORD 'zproxy'); ```	
Create user mappings for each user on Db2 LUW.	Create user mappings for all the users on the Db2 LUW database on AWS who need to access remote data through the proxy user. To create the user mappings, run the following command. ``` CREATE USER MAPPING FOR PERSON1 SERVER ndbserver OPTIONS (REMOTE_AUTHID 'USERZID', USE_TRUST ED_CONTEXT 'Y'); ``` The statement specifies that a user on Db2 LUW (`PERSON1`) can establish a trusted connection to the remote Db2 z/OS database (`USE_TRUSTED_CONTEXT 'Y'`). After the connection is established through the proxy user, the user can access the data by using the Db2 z/OS user ID (`REMOTE_AUTHID 'USERZID'`).	DBA

Create the trusted context object

Task	Description	Skills required
Create the trusted context object.	To create the trusted context object on the remote Db2 z/OS database, use the following example command. `CREATE TRUSTED CONTEXT CTX_LUW_ZOS BASED UPON CONNECTION USING SYSTEM AUTHID ZPROXY ATTRIBUTES ( ADDRESS '10.10.10.10' ) NO DEFAULT ROLE ENABLE WITH USE FOR PUBLIC WITHOUT AUTHENTICATION;` In this definition, `CTX_LUW_ZOS` is an arbitrary name for the trusted context object. The object contains the proxy user ID and the IP address of the server from which the trusted connection must originate. In this example, the server the Db2 LUW database on AWS. You can use the domain name instead of the IP address. The clause `WITH USE FOR PUBLIC WITHOUT AUTHENTICATION` indicates that switching the user ID on a trusted connection is	DBA

Task	Description	Skills required
	allowed for every user ID. A password doesn't need to be provided.	

Related resources

- IBM Resource Access Control Facility (RACF)
- IBM Db2 LUW Federation
- Trusted contexts

Additional information

Db2 trusted contexts

A trusted context is a Db2 database object that defines a trust relationship between a federated server and a remote database server. To define a trusted relationship, the trusted context specifies trust attributes. There are three types of trust attributes:

- The system authorization ID that makes the initial database connection request
- The IP address or domain name from which the connection is made
- The encryption setting for data communications between the database server and the database client

A trusted connection is established when all attributes of a connection request match the attributes specified in any trusted context object defined on the server. There are two types of trusted connections: implicit and explicit. After an implicit trusted connection is established, a user inherits a role that isn't available to them outside the scope of that trusted connection definition. After an explicit trusted connection is established, users can be switched on the same physical connection, with or without authentication. In addition, Db2 users can be granted roles that specify privileges that are for use only within the trusted connection. This pattern uses an explicit trusted connection.

Trusted context in this pattern

After the pattern is complete, PERSON1 on Db2 LUW accesses remote data from Db2 z/OS by using a federated trusted context. The connection for PERSON1 is established through a proxy user if the connection originates from the IP address or domain name that is specified in the trusted context definition. After the connection is established, PERSON1's corresponding Db2 z/OS user ID is switched without re-authentication, and the user can access the data or objects based on the Db2 privileges set up for that user.

Benefits of federated trusted contexts

- This approach maintains the principle of least privilege by eliminating the use of a common user ID or application ID that would need a superset of all the privileges required by all users.
- The real identity of the user who performs the transaction on both the federated and remote database is always known and can be audited.
- Performance improves because the physical connection is being reused across the users without the need for the federated server to re-authenticate.

Transfer large-scale Db2 z/OS data to Amazon S3 in CSV files

Bruno Sahinoglu, Abhijit Kshirsagar, and Ivan Schuster, Amazon Web Services

Summary

A mainframe is still a system of record in many enterprises, containing a massive amount of data including master data entities with records of current as well as the historical business transactions. It is often siloed and not easily accessed by the distributed systems within the same enterprise. With the emergence of cloud technology and big data democratization, enterprises are interested in using the insights hidden in the mainframe data to develop new business capabilities.

With that objective, enterprises are looking to open their mainframe Db2 data to their Amazon Web Services (AWS) Cloud environment. The business reasons are several and the transfer methods differ from case to case. You might prefer to connect your application directly to the mainframe, or you might prefer to replicate your data in near real time. If the use case is to feed a data warehouse or a data lake, having an up-to-date copy is no longer a concern, and the procedure described in this pattern might suffice, especially if you want to avoid any third-party product licensing costs. Another use case might be the mainframe data transfer for a migration project. In a migration scenario, data is required for performing the functional equivalence testing. The approach described in this post is a cost effective way to transfer the Db2 data to the AWS Cloud environment.

Because Amazon Simple Storage Service (Amazon S3) is one of the most integrated AWS services, you can access the data from there and gather insights directly by using other AWS services such as Amazon Athena, AWS Lambda functions, or Amazon QuickSight . You can also load the data to Amazon Aurora or Amazon DynamoDB by using AWS Glue or AWS Database Migration Service (AWS DMS). With that aim in mind, this describes how to unload Db2 data in CSV files in ASCII format on the mainframe and transfer the files to Amazon S3.

For this purpose, [mainframe scripts](#) have been developed to help to generate job control languages (JCLs) to unload and transfer as many Db2 tables as you need.

Prerequisites and limitations

Prerequisites

- An IBM z/OS operating system user with authorization to run Restructured Extended Executor (REXX) and JCL scripts.

- Access to z/OS Unix System Services (USS) to generate SSH (Secure Shell) private and public keys.
- A writable S3 bucket. For more information, see [Create your first S3 bucket] in the Amazon S3 documentation.
- An AWS Transfer Family SSH File Transfer Protocol (SFTP)-enabled server using **Service managed** as the identity provider and Amazon S3 as the AWS storage service. For more information, see [Create an SFTP-enabled server] in the AWS Transfer Family documentation.

Limitations

- This approach isn't suitable for near real-time or real-time data synchronization.
- Data can be moved only from Db2 z/OS to Amazon S3, not the other way around.

Architecture

Source technology stack

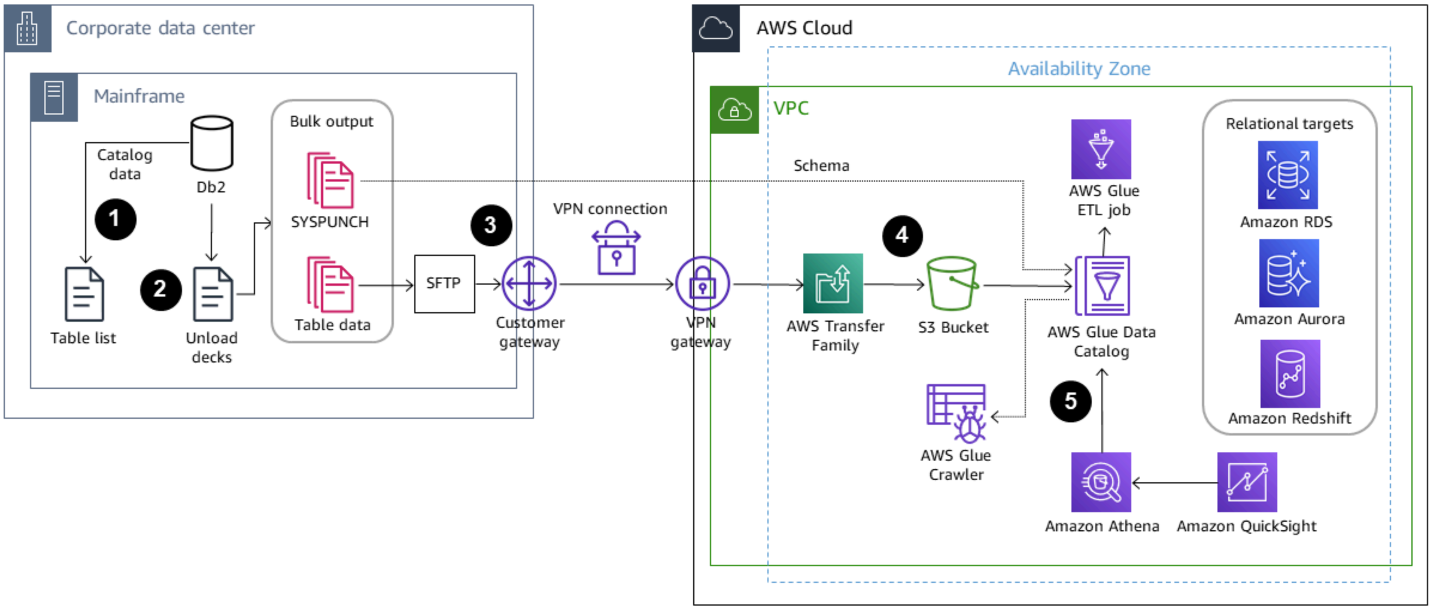
- Mainframe running Db2 on z/OS

Target technology stack

- AWS Transfer Family
- Amazon S3
- Amazon Athena
- Amazon QuickSight
- AWS Glue
- Amazon Relational Database Service (Amazon RDS)
- Amazon Aurora
- Amazon Redshift

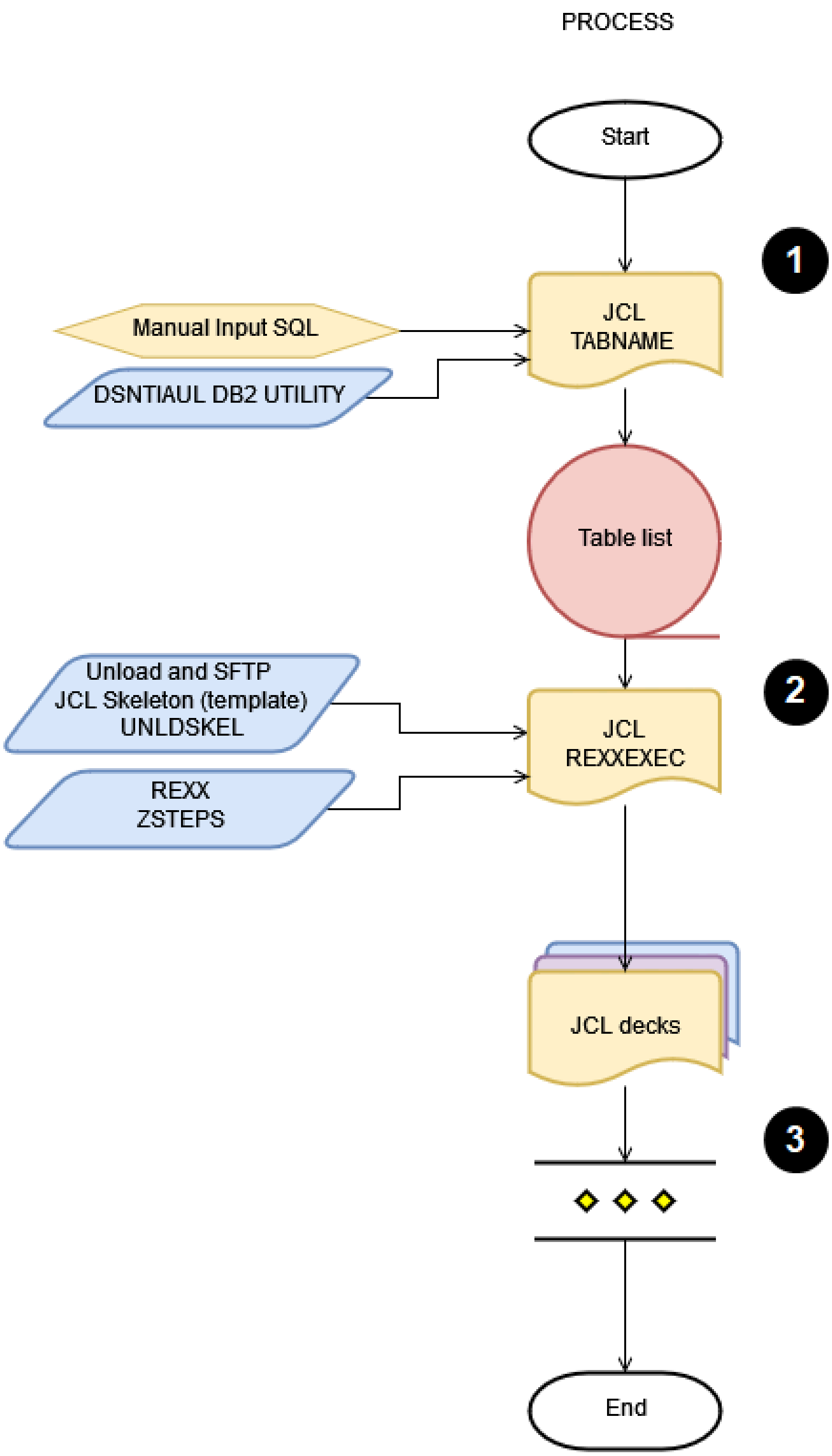
Source and target architecture

The following diagram shows the process for generating, extracting, and transferring Db2 z/OS data in ASCII CSV format to an S3 bucket.



1. A list of tables is selected for data migration from the Db2 catalog.
2. The list is used to drive the generation of unload jobs with the numeric and data columns in the external format.
3. The data is then transferred over to Amazon S3 by using AWS Transfer Family.
4. An AWS Glue extract, transform, and load (ETL) job can transform the data and load it to a processed bucket in the specified format, or AWS Glue can feed the data directly into the database.
5. Amazon Athena and Amazon QuickSight can be used to query and render the data to drive analytics.

The following diagram shows a logical flow of the entire process.



1. The first JCL, called TABNAME, will use the Db2 utility DSNTIAUL to extract and generate the list of tables that you plan to unload from Db2. To choose your tables, you must manually adapt the SQL input to select and add filter criteria to include one or more Db2 schemas.

2. The second JCL, called REXXEXEC, will use the a JCL skeleton and the REXX program that is provided to process the Table list created by the JCL TABNAME and generate one JCL per table name. Each JCL will contain one step for unloading the table and another step for sending the file to the S3 bucket by using the SFTP protocol.
3. The last step consists of running the JCL to unload the table and transferring the file to AWS. The entire process can be automated using a scheduler on premises or on AWS.

Tools

AWS services

- Amazon Athena is an interactive query service that helps you analyze data directly in Amazon Simple Storage Service (Amazon S3) by using standard SQL.
- Amazon Aurora is a fully managed relational database engine that's built for the cloud and compatible with MySQL and PostgreSQL.
- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- Amazon QuickSight is a cloud-scale business intelligence (BI) service that helps you visualize, analyze, and report your data in a single dashboard.
- Amazon Redshift is a managed petabyte-scale data warehouse service in the AWS Cloud.
- Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database in the AWS Cloud.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Transfer Family is a secure transfer service that enables you to transfer files into and out of AWS storage services.

Mainframe tools

- SSH File Transfer Protocol (SFTP) is a secure file transfer protocol that allows remote login to and file transfer between servers. SSH provides security by encrypting all traffic.
- DSNTIAUL is a sample program provided by IBM for unloading data.
- DSNUTILB is a utilities batch program provided by IBM for unloading data with different options from DSNTIAUL.

- z/OS OpenSSH is a port of Open Source Software SSH running on the Unix System Service under the IBM operating system z/OS. SSH is a secure, encrypted connection program between two computers running on a TCP/IP network. It provides multiple utilities, including ssh-keygen.
- REXX (Restructured Extended Executor) script is used to automate JCL generation with the Db2 Unload and SFTP steps.

Code

The code for this pattern is available in the GitHub unloaddb2 repository.

Best practices

For the first unload, the generated JCLs should unload the entire table data.

After the first full unload, perform incremental unloads for better performance and cost savings. pdate the SQL query in the template JCL deck to accommodate any changes to the unload process.

You can convert the schema manually or by using a script on Lambda with the Db2 SYSPUNCH as an input. For an industrial process, AWS Schema Conversion Tool (SCT) is the preferred option.

Finally, use a mainframe-based scheduler or a scheduler on AWS with an agent on the mainframe to help manage and automate the entire process.

Epics

Set up the S3 bucket

Task	Description	Skills required
Create the S3 bucket.	For instructions, see Create your first S3 bucket.	General AWS

Set up the Transfer Family server

Task	Description	Skills required
Create an SFTP-enabled server.	To open and create an SFTP server on the AWS Transfer	General AWS

Task	Description	Skills required
	Family console, do the following: 1. On the Choose protocols, page, select the **SFTP (SSH File Transfer Protocol) – file transfer over Secure Shell** check box. 2. For the identity provider, choose **Service managed**. 3. For the endpoint, choose **Publicly accessible**. 4. For the domain, choose **Amazon S3**. 5. On the **Configure additional details** page, keep the default settings. 6. Create the server.	
Create an IAM role for Transfer Family.	To create an AWS Identity and Access Management (IAM) role for Transfer Family to access Amazon S3, follow the instructions in Create an IAM role and policy.	AWS administrator
Add an Amazon S3 service-m anaged user.	To add the Amazon S3 service-managed user, follow the instructions in the AWS documentation, and use your mainframe user ID.	General AWS

Secure the communication protocol

Task	Description	Skills required
Create the SSH key.	Under your mainframe USS environment, run the following command. `ssh-keygen -t rsa` **Note** When prompted for a passphrase, keep it empty.	Mainframe developer

Give the right authorization levels to the SSH folder and key files.	By default, the public and private keys will be stored in the user directory `/u/home/username/.ssh` .
	You must give the authorization 644 to the key files and 700 to the folder.
	`chmod 644 .ssh/id_rsa`
	`chmod 700 .ssh`
		Mainframe developer

Copy the public key content to your Amazon S3 service-managed user.	To copy the USS-generated public key content, open the [AWS Transfer Family console](https://console.aws.amazon.com/transfer/). 1. In the navigation pane, choose **Servers**. 2. Choose the identifier in the **Server ID** column to see the **Server details**	Mainframe developer

Task	Description	Skills required
	3. Under **Users**, choose a user name to see the **User details** 4. Under **SSH public keys**, choose **Add SSH public key** to add the public key to a user. For the SSH public key, enter your public key.Your key is validated by the service before you can add your new user. 5. Choose **Add key**.	

Generate the JCLs

Task	Description	Skills required
Generate the in-scope Db2 table list.	Provide input SQL to create a list of the tables that are scoped for data migration . This step requires you to specify selection criteria quering the Db2 catalog table SYSIBM.SYSTABLES using a SQL where clause. Filters can be customized to include a specific schema or table names that start with a particular prefix or based on a timestamp for incremental unload. Output is captured in a physical sequential (PS) dataset on the mainframe.	Mainframe developer

Task	Description	Skills required
	This dataset will act as input for the next phase of JCL generation. Before you use the JCL TABNAME (You can rename it if necessary), make the following changes: 1. Substitute <Jobcard> with a job class and a user that is authorized to run Db2 utilities. 2. Substitute <HLQ1> or customize the output dataset names to meet your site standards. 3. Update the STEPLIB stack of PDSEs (partitioned data set extended) according to your site standards. The example in this pattern uses the IBM defaults. 4. Substitute the PLAN name and LIB with your installat ion-specific values. 5. Substitute <Schema> and <Prefix> with your selection criteria for the Db2 catalog. 6. Save the resultant JCL in a PDS (partitioned data set) library. 7. Submit the JCL.	

Task	Description	Skills required

	Db2 table list extraction job
	``
	<Jobcard>
	//*
	//* UNLOAD ALL THE TABLE NAMES FOR A PARTICULAR SCHEMA
	//*
	//STEP01 EXEC PGM=IEFBR14
	//*
	//DD1 DD DISP=(MOD,DELETE,DELETE),
	// UNIT=SYSDA,
	// SPACE=(1000,(1,1)),
	// DSN=<HLQ1>.DSN81210.TABLIST
	//*
	//DD2 DD DISP=(MOD,DELETE,DELETE),
	// UNIT=SYSDA,
	// SPACE=(1000,(1,1)),
	// DSN=<HLQ1>.DSN81210.SYSPUNCH
	//*
	//UNLOAD EXEC PGM=IKJEFT01,DYNAMNBR=20
	//SYSTSPRT DD SYSOUT=*
	//STEPLIB DD DISP=SHR,DSN=DSNC10.DBCG.SDSNEXIT
	// DD DISP=SHR,DSN=DSNC10.SDSNLOAD
	// DD DISP=SHR,DSN=CEE.SCEERUN
	// DD DISP=SHR,DSN=DSNC10.DBCG.RUNLIB.LOAD
	``	

Task	Description	Skills required

	``
	//SYSTSIN DD *
	  DSN SYSTEM(DBCG)
	  RUN PROGRAM(D
	SNTIAUL) PLAN(DSNT
	IB12) PARMS('SQL') -
	       LIB('DSNC
	10.DBCG.RUNLIB.LOAD')
	  END
	//SYSPRINT DD SYSOUT=*
	//*
	//SYSUDUMP DD SYSOUT=*
	//*
	//SYSREC00 DD DISP=(NEW
	,CATLG,DELETE),
	// UNIT=SYSD
	A,SPACE=(32760,(10
	00,500)),
	// DSN=<HLQ1
	>.DSN81210.TABLIST
	//*
	//SYSPUNCH DD DISP=(NEW
	,CATLG,DELETE),
	// UNIT=SYSD
	A,SPACE=(32760,(10
	00,500)),
	// VOL=SER=S
	CR03,RECFM=FB,LREC
	L=120,BLKSIZE=12
	// DSN=<HLQ1
	>.DSN81210.SYSPUNCH
	//*
	//SYSIN DD *
	   SELECT CHAR(CREA
	TOR), CHAR(NAME)
	     FROM SYSIBM.SY
	STABLES
	    WHERE OWNER =
	 '<Schema>'
	      AND NAME LIKE
	 '<Prefix>%'
	      AND TYPE = 'T';
	``	

Task	Description	Skills required
	``` /* ```	

Task	Description	Skills required
Modify the JCL templates.	The JCL templates that are provided with this pattern contain a generic job card and library names. However, most mainframe sites will have their own naming standards for dataset names, library names, and job cards. For example, a specific job class might be required to run Db2 jobs. The Job Entry Subsytem implementations JES2 and JES3 can impose additional changes. Standard load libraries might have a different first qualifier than SYS1, which is IBM default. Therefore, customize the templates to account for your site-specific standards before you run them. Make the following changes in the skeleton JCL UNLDSKEL: 1. Modify the job card with a job class and user that is authorized to run Db2 utilities. 2. Customize output dataset names to meet your site standards. 3. Update the STEPLIB stack of PDSEs according to	Mainframe developer

Task	Description	Skills required

	your site standards. The example in this pattern uses the IBM defaults.
	4. Substitute <DSN> with your Db2 subsystem name and correlation ID.
	5. Save the resultant JCL in a PDS library that is part of your ISPSLIB stack, which is the standard skeleton template library for ISPF.
	
	Unload and SFTP JCL skeleton
	
	`//&USRPFX.U JOB (DB2UNLOAD),'JOB', CLASS=A,MSGCLASS=A,`
	`//         TIME=1440 ,NOTIFY=&USRPFX`
	`//* DELETE DATASETS`
	`//STEP01   EXEC PGM=IEFBR14`
	`//DD01     DD DISP=(MOD ,DELETE,DELETE),`
	`//            UNIT=SYSDA,`
	`//            SPACE=(TRK,(1,1)),`
	`// DSN=&USRPFX..DB2.PUNCH.&JOBNAME`
	`//DD02     DD DISP=(MOD ,DELETE,DELETE),`
	`//            UNIT=SYSDA,`
	`//            SPACE=(TRK,(1,1)),`	

Task	Description	Skills required

	``
	// DSN=&USRPFX..DB2.U
	NLOAD.&JOBNAME
	//*
	//* RUNNING DB2
	 EXTRACTION BATCH JOB
	 FOR AWS DEMO
	//*
	//UNLD01 EXEC
	 PGM=DSNUTILB,REGIO
	N=0M,
	// PARM='<DSN>,UNLOAD'
	//STEPLIB DD
	 DISP=SHR,DSN=DSNC1
	0.DBCG.SDSNEXIT
	// DD DISP=SHR,
	DSN=DSNC10.SDSNLOAD
	//SYSPRINT DD SYSOUT=*
	//UTPRINT DD SYSOUT=*
	//SYSOUT DD SYSOUT=*
	//SYSPUN01 DD
	 DISP=(NEW,CATLG,DE
	LETE),
	// SPACE=(CY
	L,(1,1),RLSE),
	// DSN=&USRPFX..DB2.P
	UNCH.&JOBNAME
	//SYSREC01 DD
	 DISP=(NEW,CATLG,DE
	LETE),
	// SPACE=(CY
	L,(10,50),RLSE),
	// DSN=&USRPFX..DB2.U
	NLOAD.&JOBNAME
	//SYSPRINT DD SYSOUT=*
	//SYSIN DD *
	  UNLOAD
	  DELIMITED COLDEL ','
	  FROM TABLE &TABNAME
	  UNLDDN SYSREC01
	  PUNCHDDN SYSPUN01
	  SHRLEVEL CHANGE
	 ISOLATION UR;
	``	

Task	Description	Skills required

	``
	 /*
	//*
	//* FTP TO AMAZON S3
	 BACKED FTP SERVER IF
	 UNLOAD WAS SUCCESSFUL
	//*
	//SFTP EXEC PGM=BPXBA
	TCH,COND=(4,LE),RE
	GION=0M
	//STDPARM DD *
	 SH cp "//'&USRP
	FX..DB2.UNLOAD.&JO
	BNAME'"
	   &TABNAME..csv;
	 echo "ascii
	 " >> uplcmd;
	 echo "PUT &TABNAME.
	.csv " >>>> uplcmd;
	 sftp -b uplcmd -i .ssh/
	id_rsa &FTPUSER.
	@&FTPSITE;
	 rm &TABNAME..csv;
	 //SYSPRINT DD SYSOUT=*
	 //STDOUT DD SYSOUT=*
	 //STDENV DD *
	 //STDERR DD SYSOUT=*
	``	

Task	Description	Skills required
Generate the Mass Unload JCL.	This step involves running a REXX script under an ISPF environment by using JCL. Provide the list of in-scope tables created on the first step as input for mass JCL generation against the `TABLIST DD` name. The JCL will generate one new JCL per table name in a user-specified partitioned dataset specified against the `ISPFILE DD` name. Allocate this library beforehand. Each new JCL will have two steps: one step to unload the Db2 table into a file, and one step to send the file to the S3 bucket. Make the following changes in the JCL REXXEXEC (you can change the name): 1. Substitute `Job card user ID` with a mainframe user ID that has unload authority on the tables. Substitute `SYSPROC`, `ISPPLIB`, `ISPSLIB`, `ISPMLIB`, and `ISPTLIB` `<HLQ1>` value or customize the `DSN` to meet your site standards. To find out your installation-specific values,	Mainframe developer

Task	Description	Skills required
	you use the command `TSO ISRDDN`. 2. Substitute `<MFUSER>` with a user ID that has job-running privileges in your installation. 3. Substitute `<FTPUSER>` with a user ID that has the USS and FTP privilege in your installation.. It is assumed that this user ID and its SSH security keys are in place in the appropriate Unix Systems Services directory on the mainframe. 4. Substitute `<AWS TransferFamily IP>` with the AWS Transfer Family IP address or the domain name. This address will be used for the SFTP step. 5. Submit the JCL after applying the site standard accommodation and updating the REXX program as described below. **Mass JCL generation job** `//RUNREXX JOB (CREATEJCL),'RUNS ISPF TABLIST',`	

Task	Description	Skills required

```
CLASS=A,MSGCLASS=A,

//         TIME=1440
,NOTIFY=&SYSUID
//* Most of the values
 required can be updated
 to your site specific
//* values using the
 command 'TSO ISRDDN' in
 your ISPF session.
//* Update all the lines
 tagged with //update
 marker to desired
//* site specific
 values.
//ISPF EXEC PGM=IKJEF
T01,REGION=2048K,D
YNAMNBR=25
//SYSPROC   DD
 DISP=SHR,DSN=USER.
Z23D.CLIST
//SYSEXEC   DD
 DISP=SHR,DSN=<HLQ1
>.TEST.REXXLIB
//ISPPLIB   DD
 DISP=SHR,DSN=ISP.S
ISPPENU
//ISPSLIB   DD
 DISP=SHR,DSN=ISP.S
ISPSENU
//          DD
 DISP=SHR,DSN=<HLQ1
>.TEST.ISPSLIB
//ISPMLIB   DD
 DSN=ISP.SISPMENU,D
ISP=SHR
//ISPTLIB   DD
 DDNAME=ISPTABL
//          DD DSN=ISP.S
ISPTENU,DISP=SHR
```

Task	Description	Skills required

	``
	//ISPTABL DD
	 LIKE=ISP.SISPTENU,
	UNIT=VIO
	//ISPPROF DD
	 LIKE=ISP.SISPTENU,
	UNIT=VIO
	//ISPLOG DD
	 SYSOUT=*,RECFM=VA,
	LRECL=125
	//SYSPRINT DD SYSOUT=*
	//SYSTSPRT DD
	 SYSOUT=*
	//SYSUDUMP DD
	 SYSOUT=*
	//SYSDBOUT DD
	 SYSOUT=*
	//SYSTSPRT DD
	 SYSOUT=*
	//SYSUDUMP DD
	 SYSOUT=*
	//SYSDBOUT DD
	 SYSOUT=*
	//SYSHELP DD
	 DSN=SYS1.HELP,DISP
	=SHR
	//SYSOUT DD SYSOUT=*
	//* Input list of
	 tablenames
	//TABLIST DD
	 DISP=SHR,DSN=<HLQ1
	>.DSN81210.TABLIST
	//* Output pds
	//ISPFILE DD
	 DISP=SHR,DSN=<HLQ1
	>.TEST.JOBGEN
	//SYSTSIN DD *
	ISPSTART CMD(ZSTEPS
	 <MFUSER> <FTPUSER> <AWS
	 TransferFamily IP>)
	/*
	``	

Task	Description	Skills required
	Before you use the REXX script, make the following changes: 1. Save the REXX script in a PDS library defined under the `SYSEXEC` stack in the JCL REXXEXEC edited in the previous step with ZSTEPS as the member name. If you want to rename it, you should update the JCL to accommodate your needs. 2. This script uses the trace option to print additiona l information in case there are errors. You can instead add error handling code after the `EXECIO`, `ISPEXEC`, and `TSO` statements, and remove the trace line. 3. This script generates member names by using the `LOD`nnnnn naming convention, which can support up to 100,000 members. If you have more than 100,000 tables, use a shorter prefix, and adjust the numbers in the `tempjob` statement.	

Task	Description	Skills required

	ZSTEPS REXX script
	``
	/*REXX - */
	/* 10/27/2021 - added new parms to accommodate ftp */
	Trace "o"
	    parse arg usrpfx ftpuser ftpsite
	    Say "Start"
	    Say "Ftpuser: " ftpuser "Ftpsite:" ftpsite
	    Say "Reading table name list"
	    "EXECIO * DISKR TABLIST (STEM LINE. FINIS"
	    DO I = 1 TO LINE.0
	      Say I
	      suffix = I
	      Say LINE.i
	      Parse var LINE.i schema table rest
	      tabname = schema !! "." !! table
	      Say tabname
	      tempjob= "LOD" !! RIGHT("0000" !! i, 5)
	      jobname=tempjob
	      Say tempjob
	      ADDRESS ISPEXEC "FTOPEN "
	      ADDRESS ISPEXEC "FTINCL UNLDSKEL"
	      /* member will be saved in ISPDSN library allocated in JCL */
	``	

Task	Description	Skills required

	``
	      ADDRESS ISPEXEC
	 "FTCLOSE NAME("tem
	pjob")"
	    END
	
	    ADDRESS TSO "FREE
	 F(TABLIST) "
	    ADDRESS TSO "FREE
	 F(ISPFILE) "
	
	exit 0
	``	

Run the JCLs

Task	Description	Skills required
Perform the Db2 Unload step.	After the JCL generation, you will have as many JCLs as you have tables that need to be unloaded. This story uses a JCL-generated example to explain the structure and the most important steps. No action is required on your part. The following information is for reference only. If your intention is to submit the JCLs that you have generated in the previous step, skip to the *Submit the LODnnnnn JCLs* task. When unloading Db2 data using a JCL with the IBM	Mainframe developer, System engineer

Task	Description	Skills required

	provided DSNUTILB Db2 utility, you must make sure that the unloaded data does not contain compressed numeric data. To accomplis h this, use the DSNUTILB `DELIMITED` parameter.
	
	The `DELIMITED` parameter supports unloading the data in CSV format by adding a character as the delimiter and double quotation marks for the text field, removing the padding in the VARCHAR column, and converting all the numeric fields into EXTERNAL FORMAT, including the DATE fields.
	
	The following example shows what the unload step in the generated JCL looks like, using the comma character as a delimiter.
	
	`UNLOAD`
	`DELIMITED COLDEL ','`
	`FROM TABLE SCHEMA_NA ME.TBNAME`
	`UNLDDN SYSREC01`
	`PUNCHDDN SYSPUN01`
	`SHRLEVEL CHANGE`
	`ISOLATION UR;`	

Task	Description	Skills required

Perform the SFTP step.	To use the SFTP protocol from a JCL, use the BPXBATCH utility.
	
	The SFTP utility can't access the MVS datasets directly. You can use the copy command (`cp`) to copy the sequential file `&USRPFX..DB2.UNLOAD.&JOBNAME` to the USS directory, where it becomes `&TABNAME..csv` .
	
	Run the `sftp` command using the private key (`id_rsa`) and using the RACF user ID as the user name to connect to the AWS Transfer Family IP address.
	
	`SH cp "//'&USRPFX..DB2.UNLOAD.&JOBNAME'" &TABNAME..csv;`
	`echo "ascii " >> uplcmd;`
	`echo "PUT &TABNAME..csv " >>>> uplcmd;`
	`sftp -b uplcmd -i .ssh/id_rsa &FTPUSER.@&FTP_TF_SITE;`
	`rm &TABNAME..csv;`
		Mainframe developer, System engineer

Task	Description	Skills required
Submit the LODnnnnn JCLs.	The prior JCL has generated all LODnnnnn JCL tables that need to be unloaded, transformed into CSV, and transferred to the S3 bucket. Run the `submit` command on all the JCLs that have been generated.	Mainframe developer, System engineer

Related resources

For more information about the different tools and solutions used in this document, see the following:

- z/OS OpenSSH User's Guide
- Db2 z/OS – Sample UNLOAD control statements
- Db2 z/OS – Unloading delimited files
- Transfer Family – Create an SFTP-enabled server
- Transfer Family – Working with service-managed users

Additional information

After you have your Db2 data on Amazon S3, you have many ways to develop new insights. Because Amazon S3 integrates with AWS data analytics services, you can freely consume or expose this data on the distributed side. For example, you can do the following:

- Build a data lake on Amazon S3, and extract valuable insights by using query-in-place, analytics, and machine learning tools without moving the data.
- Initiate a Lambda function by setting up a post-upload processing workflow that is integrated with AWS Transfer Family.
- Develop new microservices for accessing the data in Amazon S3 or in fully managed database by using AWS Glue, which is a serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development.

In a migration use case, because you can transfer any data from the mainframe to S3, you can do the following:

- Retire physical infrastructure, and create a cost-effective data archival strategy with Amazon S3 Glacier and S3 Glacier Deep Archive.
- Build scalable, durable, and secure backup and restore solutions with Amazon S3 and other AWS services, such as S3 Glacier and Amazon Elastic File System (Amazon EFS), to augment or replace existing on-premises capabilities.

More patterns

- Deploy the Security Automations for AWS WAF solution by using Terraform
- Replicate mainframe databases to AWS by using Precisely Connect

Management

Topics

- Cost management
- End-user computing
- High-performance computing
- Hybrid cloud
- Management & governance
- Messaging & communications

Cost management

Topics

- Create detailed cost and usage reports for AWS Glue jobs by using AWS Cost Explorer
- Create detailed cost and usage reports for Amazon EMR clusters by using AWS Cost Explorer
- More patterns

Create detailed cost and usage reports for AWS Glue jobs by using AWS Cost Explorer

Parijat Bhide and Aromal Raj Jayarajan, Amazon Web Services

Summary

This pattern shows how to track the usage costs of AWS Glue data integration jobs by configuring user-defined cost allocation tags. You can use these tags to create detailed cost and usage reports in AWS Cost Explorer for jobs across multiple dimensions. For example, you can track usage costs at the team, project, or cost center level.

Prerequisites and limitations

Prerequisites

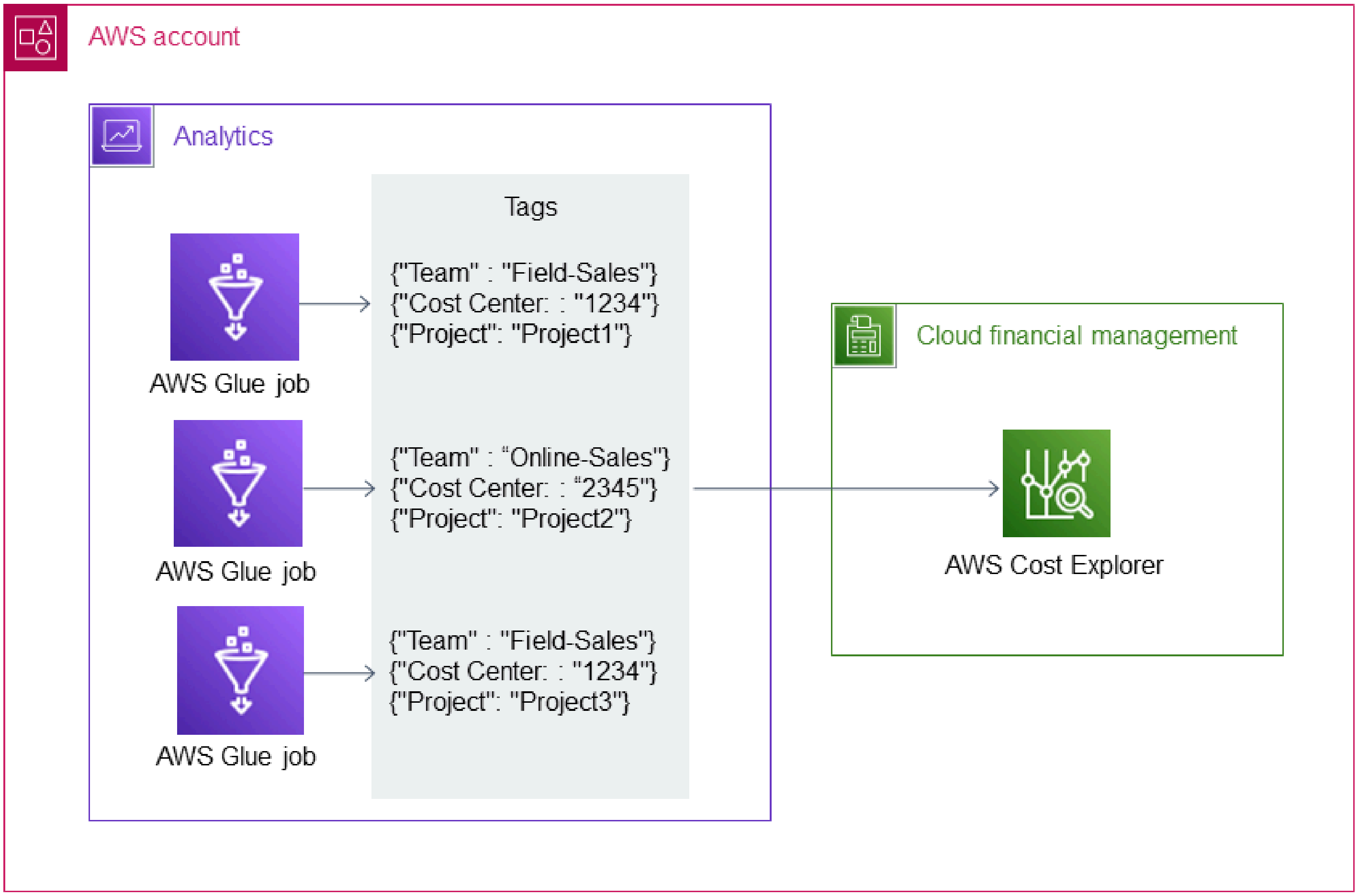
- An active AWS account
- One or more AWS Glue jobs that have user-defined tags activated

Architecture

Target technology stack

- AWS Glue
- AWS Cost Explorer

The following diagram shows how you can apply tags to track usage costs for AWS Glue jobs.



The diagram shows the following workflow:

1. A data engineer or AWS administrator creates user-defined cost allocation tags for the AWS Glue jobs.
2. An AWS administrator activates the tags.
3. The tags report metadata to AWS Cost Explorer.

Tools

- AWS Glue is a fully managed extract, transform, and load (ETL) service. It helps you reliably categorize, clean, enrich, and move data between data stores and data streams.
- AWS Cost Explorer helps you view and analyze your AWS costs and usage.

Epics

Create and activate tags for your AWS Glue jobs

Task	Description	Skills required
Create user-defined cost allocation tags for your AWS Glue jobs.	**To add tags to an existing AWS Glue job** 1. Sign in to the AWS Management console, and then open the [AWS Glue console](https://console.aws.amazon.com/glue/). 2. In the left navigation pane, under **ETL**, choose **Jobs**. 3. In the **Your jobs** section, choose the name of the job that you're tagging. 4. Choose the **Job details** tab. Then, expand the **Advanced properties** section. 5. For **Tags**, choose **Add new tag**. 6. For **Key**, enter a name for your tag. 7. (Optional) For **Value**, enter a value that you want associated with the key. 8. (Optional) Repeat steps 5-7 for each tag that you want to create for the job. 9. Choose **Save**.	Data engineer

Task	Description	Skills required
	To add tags to a new AWS Glue job 1. Create a new AWS Glue job based on your use case requirements. For instructions, see [Working with jobs on the AWS Glue Console]() in the *AWS Glue Developer Guide*. 2. When you configure the **Job details** settings, follow steps 4-9 of the **To add tags to an existing AWS Glue job** section of this task. **Note** For more information, see [AWS tags in AWS Glue]() in the *AWS Glue Developer Guide.*	
Activate the user-defined cost allocation tags.	Follow the instructions in [Activating user-defined cost allocation tags]() in the *AWS Billing User Guide*.	AWS administrator

Create cost and usage reports for your AWS Glue jobs

Task	Description	Skills required
Create cost and usage reports for your AWS Glue jobs by using tag filters in AWS Cost Explorer.	1. Sign in to the AWS Management Console and open the [AWS Cost Management console](https://console.aws.amazon.com/cost-management/home). 2. In the left navigation pane, choose **Reports**. 3. Choose **Create new report**. 4. For **Select a report type**, choose **Cost and usage (recommended)**. Then, choose **Create Report**. 5. For **Filters**, choose **Service**. The **Service** dropdown appears. 6. Select the check boxes next to **Glue**. Then, choose **Apply filters**. 7. For **Filters**, choose **Tag**. The **Tag** dropdown appears. 8. Choose **Team**. Then, select the check boxes next to the teams that you've assigned tags to. Exclude any teams that you haven't assigned tags to. Then, choose **Apply filters**. 9. At the top of the chart, choose **Tag**. Then, choose the tags for the AWS Glue	General AWS, AWS administrator

Task	Description	Skills required
	jobs that you want to create a report for. 10. At the top of the chart, choose the **Last 3 Months** dropdown and choose the timeframe that you want the report to cover. Then, choose the **Monthly** dropdown and choose how you want the line items in the report to be aggregated based on timeframe. 11. Choose **Save as**. Then, enter a title for your report. 12. Choose **Save Report**. For more information, see Exploring your data using Cost Explorer in the *AWS Cost Management User Guide*.	

Create detailed cost and usage reports for Amazon EMR clusters by using AWS Cost Explorer

Parijat Bhide and Aromal Raj Jayarajan, Amazon Web Services

Summary

This pattern shows how to track the usage costs of Amazon EMR clusters by configuring user-defined cost allocation tags. You can use these tags to create detailed cost and usage reports in AWS Cost Explorer for clusters across multiple dimensions. For example, you can track usage costs at the team, project, or cost center level.

Prerequisites and limitations

Prerequisites

- An active AWS account
- One or more EMR clusters that have user-defined tags activated

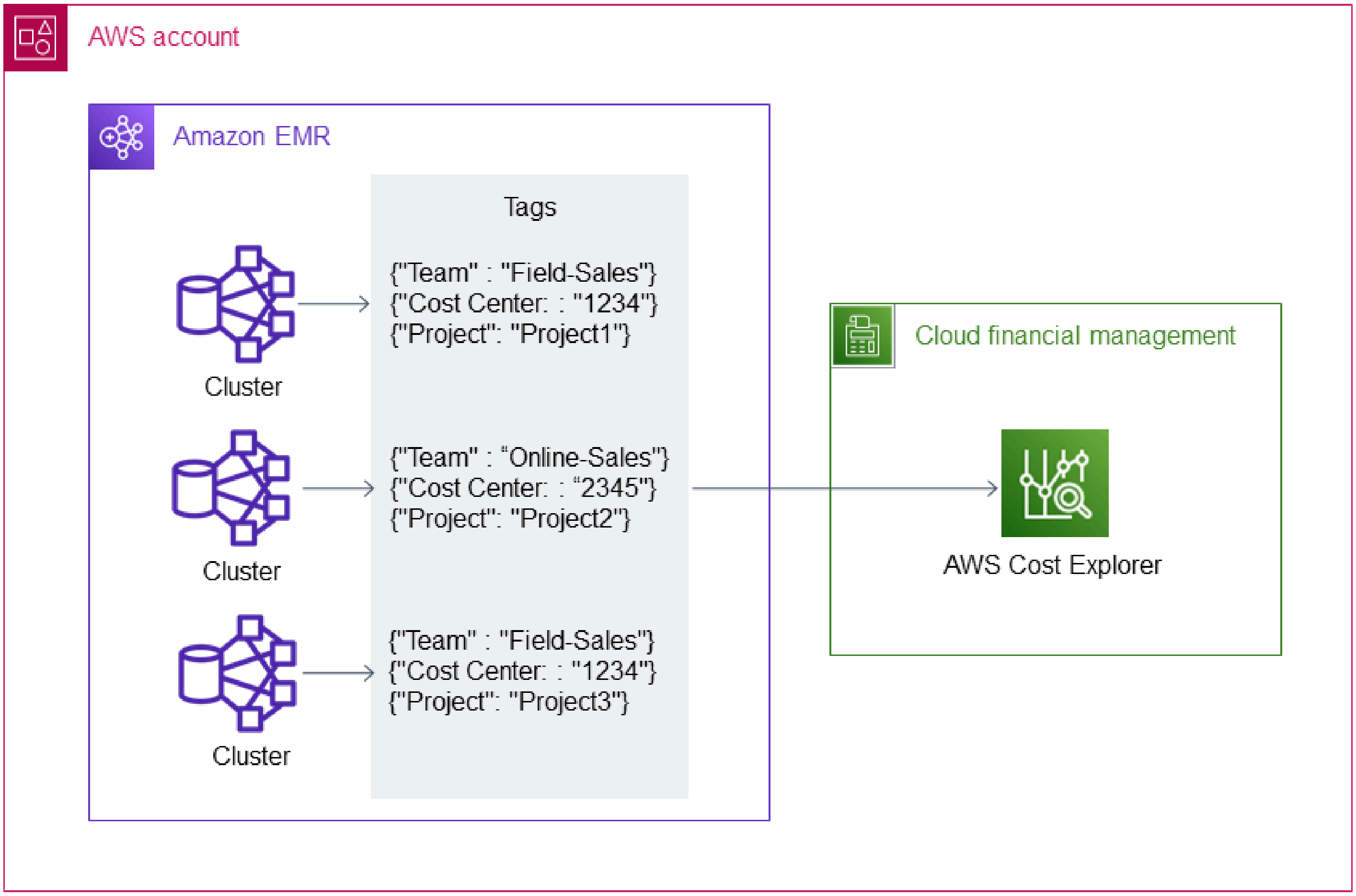
Architecture

Target technology stack

- Amazon EMR
- AWS Cost Explorer

Target architecture

The following diagram shows how you can apply tags to track usage costs for specific Amazon EMR clusters.



The diagram shows the following workflow:

1. A data engineer or AWS administrator creates user-defined cost allocation tags for the Amazon EMR clusters.
2. An AWS administrator activates the tags.
3. The tags report metadata to AWS Cost Explorer.

Tools

Tools

- Amazon EMR is a managed cluster platform that simplifies running big data frameworks on AWS to process and analyze large amounts of data.
- AWS Cost Explorer helps you view and analyze your AWS costs and usage.

Epics

Create and activate tags for your Amazon EMR clusters

Task	Description	Skills required
Create user-defined cost allocation tags for your Amazon EMR clusters.	**To add tags to an existing Amazon EMR cluster** Follow the instructions in Adding tags to an existing cluster in the *Amazon EMR Management Guide*. **To add tags to a new Amazon EMR cluster** Follow the instructions in Add tags to a new cluster in the *Amazon EMR Management Guide*. For more information about how to set up an Amazon EMR cluster, see Plan and configure clusters in the *Amazon EMR Management Guide*.	Data engineer
Activate the user-defined cost allocation tags.	Follow the instructions in Activating user-defined cost allocation tags in the *AWS Billing User Guide*.	AWS administrator

Create cost and usage reports for your Amazon EMR clusters

Task	Description	Skills required
Create cost and usage reports for your Amazon EMR clusters by using tag filters in AWS Cost Explorer.	1. Sign in to the AWS Management Console and open the [AWS Cost Management console](https://console.aws.amazon.com/cost-management/home). 2. In the left navigation pane, choose **Reports**. 3. Choose **Create new report**. 4. For **Select a report type**, choose **Cost and usage (recommended)**. Then, choose **Create Report**. 5. For **Filters**, choose **Service**. The **Service** dropdown appears. 6. Select the check boxes next to **EMR (Elastic MapReduce)** and **EC2-Instances (Elastic Compute Cloud – Compute)**. Then, choose **Apply filters**. 7. For **Filters**, choose **Tag**. The **Tag** dropdown appears. 8. Choose **Team**. Then, select the check boxes next to the teams that you've assigned tags to. Exclude any teams that you haven't assigned tags to. Then, choose **Apply filters**.	General AWS, AWS administrator

Task	Description	Skills required
	9. At the top of the chart, choose **Tag**. Then, choose the tags for the Amazon EMR clusters that you want to create a report for. 10. At the top of the chart, choose the **Last 3 Months** dropdown and choose the timeframe that you want the report to cover. Then, choose the **Monthly** dropdown and choose how you want the line items in the report to be aggregate d based on timeframe. 11. Choose **Save as**. Then, enter a title for your report. 12. Choose **Save Report**. For more information, see Exploring your data using Cost Explorer in the *AWS Cost Management User Guide*.	

More patterns

- Automate AWS infrastructure operations by using Amazon Bedrock
- Automatically inventory AWS resources across multiple accounts and Regions
- Automate the creation of AppStream 2.0 resources using AWS CloudFormation
- Automatically archive items to Amazon S3 using DynamoDB TTL
- Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows
- Create detailed cost and usage reports for Amazon RDS and Amazon Aurora
- Delete unused Amazon EBS volumes by using AWS Config and AWS Systems Manager
- Estimate storage costs for an Amazon DynamoDB table
- Estimate the cost of a DynamoDB table for on-demand capacity
- Set up event-driven auto scaling in Amazon EKS by using Amazon EKS Pod Identity and KEDA
- Coordinate resource dependency and task execution by using the AWS Fargate WaitCondition hook construct

End-user computing

Topics

- Implement SAML 2.0 authentication for Amazon WorkSpaces by using Auth0 and AWS Managed Microsoft AD
- More patterns

Implement SAML 2.0 authentication for Amazon WorkSpaces by using Auth0 and AWS Managed Microsoft AD

Siva Vinnakota and Shantanu Padhye, Amazon Web Services

Summary

This pattern explores how you can integrate Auth0 with AWS Directory Service for Microsoft Active Directory to create a robust SAML 2.0 authentication solution for your Amazon WorkSpaces environment. It explains how to establish federation between these AWS services to enable advanced features such as multi-factor authentication (MFA) and custom login flows while preserving seamless desktop access through AWS Managed Microsoft AD. Whether you're managing only a handful of users or thousands, this integration helps provide flexibility and security for your organization. This pattern provides the steps for the setup process so you can implement this solution in your own environment.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Managed Microsoft AD
- A provisioned desktop in Amazon WorkSpaces Personal that is associated with AWS Managed Microsoft AD
- An Amazon Elastic Compute Cloud (Amazon EC2) instance
- An Auth0 account

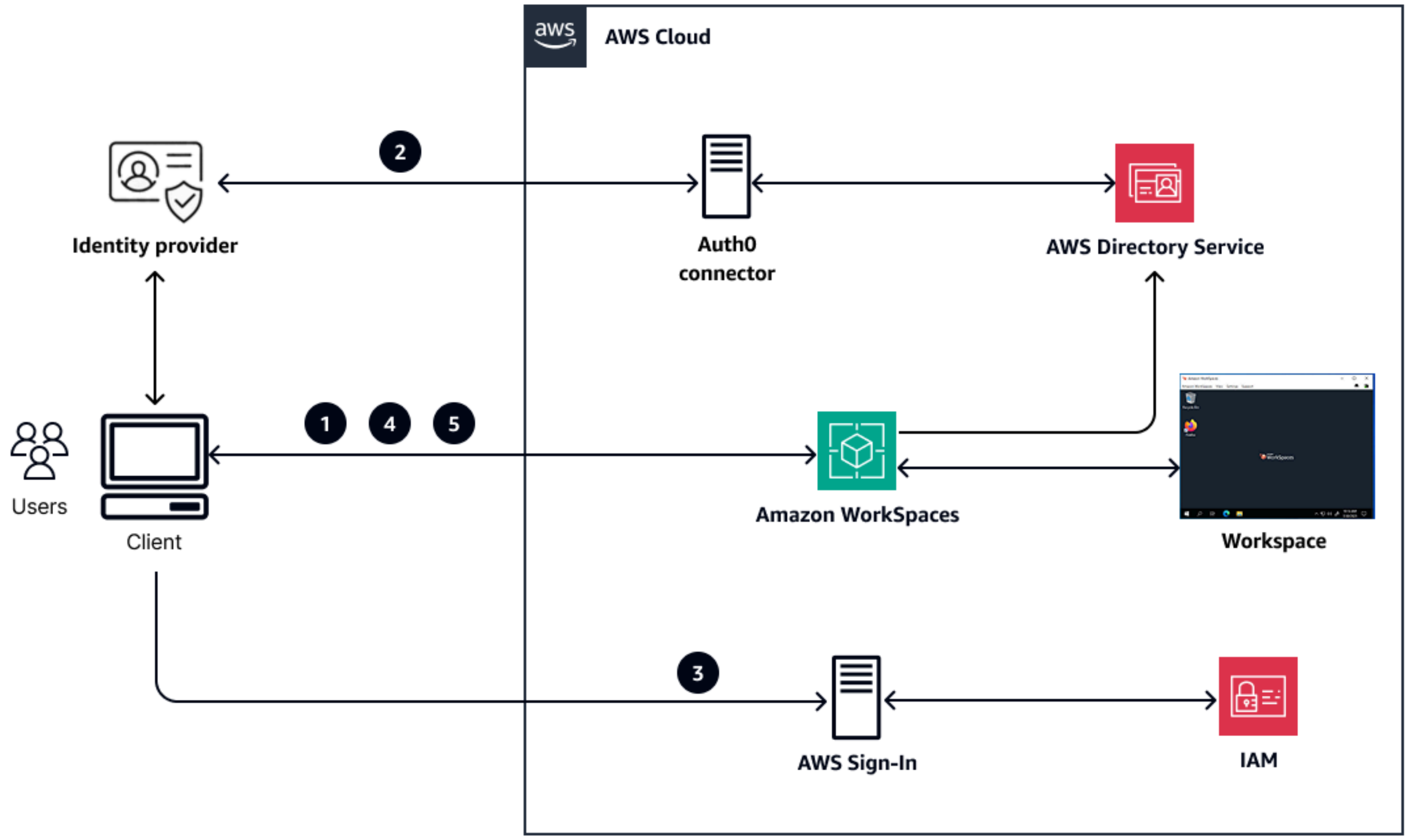
Limitations

Some AWS services aren’t available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

The SAML 2.0 authentication process for a WorkSpaces client application consists of five steps that are illustrated in the following diagram. These steps represent a typical workflow for logging in.

You can use this distributed approach to authentication after you follow the instructions in this pattern, to help provide a structured and secure method for user access.



Workflow:

1. **Registration**. The user launches the client application for WorkSpaces and enters the WorkSpaces registration code for their SAML-enabled WorkSpaces directory. WorkSpaces returns the Auth0 identity provider (IdP) URL to the client application.
2. **Login.** The WorkSpaces client redirects to the user's web browser by using the Auth0 URL. The user authenticates with their username and password. Auth0 returns a SAML assertion to the client browser. The SAML assertion is an encrypted token that asserts the user's identity.
3. **Authenticate**. The client browser posts the SAML assertion to the AWS Sign-In endpoint to validate it. AWS Sign-In allows the caller to assume an AWS Identity and Access Management (IAM) role. This returns a token that contains temporary credentials for the IAM role.
4. **WorkSpaces login**. The WorkSpaces client presents the token to the WorkSpaces service endpoint. WorkSpaces exchanges the token for a session token and returns the session token to the WorkSpaces client with a login URL. When the WorkSpaces client loads the login page. the username value is populated by the `NameId` value that's passed in the SAML response.

5. **Streaming**. The user enters their password and authenticates against the WorkSpaces directory. After authentication, WorkSpaces returns a token to the client. The client redirects back to the WorkSpaces service and presents the token. This brokers a streaming session between the WorkSpaces client and the WorkSpace.

> **Note**
>
> To set up a seamless single sign-on experience that doesn't require a password prompt, see the Certificate-based authentication and WorkSpaces Personal in the WorkSpaces documentation.

Tools

AWS services

- Amazon WorkSpaces is a fully managed virtual desktop infrastructure (VDI) service that provides users with cloud-based desktops without having to procure and deploy hardware or install complex software.
- AWS Directory Service for Microsoft Active Directory enables your directory-aware workloads and AWS resources to use Microsoft Active Directory in the AWS Cloud.

Other tools

- Auth0 is an authentication and authorization platform that helps you manage access to your applications.

Epics

Configure the Active Directory LDAP Connector in Auth0

Task	Description	Skills required
Install the Active Directory LDAP connector in Auth0 with AWS Managed Microsoft AD.	1. Log in to the Auth0 dashboard and choose **Authentication**, **Enterprise**, **Active Directory / LDAP**.	Cloud administrator, Cloud architect

Task	Description	Skills required
	Choose **Create a connection**. 2. Provide the name for your Active Directory connection, and then choose **Create**. 3. On the **Setup** tab, download the agent for your operating system. When the installation is complete, your default browser displays a prompt for the ticket URL. 4. Enter the provisioning ticket URL, which should be unique for your connection string, and then choose **Continue**. 5. In the **AD LDAP Configuration** dialog box, for **Username** and **Password**, enter your administrator credentials, and then choose **Save**. When the connection to Auth0 has been established, the configuration log is displayed with a reporting status of **OK** for all checks.	

Task	Description	Skills required

Create an application in Auth0 to generate the SAML metadata manifest file.	1. Log in to the Auth0 dashboard and create a new application by following the instructions in the [Auth0 documentation]().
	 2. On the Auth0 dashboard, choose the application name to access its configuration settings. On the **AddOns** tab, choose **SAML2 Web App**.
	 3. On the **Settings** tab for the add-on, for **Application Callback URL**, enter **https://signin.aws.amazon.com/saml**. This is where the SAML token will send the POST request.
	 4. On the **Settings** tab, in the **Settings** box, paste the following SAML configuration code in JSON format:
	 `{`
	 `  "audience": "https://signin.aws.amazon.com/saml",`
	 `  "mappings": {`
	 `    "email": "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/sAMAccountName",`
	 `    "name": "http://schemas.xmlsoap.org/`
		Cloud administrator, Cloud architect

Task	Description	Skills required

	``
	ws/2005/05/identity/
	claims/name"
	  },
	  "createUpnClaim":
	 false,
	  "passthroughClaims
	WithNoMapping":
	 false,
	  "mapUnknownClaimsA
	sIs": false,
	  "mapIdentities":
	 false,
	  "nameIdentifierFor
	mat": "urn:oasi
	s:names:tc:SAML:2.
	0:nameid-format:pe
	rsistent",
	  "nameIdentifierPro
	bes": [
	    "http://s
	chemas.auth0.com/s
	AMAccountName"
	  ]
	}
	``
	5. Save, and then choose **Enable**.
	6. Choose the **Usage** tab, and download the metadata manifest file for the identity provider. This information is required for the next steps.
	7. Close the **SAML2 Web App** window.
	8. On the applications screen, choose **Connections**. Under **Enterprise**, select the correct Active Directory	

Task	Description	Skills required
	/LDAP connector and enable it.	

Set up IdP, role, and policy for SAML 2.0 in IAM

Task	Description	Skills required
Create a SAML 2.0 IdP in IAM.	To set up SAML 2.0 as an IdP, follow the steps that are outlined in [Create a SAML identity provider in IAM](#) in the IAM documentation.	Cloud administrator
Create an IAM role and policy for SAML 2.0 federation.	1. Create an IAM role for SAML 2.0 federation. For instructions, see [step 2](#) in the instructions for setting up SAML 2.0 for WorkSpaces Personal in the WorkSpaces documentation. 2. Create an IAM policy and associate it with the role you created in the previous step. For instructions, see [step 3](#) in the instructions for setting up SAML 2.0 for WorkSpaces Personal in the WorkSpaces documentation.	Cloud administrator

Configure assertions in Auth0

Task	Description	Skills required
Configure Auth0 and SAML assertions.	You can use Auth0 actions to configure assertions in SAML 2.0 responses. A SAML assertion is an encrypted token that asserts the user's identity. 1. Sign in to the Auth0 dashboard. Choose **Actions**, **Library**, **Create Action**, **Build from scratch**. 2. Provide the following values and then choose **Create**. **Name: Specify a name for the action** **Trigger: Choose Login/Post Login** **Runtime: Choose Node 18** 3. On the next screen, enter the following code: `exports.onExecutePostLogin = async (event, api) => {   if (event.client.name === "Workspace_Saml") {     const awsRole = 'arn:aws:iam::030784294031:role/Workspace_Auth0,arn:aw`	Cloud administrator

Task	Description	Skills required

	``
	s:iam::03078429403
	1:saml-provider/Au
	th0';
	     const awsRoleSe
	ssion = event.use
	r.sAMAccountName;
	     const email =
	 event.user.emails[
	0];
	     api.samlR
	esponse.setDestina
	tion('https://sign
	in.aws.amazon.com/
	saml');
	     api.samlR
	esponse.setAttribu
	te('https://aws.am
	azon.com/SAML/
	Attributes/Role',
	 awsRole)
	     api.samlR
	esponse.setAttribu
	te('https://aws.am
	azon.com/SAML/Attr
	ibutes/RoleSession
	Name', awsRoleSe
	ssion)
	     api.samlR
	esponse.setAttribu
	te('https://aws.am
	azon.com/SAML/Attr
	ibutes/PrincipalTa
	g:Email', email)
	   } return;
	};
	``
	4. Choose **Deploy**.
	
	This completes the setup of SAML 2.0 authentication for WorkSpaces Personal	

Task	Description	Skills required
	desktops. The Architect ure section illustrates the authentication process after setup.	

Troubleshooting

Issue	Solution
SAML 2.0 authentication issues in WorkSpaces	If you encounter any issues when you implement SAML 2.0 authentication for WorkSpaces Personal, follow the steps and links outlined in the AWS re:Post article on troubleshooting SAML 2.0 authentication. For additional information about investigating SAML 2.0 errors while accessing WorkSpaces, see: • How do I create a HAR file and Console logs from my browser for an AWS Support case? (AWS re:Post) • View a SAML response in your browser (IAM documentation)

Related resources

- Set up SAML 2.0 for WorkSpaces Personal (WorkSpaces documentation)
- Auth0 documentation

More patterns

- Automate the creation of AppStream 2.0 resources using AWS CloudFormation
- Connect to an Amazon EC2 instance by using Session Manager
- Improve call quality on agent workstations in Amazon Connect contact centers
- Run AWS Systems Manager Automation tasks synchronously from AWS Step Functions

High-performance computing

Topics

- Deploy a Lustre file system for high-performance data processing by using Terraform and DRA
- Set up a Grafana monitoring dashboard for AWS ParallelCluster
- More patterns

Deploy a Lustre file system for high-performance data processing by using Terraform and DRA

Arun Bagal and Ishwar Chauthaiwale, Amazon Web Services

Summary

This pattern automatically deploys a Lustre file system on AWS and integrates it with Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3).

This solution helps you quickly set up a high performance computing (HPC) environment with integrated storage, compute resources, and Amazon S3 data access. It combines Lustre's storage capabilities with the flexible compute options provided by Amazon EC2 and the scalable object storage in Amazon S3, so you can tackle data-intensive workloads in machine learning, HPC, and big data analytics.

The pattern uses a HashiCorp Terraform module and Amazon FSx for Lustre to streamline the following process:

- Provisioning a Lustre file system
- Establishing a data repository association (DRA) between FSx for Lustre and an S3 bucket to link the Lustre file system with Amazon S3 objects
- Creating an EC2 instance
- Mounting the Lustre file system with the Amazon S3-linked DRA on the EC2 instance

The benefits of this solution include:

- Modular design. You can easily maintain and update the individual components of this solution.
- Scalability. You can quickly deploy consistent environments across AWS accounts or Regions.
- Flexibility. You can customize the deployment to fit your specific needs.
- Best practices. This pattern uses preconfigured modules that follow AWS best practices.

For more information about Lustre file systems, see the Lustre website.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A least privilege AWS Identity and Access Management (IAM) policy (see instructions)

Limitations

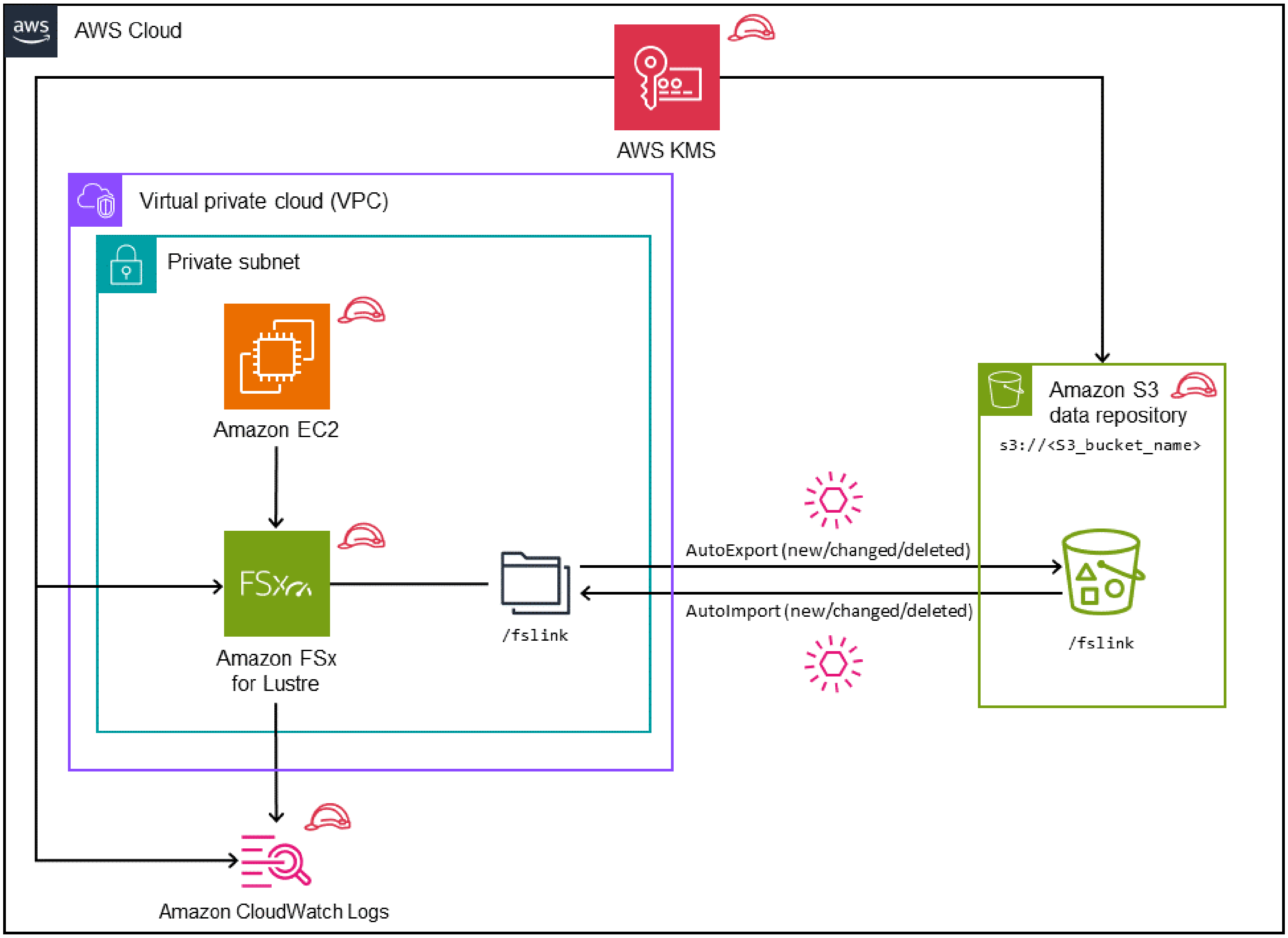
FSx for Lustre limits the Lustre file system to a single Availability Zone, which could be a concern if you have high availability requirements. If the Availability Zone that contains the file system fails, access to the file system is lost until recovery. To achieve high availability, you can use DRA to link the Lustre file system with Amazon S3, and transfer data between Availability Zones.

Product versions

- Terraform version 1.9.3 or later
- HashiCorp AWS Provider version 4.0.0 or later

Architecture

The following diagram shows the architecture for FSx for Lustre and complementary AWS services in the AWS Cloud.



The architecture includes the following:

- An S3 bucket is used as a durable, scalable, and cost-effective storage location for data. The integration between FSx for Lustre and Amazon S3 provides a high-performance file system that is seamlessly linked with Amazon S3.
- FSx for Lustre runs and manages the Lustre file system.
- Amazon CloudWatch Logs collects and monitors log data from the file system. These logs provide insights into the performance, health, and activity of your Lustre file system.
- Amazon EC2 is used to access Lustre file systems by using the open source Lustre client. EC2 instances can access file systems from other Availability Zones within the same virtual private cloud (VPC). The networking configuration allows for access across subnets within the VPC. After the Lustre file system is mounted on the instance, you can work with its files and directories just as you would use a local file system.

- AWS Key Management Service (AWS KMS) enhances the security of the file system by providing encryption for data at rest.

Automation and scale

Terraform makes it easier to deploy, manage, and scale your Lustre file systems across multiple environments. In FSx for Lustre, a single file system has size limitations, so you might need to horizontally scale by creating multiple file systems. You can use Terraform to provision multiple Lustre file systems based on your workload needs.

Tools

AWS services

- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon FSx for Lustre makes it easy and cost-effective to launch, run, and scale a high-performance Lustre file system.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code repository

The code for this pattern is available in the GitHub Provision FSx for Lustre Filesystem using Terraform repository.

Best practices

- The following variables define the Lustre file system. Make sure to configure these correctly based on your environment, as instructed in the Epics section.
  - `storage_capacity` – The storage capacity of the Lustre file system, in GiBs. The minimum and default setting is 1200 GiB.

- deployment_type – The deployment type for the Lustre file system. For an explanation of the two options, PERSISTENT_1 and PERSISTENT_2 (default), see the FSx for Lustre documentation.
- per_unit_storage_throughput – The read and write throughput, in MBs per second per TiB.
- subnet_id – The ID of the private subnet where you want to deploy FSx for Lustre.
- vpc_id – The ID of your virtual private cloud on AWS where you want to deploy FSx for Lustre.
- data_repository_path – The path to the S3 bucket that will be linked to the Lustre file system.
- iam_instance_profile – The IAM instance profile to use to launch the EC2 instance.
- kms_key_id – The Amazon Resource Name (ARN) of the AWS KMS key that will be used for data encryption.

- Ensure proper network access and placement within the VPC by using the security_group and vpc_id variables.
- Run the terraform plan command as described in the Epics section to preview and verify changes before applying them. This helps catch potential issues and ensures that you are aware of what will be deployed.
- Use the terraform validate command as described in the Epics section to check for syntax errors and to confirm that your configuration is correct.

Epics

Set up your environment

Task	Description	Skills required
Install Terraform.	To install Terraform on your local machine, follow the instructions in the Terraform documentation.	AWS DevOps, DevOps engineer
Set up AWS credentials.	To set up the AWS Command Line Interface (AWS CLI) profile for the account, follow	AWS DevOps, DevOps engineer

Task	Description	Skills required
	the instructions in the [AWS documentation]().	
Clone the GitHub repository.	To clone the GitHub repositor y, run the command: `git clone https://github.com/aws-samples/provision-fsx-lustre-with-terraform.git`	AWS DevOps, DevOps engineer

Configure and deploy FSx for Lustre

Task	Description	Skills required
Update the deployment configuration.	1. In the cloned repositor y on your local machine, navigate to the `fsx_deployment` directory: `cd fsx_deployment` 2. Open the `terraform .tfvars` file, and update the values of the following variables: • `vpc_id` • `subnet_id` • `data_repository_path` • `iam_instance_profile`	AWS DevOps, DevOps engineer

Task	Description	Skills required
	• `kms_key_id` For descriptions of these variables, see the Best practices section. 3. In the same directory, open the `locals.tf` file and update the CIDR ranges for the `fsx_inress` and `fsx_egress` security group variables. 4. If needed, open the `variables.tf` file and update the default values of these variables: • `storage_capacity` • `deployment_type` • `per_unit_storage_throughput` For descriptions of these variables, see the Best practices section.	
Initialize the Terraform environment.	To initialize your environment to run the Terraform `fsx_deployment` module, run: `terraform init`	AWS DevOps, DevOps engineer

Task	Description	Skills required
Validate the Terraform syntax.	To check for syntax errors and to confirm that your configuration is correct, run: `terraform validate`	AWS DevOps, DevOps engineer
Validate the Terraform configuration.	To create a Terraform execution plan and preview the deployment, run: `terraform plan -var-file terraform.tfvars`	AWS DevOps, DevOps engineer
Deploy the Terraform module.	To deploy the FSx for Lustre resources, run: `terraform apply -var-file terraform.tfvars`	AWS DevOps, DevOps engineer

Clean up AWS resources

Task	Description	Skills required
Remove AWS resources.	After you finish using your FSx for Lustre environment, you can remove the AWS resources deployed by Terraform to avoid incurring unnecessary charges. The Terraform module provided in the code repository automates this cleanup.	AWS DevOps, DevOps engineer

Task	Description	Skills required

	1. In your local repository, navigate to the `fsx_deployment` directory:
	`cd fsx_deployment`
	2. Run the command:
	`terraform destroy -var-file terraform.tfvars`	

Troubleshooting

Issue	Solution
FSx for Lustre returns errors.	For help with FSx for Lustre issues, see Troubleshooting Amazon FSx for Lustre in the FSx for Lustre documentation.

Related resources

- Building Amazon FSx for Lustre by using Terraform (AWS Provider reference in the Terraform documentation)
- Getting started with Amazon FSx for Lustre (FSx for Lustre documentation)
- AWS blog posts about Amazon FSx for Lustre

Set up a Grafana monitoring dashboard for AWS ParallelCluster

Dario La Porta and William Lu, Amazon Web Services

Summary

AWS ParallelCluster helps you deploy and manage high performance computing (HPC) clusters. It supports AWS Batch and Slurm open source job schedulers. Although AWS ParallelCluster is integrated with Amazon CloudWatch for logging and metrics, it doesn't provide a monitoring dashboard for the workload.

The Grafana dashboard for AWS ParallelCluster (GitHub) is a monitoring dashboard for AWS ParallelCluster. It provides job scheduler insights and detailed monitoring metrics at the operating system (OS) level. For more information about the dashboards included in this solution, see Example Dashboards in the GitHub repository. These metrics help you better understand the HPC workload and its performance. However, the dashboard code is not updated for the latest versions of AWS ParallelCluster or the open source packages that are used in solution. This pattern enhances the solution to provide the following benefits:

- Supports AWS ParallelCluster v3
- Uses the latest version of open source packages, including Prometheus, Grafana, Prometheus Slurm Exporter, and NVIDIA DCGM-Exporter
- Increases the number of CPU cores and GPUs that the Slurm jobs use
- Adds a job monitoring dashboard
- Enhances the GPU node monitoring dashboard for nodes with 4 or 8 graphics processing units (GPUs)

This version of the enhanced solution has been implemented and verified in an AWS customer's HPC production environment.

Prerequisites and limitations

Prerequisites

- AWS ParallelCluster CLI, installed and configured.
- A supported network configuration for AWS ParallelCluster. This pattern uses the AWS ParallelCluster using two subnets configuration, which requires a public subnet, private subnet, internet gateway, and NAT gateway.

- All AWS ParallelCluster cluster nodes must have internet access. This is required so that the installation scripts can download the open source software and Docker images.
- A key pair in Amazon Elastic Compute Cloud (Amazon EC2). Resources that have this key pair have Secure Shell (SSH) access to the head node.

Limitations

- This pattern is designed to support Ubuntu 20.04 LTS. If you're using a different version of Ubuntu or if you use Amazon Linux or CentOS, then you need to modify the scripts provided with this solution. These modifications are not included in this pattern.

Product versions

- Ubuntu 20.04 LTS
- ParallelCluster 3.X

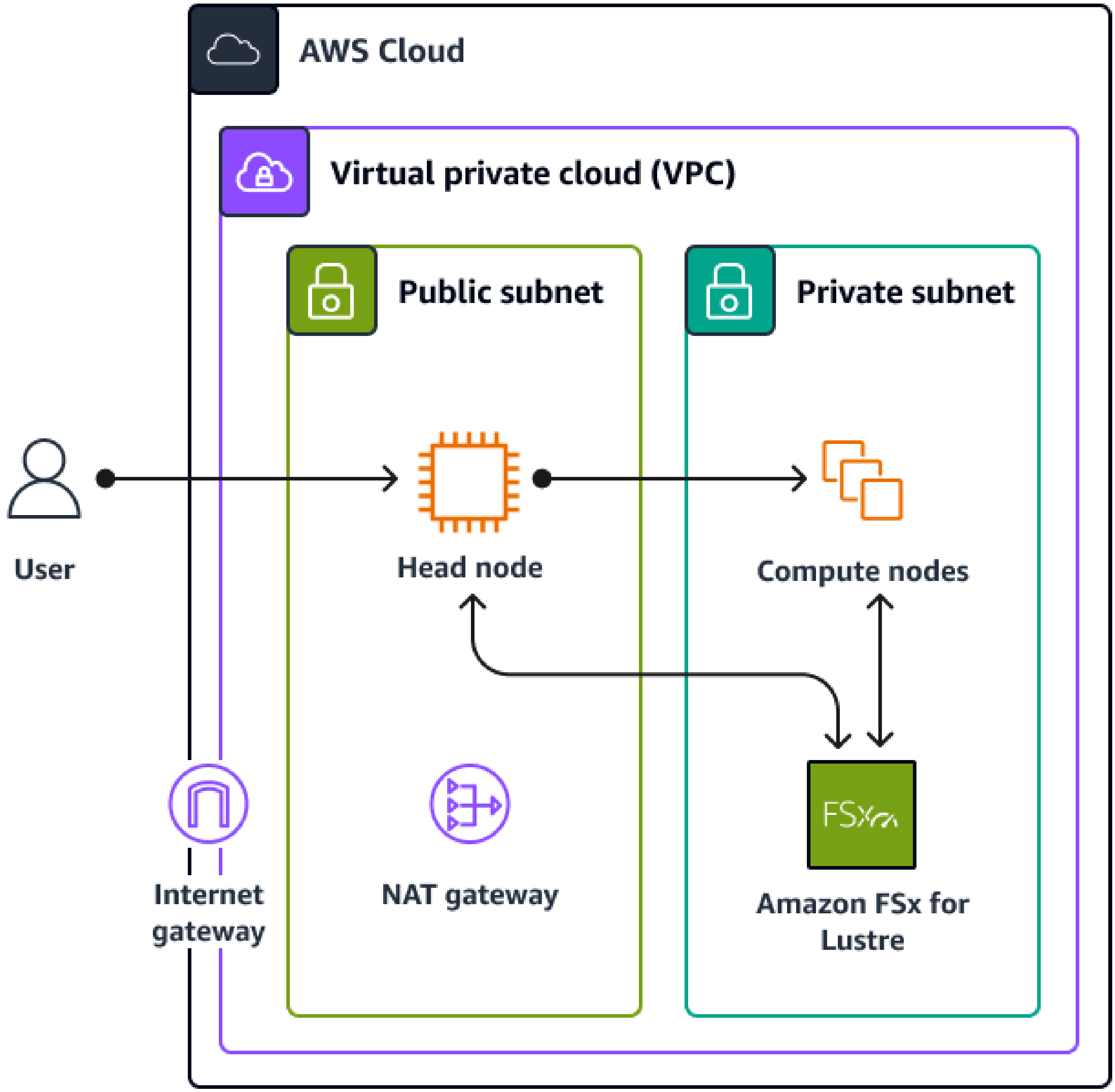
Billing and cost considerations

- The solution deployed in this pattern is not covered by the free tier. Charges apply for Amazon EC2, Amazon FSx for Lustre, the NAT gateway in Amazon VPC, and Amazon Route 53.

Architecture

Target architecture

The following diagram shows how a user can access the monitoring dashboard for AWS ParallelCluster on the head node. The head node runs NICE DCV, Prometheus, Grafana, Prometheus Slurm Exporter, Prometheus Node Exporter, and NGINX Open Source. The compute nodes run Prometheus Node Exporter, and they also run NVIDIA DCGM-Exporter if the node contains GPUs. The head node retrieves information from the compute nodes and displays that data in the Grafana dashboard.



In most cases, the head node is not heavily loaded because the job scheduler doesn't require a significant amount of CPU or memory. Users access the dashboard on the head node by using SSL on port 443.

All authorized viewers can anonymously view the monitoring dashboards. Only the Grafana administrator can modify dashboards. You configure a password for the Grafana administrator in the `aws-parallelcluster-monitoring/docker-compose/docker-compose.head.yml` file.

Tools

AWS services

- NICE DCV is a high-performance remote display protocol that helps you deliver remote desktops and application streaming from any cloud or data center to any device, over varying network conditions.
- AWS ParallelCluster helps you deploy and manage high performance computing (HPC) clusters. It supports AWS Batch and Slurm open source job schedulers.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined.

Other tools

- Docker is a set of platform as a service (PaaS) products that use virtualization at the operating-system level to deliver software in containers.
- Grafana is an open source software that helps you query, visualize, alert on, and explore metrics, logs, and traces.
- NGINX Open Source is an open source web server and reverse proxy.
- NVIDIA Data Center GPU Manager (DCGM) is a suite of tools for managing and monitoring NVIDIA data center graphics processing units (GPUs) in cluster environments. In this pattern, you use DCGM-Exporter, which helps you export GPU metrics from Prometheus.
- Prometheus is an open source system-monitoring toolkit that collects and stores its metrics as time-series data with associated key-value pairs, which are called *labels*. In this pattern, you also use Prometheus Slurm Exporter to collect and export metrics, and you use Prometheus Node Exporter to export metrics from the compute nodes.
- Ubuntu is an open source, Linux-based operating system that is designed for enterprise servers, desktops, cloud environments, and IoT.

Code repository

The code for this pattern is available in the GitHub [pcluster-monitoring-dashboard](https://github.com/aws-samples/parallelcluster-monitoring-dashboard) repository.

Epics

Create the required resources

Task	Description	Skills required
Create an S3 bucket.	Create an Amazon S3 bucket. You use this bucket to store the configuration scripts. For instructions, see [Creating a bucket](#) in the Amazon S3 documentation.	General AWS
Clone the repository.	Clone the GitHub [pcluster-monitoring-dashboard](#) repo by running the following command. `git clone https://github.com/aws-samples/parallelcluster-monitoring-dashboard.git`	DevOps engineer
Create an admin password.	1. Choose the `aws-parallelcluster-monitoring` folder, choose the `docker-compose` folder, and then open the **docker-compose.head.yml** file. 2. In the `GF_SECURITY_ADMIN_PASSWORD` variable, replace `Grafana4PC!` with a password of your choice. This is the administrative	Linux Shell scripting

Task	Description	Skills required
	password that you use to manage the Grafana account. 3. Save and close the **docker-compose.head.yml** file.	
Copy the required files into the S3 bucket.	Copy the post_install.sh script and the aws-parallelcluster-monitoring folder into the S3 bucket you created. For instructions, see Uploading objects in the Amazon S3 documentation.	General AWS

Task	Description	Skills required
Configure an additional security group for the head node.	1. Create a security group for the head node. This security group will allow inbound traffic to the monitoring dashboards on the head node. For instructions, see [Create a security group](#) in the Amazon VPC documentation. 2. Add an inbound rule to the security group. For instructions, see [Add rules to a security group](#) in the Amazon VPC documentation. Use the following parameters for the rule: • **Type** – HTTPS • **Protocol** – TCP • **Port range** – 443 • **Source** – Enter your IP address • **Description** – Allow users to access the monitoring dashboard	AWS administrator

Task	Description	Skills required
Configure an IAM policy for the head node.	Create an identity-based policy for the head node. This policy allows the node to retrieve metric data from Amazon CloudWatch. The GitHub repo contains an example policy. For instructions, see Creating IAM policies in the AWS Identity and Access Management (IAM) documentation.	AWS administrator
Configure an IAM policy for the compute nodes.	Create an identity-based policy for the compute nodes. This policy allows the node to create the tags that contain the job ID and job owner. The GitHub repo contains an example policy. For instructions, see Creating IAM policies in the IAM documentation. If you use the provided example file, replace the following values: • **<REGION>** – The AWS Region where the cluster is hosted • **<ACCOUNT_ID>** – The AWS account ID	AWS administrator

Create the cluster

Task	Description	Skills required
Modify the provided cluster template file.	Create the AWS ParallelC luster cluster. Use the provided cluster.yaml AWS CloudFormation template file as a starting point to create the cluster. Replace the following values in the provided template: • **<REGION>** – The AWS Region where the cluster is hosted. • **<HEADNODE_SUBNET>** – The public subnet of the VPC. • **<ADDITIONAL_HEAD_N ODE_SG>** – The name of the security group that you created for the head node. • **<KEY_NAME>** – Enter the name of an existing Amazon EC2 key pair. Resources that have this key pair have Secure Shell (SSH) access to the head node. • **<ALLOWED_IPS>** -–Enter the CIDR-formatted IP address range that is allowed to make SSH connections to the head node.	AWS administrator

Task	Description	Skills required
	• **<ADDITIONAL_HEAD_NODE_POLICY>** – Enter the name of the IAM policy that you created for the head node. • **<BUCKET_NAME>** – Enter the name of the S3 bucket you created. • **<COMPUTE_SUBNET>** – Enter the name of the private subnet in the VPC. • **<ADDITIONAL_COMPUTE_NODE_POLICY>** – Enter the name of the IAM policy that you created for the compute node.	
Create the cluster.	In the AWS ParallelCluster CLI, enter the following command. This deploys the CloudFormation template and creates the cluster. For more information about this command, see [pcluster create-cluster](https://docs.aws.amazon.com/parallelcluster/latest/ug/pcluster.create-cluster-v3.html) in the AWS ParallelCluster documentation. `pcluster create-cluster -n <cluster_name> -c cluster.yaml`	AWS administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Monitor the cluster creation.</td><td>Enter the following command to monitor the cluster creation. For more informati on about this command, see pcluster describe-cluster in the AWS ParallelCluster documentation.<pre><code>pcluster describe-cluster -n <cluster_name></code></pre></td><td>AWS administrator</td></tr>
</table>

Using the Grafana dashboards

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Access to the Grafana portal.</td><td>1. Enter the following command to retrieve the public IP address of the head node.<pre><code>pcluster describe-cluster -n <cluster_name> --query headNode.publicIpAddress</code></pre>2. In a web browser, navigate to the following URL in order to access the Grafana dashboard.

https://<head_node_public_ip_address></td><td>AWS administrator</td></tr>
</table>

Task	Description	Skills required
	3. On the Grafana front page, choose the 4-square **Dashboard** icon on the left menu, and then choose **General**. This shows a list of configured dashboards. The following dashboards are available in Grafana: • **Cluster Cost** – Contains information about the cost of the cluster • **Cluster Logs** – Contains information about the logs of the cluster • **Compute Node Details** – Contains information about usage statistics of the compute nodes • **Compute Node List** – Contains the list of the compute nodes of the cluster • **GPU Nodes** – Contains information about usage statistics of the GPU nodes • **Jobs Details** – Contains information about the jobs resources utilization • **Head Node Details** – Contains information about usage statistics of the head node	

Task	Description	Skills required
	• **ParallelCluster Summary** – Contains information about cluster usage	

Clean up the solution to stop incurring associated costs

Task	Description	Skills required
Delete the cluster.	Enter the following command to delete the cluster. For more information about this command, see [pcluster delete-cluster](#) in the AWS ParallelCluster documentation. `pcluster delete-cluster -n <cluster_name>`	AWS administrator
Delete the IAM policies.	Delete the policies that you created for the head node and compute node. For more information about deleting policies, see [Deleting IAM policies](#) in the IAM documentation.	AWS administrator
Delete the security group and rule.	Delete the security group that you created for the head node. For more information, see [Delete security group rules](#) and [Delete a security](#)	AWS administrator

Task	Description	Skills required
	group in the Amazon VPC documentation.	
Delete the S3 bucket.	Delete the S3 bucket that you created to store the configuration scripts. For more information, see Deleting a bucket in the Amazon S3 documentation.	General AWS

Troubleshooting

Issue	Solution
The head node is not accessible in the browser.	Check the security group and confirm that the inbound port 443 is open.
Grafana doesn't open.	On the head node, check the container log for `docker logs Grafana`.
Some metrics have no data.	On the head node, check the container logs of all containers.

Related resources

AWS documentation

- IAM policies for Amazon EC2

Other AWS resources

- AWS ParallelCluster
- Monitoring dashboard for AWS ParallelCluster (AWS blog post)

Other resources

- Prometheus monitoring system
- Grafana

More patterns

- Implement AI-powered Kubernetes diagnostics and troubleshooting with K8sGPT and Amazon Bedrock integration

Hybrid cloud

Topics

- Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode
- Configure VMware vRealize Automation to provision VMs on VMware Cloud on AWS
- Integrate VMware vRealize Network Insight with VMware Cloud on AWS
- Migrate VMs to VMware Cloud on AWS by using HCX OS Assisted Migration
- Send logs from VMware Cloud on AWS to Splunk by using VMware Aria Operations for Logs
- Set up a CI/CD pipeline for hybrid workloads on Amazon ECS Anywhere by using AWS CDK and GitLab
- More patterns

Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode

Deepak Kumar, Amazon Web Services

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

This pattern describes how you can use Hybrid Linked Mode to view and manage inventories in an on-premises data center and a VMware Cloud on AWS software-defined data center (SDDC) by using a single VMware vSphere Client interface.

By configuring Hybrid Linked Mode, you can migrate your on-premises virtual machines (VMs) and applications to the cloud SDDC. Your IT teams can then manage your cloud-based resources with familiar VMware tools and without requiring any new tools. You can also ensure consistent operations and simplified administration by using the VMware Cloud Gateway Appliance.

This pattern provides two options for configuring Hybrid Linked Mode, but you can only use one option at a time. The first option installs the Cloud Gateway Appliance and uses it to link from the on-premises vCenter Server to the cloud SDDC. The second option configures Hybrid Linked Mode from the cloud SDDC.

Prerequisites and limitations

Prerequisites (both options)

- An existing on-premises data center and a cloud SDDC.
- An existing connection between the on-premises data center and the cloud SDDC, using AWS Direct Connect, a VPN, or both.
- The on-premises data center and cloud SDDC are synchronized with network time protocol (NTP) or another authoritative time source.
- The maximum latency of a round-trip time between the on-premises data center and the cloud SDDC doesn't exceed 100ms.

- Cloud administrators with access to your on-premises environment.
- The vCenter Server’s fully qualified domain name (FQDN) must resolve to a private IP address.

Prerequisites for Option 1

- The on-premises environment should run on vSphere 6.5.0d or later.
- The Cloud Gateway Appliance and vCenter Server can communicate over AWS Direct Connect, a VPN, or both.
- The Cloud Gateway Appliance meets hardware requirements.
- Firewall ports are open.

Prerequisites for Option 2

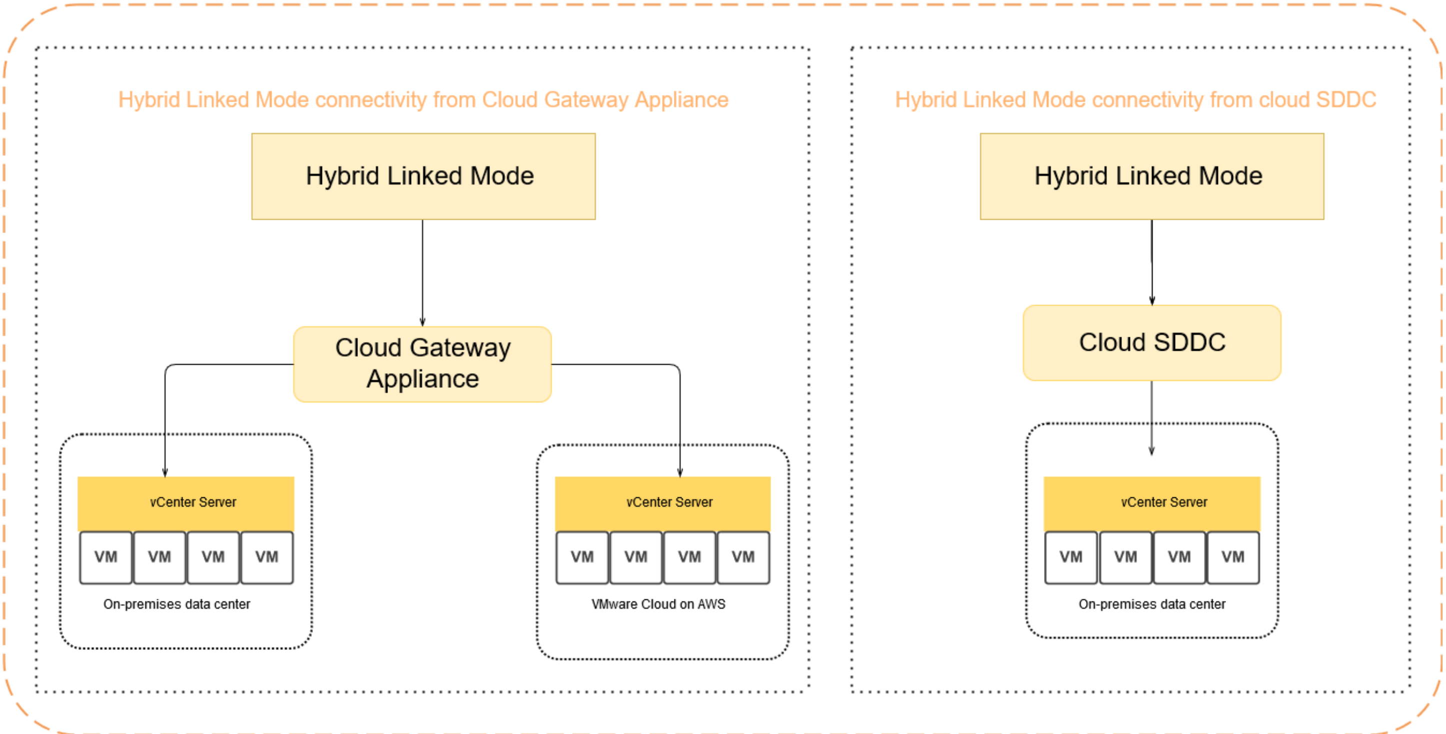
- The on-premises vCenter Server runs on vSphere 6.0 Update 3 or later, or on vSphere 6.5.0d or later .
- Login credentials are available for the on-premises vSphere single sign-on (SSO) domain.
- Users in the on-premises environment have read-only access to the base distinguished name (Base DN).
- The on-premises Domain Name System (DNS) server is configured for VMware Management Gateway.
- Implement network connectivity tests using the VMware Connectivity Validator.
- Firewall ports are open.

Limitations

- Hybrid Linked Mode can only connect one on-premises vCenter Sever Enhanced Linked Mode domain.
- Hybrid Linked Mode only supports on-premises vCenter Server running version 6.7 or later.

Architecture

The following diagram shows both options for configuring Hybrid Linked Mode.

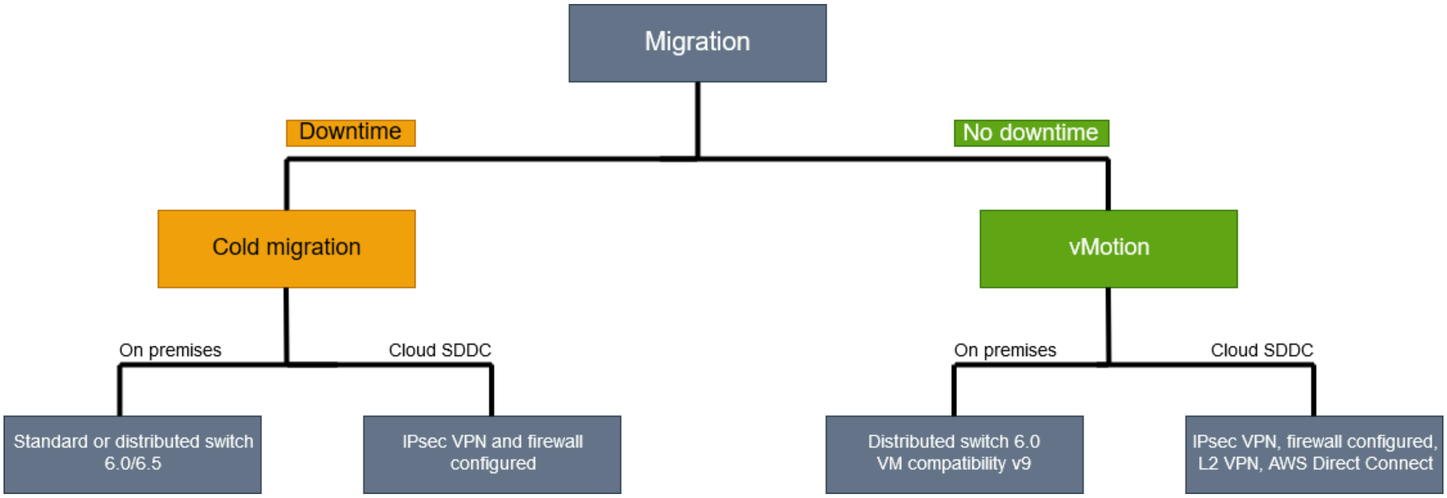


Migrating different workload types using Hybrid Linked Mode

Hybrid Linked Mode supports migrating workloads between an on-premises data center and a cloud SDDC by using either a [cold migration](#) or a live migration with [VMware vSphere vMotion](#). Factors that must be considered when choosing the migration method include the virtual switch type and version, the connection type to the cloud SDDC, and the virtual hardware version.

A cold migration is appropriate for VMs that experience downtime. You can shut down the VMs, migrate them, and then turn them back on. The migration time is faster because there is no need to copy active memory. We recommend using a cold migration for applications that accept downtime (for example, tier 3 applications or development and testing workloads). If your VMs cannot experience downtime, you should consider a live migration using vMotion for your mission-critical applications.

The following diagram provides an overview of the different workload migration types using Hybrid Linked Mode.



Tools

- VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware.
- VMware Cloud Gateway Appliance enables a number of hybrid cloud use cases where on-premises resources are connected to cloud resources.
- VMware vSphere is VMware's virtualization platform, which transforms data centers into aggregated computing infrastructures that include CPU, storage, and networking resources.

Epics

Option 1 - Use Hybrid Linked Mode with the Cloud Gateway Appliance

Task	Description	Skills required
Configure the Cloud Gateway Appliance.	1. Log in to the VMware Cloud on AWS console and download the Cloud Gateway Appliance. 2. Install the Cloud Gateway Appliance in your on-premises environment with the following two steps: • Choose **Start to configure** and then	Cloud administrator

Task	Description	Skills required
	deploy the Cloud Gateway Appliance. • Configure Hybrid Linked Mode. For more information and detailed steps, see [Configuring Hybrid Linked Mode using the vCenter Cloud Gateway Appliance] in the VMware documentation.	

Option 2 - Use Hybrid Linked Mode from the cloud SDDC

Task	Description	Skills required
Configure Hybrid Linked Mode from the cloud SDDC.	1. Log in to the VMware Cloud on AWS console and use the Connectivity Validator to check all required network connectivity. For more information about this, see [Validate network connectivity for Hybrid Linked Mode] in the VMware documentation. 2. Log in to the vSphere Client of the cloud SDDC, choose **Menu**, choose **Administration**, and then choose **Domains**. 3. In the **Hybrid Cloud** section, choose **Linked**	Cloud administrator

Task	Description	Skills required
	Domains and then connect to your on-premises vCenter Server. 4. Add an identity source to the cloud SDDC Lightweig ht Directory Access Protocol (LDAP) domain. For more information about this, see Add an Identity Source to the SDDC LDAP Domain in the VMware documentation.	

Related resources

- Configuring Hybrid Linked Mode
- Configuring Hybrid Linked Mode for VMware Cloud on AWS

Configure VMware vRealize Automation to provision VMs on VMware Cloud on AWS

Deepak Kumar, Amazon Web Services

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

[VMware vRealize Automation](#) is automation software that you can use to request and manage IT resources. By choosing to configure vRealize Automation with VMware Cloud on AWS, you can automate the delivery of virtual machines (VMs), applications, and IT services across multiple data centers and cloud environments.

Your IT teams can then create catalog items to configure service provisioning and operational capabilities that your users can request and use with their existing vRealize Automation tools. You can also improve your IT agility and efficiency by integrating VMware Cloud on AWS with vRealize Automation Cloud Assembly.

This pattern describes how to configure VMware vRealize Automation to automatically build VMs or application capabilities on VMware Cloud on AWS.

Prerequisites and limitations

Prerequisites

- An existing on-premises data center and a VMware Cloud on AWS software-defined data center (SDDC). For more information about the cloud SDCC, see About Software-Defined Data Centers in the VMware documentation.
- An existing connection between the on-premises data center and the cloud SDDC, using AWS Direct Connect, a VPN (route or policy-based), or both.
- The on-premises data center and cloud SDDC are synchronized with network time protocol (NTP) or another authoritative time source.
- The maximum latency of a round-trip time between the on-premises data center and the cloud SDDC doesn’t exceed 100ms.

- The vCenter Server's fully qualified domain name (FQDN) must resolve to a private IP address.
- Cloud SDDC users with access to your on-premises environment.
- Organization owner access in the vRealize Automation Cloud Assembly service role.
- End users with permission in vRealize Automation Service Broker to consume service.
- The on-premises data center's Classless Inter-Domain Routing (CIDR) range must be open for the generating of API tokens from the VMware Cloud on AWS console. The following list provides the minimum roles required to generate API tokens:
  - Organization member
  - Organization owner
  - Service Roles - VMware Cloud on AWS
  - Administrator
  - NSX Cloud Administrator
  - NSX Cloud Auditor

For more information about this, see Connectivity Options for VMware Cloud on AWS SDDCs from the AWS Partner Network Blog.

Limitations

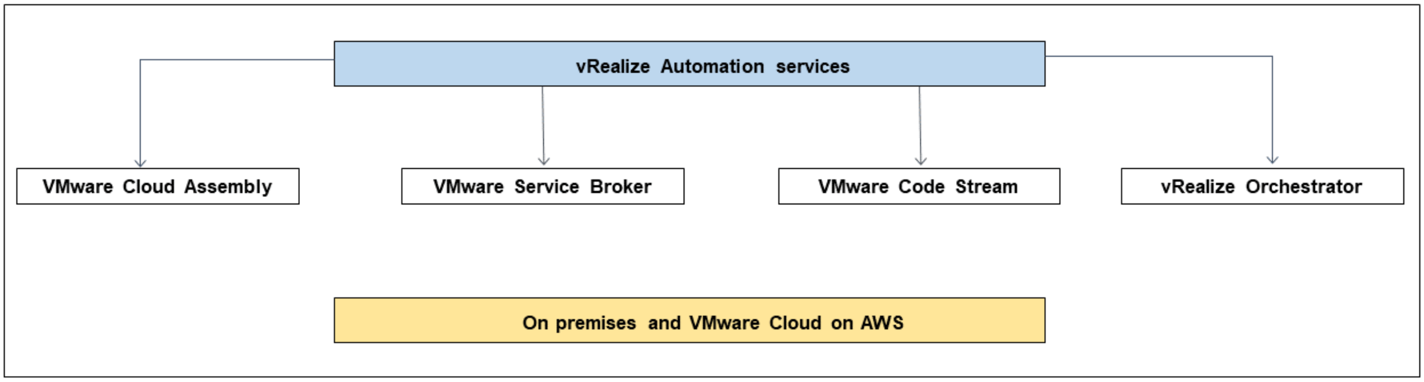
- You can only configure 20 VMware Cloud accounts with public endpoints in one vRealize Automation. For more information about this, see Scalability and concurrency maximums in the VMware documentation.

Product versions

- vRealize Automation version 8.x or later
- VMware vRealize Identity Manager version 3.x or later
- VMware vRealize Suite Lifecycle Manager version 8.x or later

Architecture

The following diagram shows the vRealize Automation services that can use infrastructure from both on-premises and VMware Cloud on AWS environments.



VMware Cloud Assembly components

VMware Cloud Assembly is a core component of vRealize Automation and you can use it to deploy and provision VMs and compute resources. The following table describes VMware Cloud Assembly components that must be configured for provisioning VMs on VMware Cloud on AWS.

Components	Definition
Cloud Account	The Cloud Account provides connection details (for example, server name, user name and password, access key, and API token). VMware Cloud Assembly uses the Cloud Account to collect an inventory of your resources.
Cloud zones	Cloud zones identify resource boundaries in the Cloud Account (for example, AWS Regions and the cloud SDDC). Cloud zones associate compute resources with the Cloud Assembly project.
Projects	A project is a logical entity that consists of users and resources such as cloud zones. It also consists of resource quotas and VM naming policies that are used when building the VM.
Flavor mappings	Flavor mapping provides information about the VM's capacity (for example, number of CPUs and amount of memory) that are used in the Cloud Template.

Image mappings	Image mapping maps the VMware vSphere VM template and Amazon Web Services (AWS) image that are used in the Cloud Template. For more information about this, see Learn more about image mappings in vRealize Automation in the VMware documentation.
Network profile	Network profile controls the placement decision to choose a network during VM provisioning.
Storage profile	Storage profile controls the placement decision to choose storage during VM provisioning.
Cloud Templates	VMware Cloud Templates are an important component of vRealize Automation because they define cloud infrastructure provisioning and orchestration. The Cloud Templates are specifications for the resources and include the resource type, resource properties, and input to be collected from users.

Tools

- VMware vRealize Automation – vRealize Automation is an infrastructure automation platform with event-driven state management and compliance. It is designed to help organizations control and secure self-service clouds, multi-cloud automation with governance, and DevOps-based infrastructure delivery.
- VMware Cloud on AWS – VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware.

Epics

Generate the API tokens

Task	Description	Skills required
Generate the API tokens from your VMware Cloud on AWS account.	1. Sign in to the VMware Cloud Console. 2. On the VMware Cloud Services toolbar, choose **My Account** and then choose **API Token**. 3. Enter a name for your API token, provide the required lifespan, and define the scopes for the token. 4. Choose the **Open ID** check box and then choose **Generate**. 5. Record the API token's credentials. For more information about this, see [How do I generate API tokens](#) in the VMware documentation.	Cloud administrator

Install vRealize Automation in your on-premises data center

Task	Description	Skills required
Download the required software.	Download the VMware vRealize Suite ISO file from the My VMware Portal. This package contains vRealize	Cloud administrator

Task	Description	Skills required
	Suite Lifecycle Manager, VMware Identity Manager, and vRealize Automation.	
Install the software.	Install the software and connect to your cloud SDCC by following the instructi ons from [Installing vRealize Suite Lifecycle Manager with Easy Installer for vRealize Automationand VMware Identity Manager]() in the VMware documentation. **Important** Make sure that the following are available for your installation: • The on-premises VMware vCenter Server setup and login credentials • The network details for the vRealize Automation IP and subnet • The vRealize Automation license key	Cloud administrator, Cloud architect

Connect VMware Cloud on AWS with VMware Cloud Assembly

Task	Description	Skills required
Configure your Cloud Accounts.	1. On the VMware Cloud Console, open the **Infrastructure** tab, choose **Manage – Cloud Accounts**, and then choose **Add Cloud Accounts**. 2. Choose **VMware Cloud on AWS** as the type. 3. Paste the API token information that you recorded earlier. This populates all available cloud SDDCs in your VMware Cloud on AWS organization. 4. Choose the required cloud SDCC and then provide the vCenter user name and password for the SDDC. 5. After you are successfu lly authenticated, you can view the integrated VMware Cloud on AWS account with an **OK** status. For more information about this, see [Create a VMware Cloud on AWS cloud account in vRealize Automation]() in the VMware documentation.	Cloud architect, Cloud administrator

Task	Description	Skills required
Configure the project.	1. On the VMware Cloud Console, open the **Projects** tab and then choose **New project**. 2. Enter the name of your project. 3. Open the **Cloud Zones** tab and choose **default VMware Cloud on AWS Cloud Account**.	Cloud administrator
Configure cloud zone.	1. On the VMware Cloud Console, open **Cloud Zones** and choose the cloud zone for your SDDC data center. 2. By default, `cloudadmin@vmc.local` (this is the default local user ID for the cloud SDDC's vCenter) only has access to provision in the `Compute-ResourcePool` . 3. Open the **Compute** tab under **Cloud Zones** and then choose **Compute-ResourcePool**.	Cloud administrator

Task	Description	Skills required
Configure flavor mapping.	1. Open the **Flavor Mappings** tab and create a new flavor mapping. 2. Enter the flavor name, choose the VMware Cloud on AWS account, and then provide the number of vCPUs and amount of memory.	Cloud administrator
Configure image mapping.	1. Open **Image Mappings** and create a new image mapping. 2. Enter the image name. 3. Choose the VMware Cloud on AWS account and provide the Cloud Account templates that are required.	Cloud administrator
Configure network profile.	1. Open **Network Profile** and create a new network profile. 2. Enter the network profile name. 3. Open the **Network** tab and choose the existing network that you want to use for provisioning.	Cloud administrator

Task	Description	Skills required
Configure storage profile.	1. Open **Storage Profile** and choose **New Storage Profile**. 2. Enter the storage profile's name. 3. In the **Policies** section, create a new policy. 4. Choose **Workload Datastore**. By default `cloudadmin@vmc.local` only has access to provision in the workload's datastore.	Cloud administrator

Task	Description	Skills required
Create the Cloud Template.	1. Open the **Design** tab, choose **Cloud Templates** , and then choose **New From** and **Blank Canvas**. 2. Provide the name and description of the Cloud Template. 3. Choose the project that you created earlier. 4. From the Cloud Template resources design page, drag components into the blank canvas according to your requirements. 5. Choose **Test** to test the template and fix any issues. 6. Choose **Deployment** and provide the deployment name to deploy the VMs. For more information about this, see [Create a basic cloud template] in the VMware documentation.	Cloud administrator

Related resources

- Connect vRealize Automation version 8.x to your SDDC:
- Deploy an SDDC from the VMware Cloud on AWS Console
- AWS Direct Connect Integration with VMware Cloud on AWS

Integrate VMware vRealize Network Insight with VMware Cloud on AWS

Deepak Kumar, Piotr Pitera, and Sachin Trivedi, Amazon Web Services

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

This pattern describes how to integrate VMware vRealize Network Insight with VMware Cloud on AWS and inspect traffic flow from your virtual machines. This integration also helps you plan application migrations to VMware Cloud on AWS.

vRealize Network Insight offers visibility into your network infrastructure. It provides network monitoring and analytics features to improve security, mitigate migration risks, and optimize performance. You can use this tool to monitor traffic flows from your virtual machines and view recommended security rules based on the observed traffic. For more information about vRealize Network Insight, see the VMware documentation.

VMware Cloud on AWS is a pay-as-you-go (on-demand) service that enables enterprises of all sizes to run workloads across VMware vSphere-based cloud environments by using a wide range of AWS services. You can start with a minimum of 2 hosts per SDDC cluster and scale up to 16 hosts per cluster in your production environment. For more information, see the VMware Cloud on AWS website. To learn more about SDDCs, see About Software-Defined Data Centers in the VMware documentation.

Prerequisites and limitations

Prerequisites

- VMware Cloud on AWS SDDC, deployed

Limitations

- For known limitations, see the VMware documentation.

Product versions

- vRealize Network Insight version 5.0.0
- VMware Cloud on AWS SDDC version 1.24

Architecture

Source technology stack

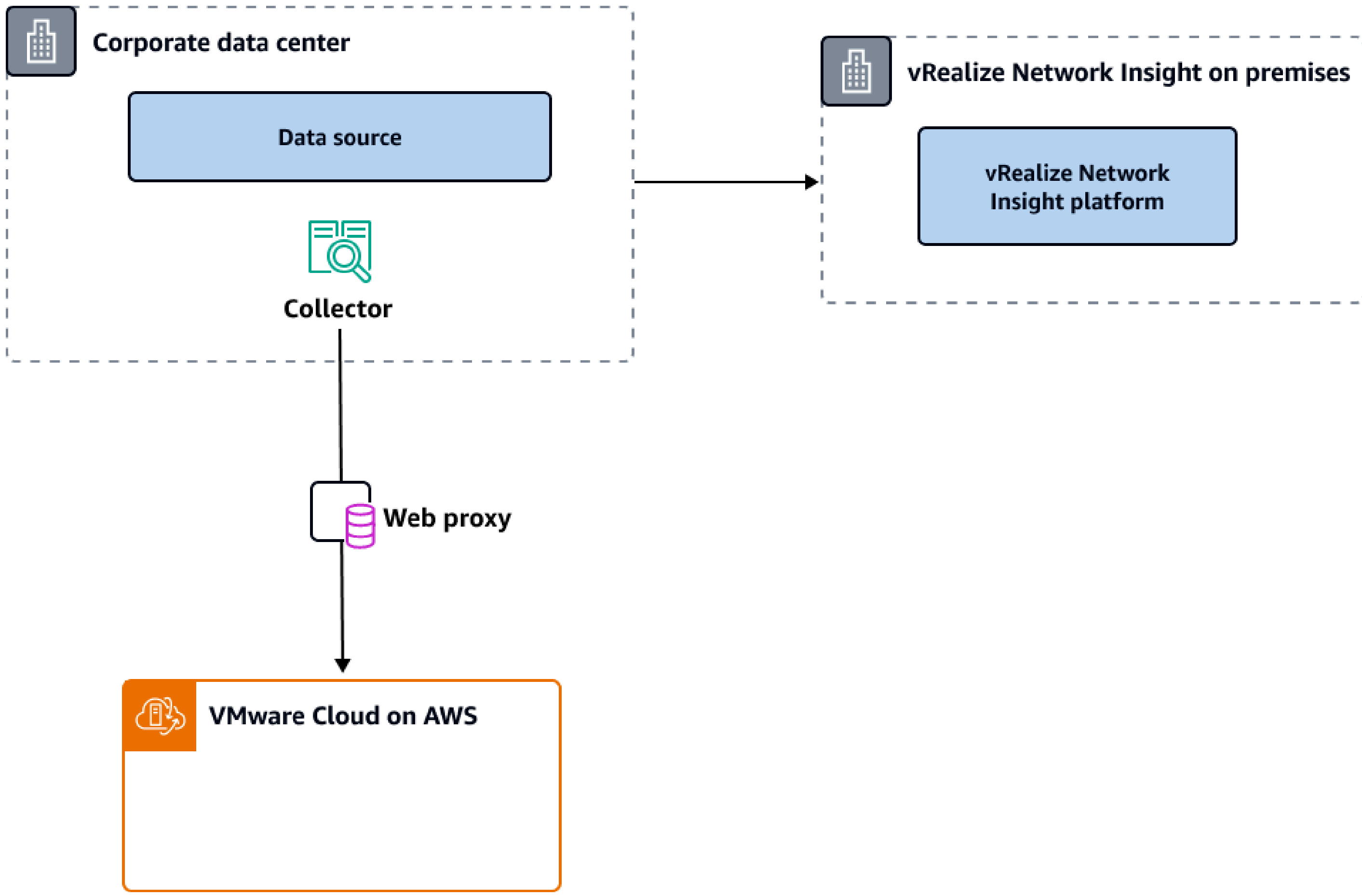
- vRealize Network Insight

Target technology stack

- VMware Cloud on AWS

Target architecture

The following diagram shows the connectivity between VMware Cloud on AWS and vRealize Network Insight on premises.



Tools

- VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware.
- VMware vRealize Network Insight is a monitoring and analytics tool that provides visibility into the network infrastructure for security planning and troubleshooting.

Epics

Set up your environment for vRealize Network Insight

Task	Description	Skills required
Create a VMware user account.	Create a VMware user account or log in to your existing VMware account.	Cloud administrator

Task	Description	Skills required
	To open a new account: 1. Sign up for a VMware Customer Connect account by completing the registrat ion form. New users will receive an email to activate their accounts. 2. Enter the authentication code from the email. 3. Log in to Customer Connect.	
Download the OVA files for vRealize Network Insight.	Download the OVA files for vRealize Network Insight:. 1. Navigate to the VMware product download page at https://my.vmware.com/group/vmware/home. 2. Search for vRealize Network Insight. 3. Download the latest vRealize Network Insight version 5.0.0 platform and collector OVA files.	Cloud administrator
Deploy vRealize Network Insight.	For deployment instructions, see the VMware documenta tion.	Cloud administrator

Add a data source and collector

Task	Description	Skills required
Add a data source.	1. Log in to vRealize Network Insight. 2. Choose **Settings**, **Accounts and Data Sources**, **Add Source**. 3. For **Type**, choose **On-premise vCenter server**. For more information, see the [VMware documentation].	Cloud administrator
Set up a collector for the data source.	For instructions, see the [VMware documentation].	Cloud administrator

Analyze application dependencies

Task	Description	Skills required
Create an application.	If you don't have an existing application in vRealize Network Insight, follow the steps in the [VMware documentation] to create one.	Cloud administrator
Discover and analyze your application.	1. Use vRealize Network Insight to discover your application. For instructi ons, see the [VMware documentation].	Cloud administrator

Task	Description	Skills required
	2. Analyze your application. For instructions, see the VMware documentation.	

Related resources

- Deploy a VMware SDDC on AWS by using VMware Cloud on AWS (AWS Prescriptive Guidance)
- Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode (AWS Prescriptive Guidance)
- Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX (AWS Prescriptive Guidance)
- VMware vRealize Network Insight documentation (VMware website)

Migrate VMs to VMware Cloud on AWS by using HCX OS Assisted Migration

Deepak Kumar and Himanshu Gupta, Amazon Web Services

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

This pattern describes how to migrate a virtual machine (VM) from a non-vSphere environment to VMware Cloud on Amazon Web Services (AWS) by using OS Assisted Migration (OSAM).

OSAM is part of VMware Hybrid Cloud Extension (HCX), which is included with VMware Cloud on AWS. You can use OSAM to migrate a non-vSphere environment such as VMware KVM or Hyper-V to VMware Cloud on AWS. OSAM uses Sentinel software, which you install on a Windows or Linux guest VM to assist in replicating the VM from your on-premises environment to a Software-Defined Data Center (SDDC) on VMware Cloud on AWS.

This pattern explains how to enable OSAM, install Sentinel software on a Windows VM, connect and register with an HCX Sentinel Gateway (SGW) appliance at the source site, and establish a forwarding connection with an HCX Sentinel Data Receiver (SDR) appliance at the destination site to initiate migration.

For more information about OSAM, see the VMware documentation.

Prerequisites and limitations

Prerequisites

- Install HCX in your source and target environments. For HCX prerequisites, see Migrate VMware SDDC to VMware Cloud on AWS using VMware HCX in the AWS Prescriptive Guidance documentation.
- For OSAM prerequisites, see the installation checklist in the VMware documentation.
- For OSAM port information, see VMware HCX port requirements on the VMware Ports and Protocols website.

Limitations

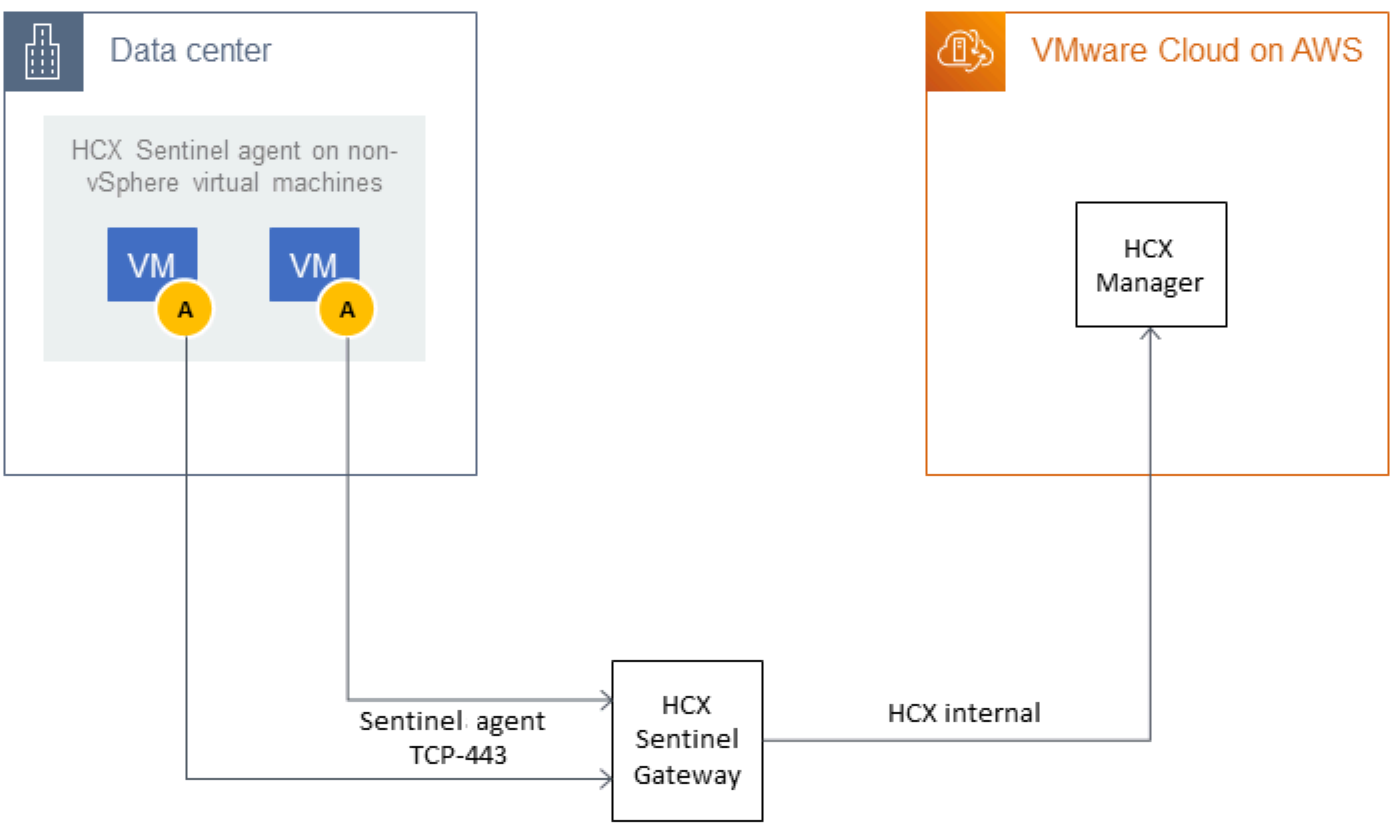
- [VMware HCX 4.2.0 Configuration Limits]
- [Considerations for OSAM Deployment]
- [Supported Guest Operating Systems]
- [Guest Operating System Considerations]

Product versions

- VMware HCX 4.2.0
- VMware SDDC 1.12

Architecture

The following diagram shows how HCX OSAM works with the Sentinel software to replicate non-vSphere VMs from your on-premises environment to VMware Cloud on AWS.



OSAM consists of three components:

- The Sentinel Gateway (SGW) appliance, which is used to connect and forward workloads and applications in the source VMware-based environment
- The Sentinel Data Receiver (SDR), which is used in the destination VMware Cloud on AWS environment to receive migrated workloads from the source
- Sentinel software, which must be installed on each guest VM that you want to migrate

OSAM uses the Sentinel software that is installed on Windows or Linux guest VMs to assist in replicating a VM from on premises to a VMware SDDC. The Sentinel software that you install on guest VMs collects the system configurations from the guest VM and assists with the data replication. This information is also used to create the inventory of guest VMs for migration and helps prepare the disks on the replica VM for replication and migration purposes.

Tools

- VMware HCX 4.2.0
- VMware Cloud on AWS SDDC

Epics

Configure HCX

Task	Description	Skills required
Deploy HCX Cloud and HCX Connector.	Follow the instructions in [HCX Connector and HCX Cloud Installations]() in the VMware documentation.	Cloud administrator, Systems administrator

Configure OSAM and migrate VMs

Task	Description	Skills required
Install HCX Sentinel.	To install Sentinel on Linux: 1. In the vCenter Server for the HCX Connector, choose	Cloud administrator

Task	Description	Skills required
	Interconnect, **Multi-Site Service Mesh**, **Sentinel Management**. 2. Choose **Download Linux Bundle**. 3. Install the Sentinel agent on a Linux machine. For more information, see Downloading and Installing HCX Sentinel Agent software in the VMware documentation.	

Task	Description	Skills required

Migrate VMs.	To migrate your VMs in groups (called *mobility groups*), follow these steps:
	
	1. In the vSphere Client, from the HCX plug-in, choose **Services**, **Migration**.
	2. Choose **Migrate**.
	3. Choose **Non vSphere Inventory**, **Remote connections**. This will show the list VMs that you installed HCX Sentinel on.
	4. For **Group name**, enter the name of the mobility group you want to create for the VMs.
	5. Choose the VMs you want to migrate, and then choose **Add** to add them to the mobility group.
	6. For each VM:
	  a. Select the destination compute container.
	  b. Select the destination storage.
	  c. Select the migration profile.
	  d. Select the destination folder.
	7. To start the migration process, choose **Go**.
		Cloud administrator

Task	Description	Skills required
	HCX validates your VM selections before migration starts. For more information, see Migrating Virtual Machines with Mobility Groups and Monitoring and Estimating Migration with Mobility Groups in the VMware documentation.	

Related resources

VMware documentation:

- VMware HCX User Guide
- Install Checklist B - HCX with a VMC SDDC Destination Environment
- VMware HCX in the VMware Cloud on AWS
- HCX OS Assisted Migration for VMware Cloud on AWS
- VMware HCX 4.2.1 Release Notes

Send logs from VMware Cloud on AWS to Splunk by using VMware Aria Operations for Logs

Deepak Kumar and Piotr Pitera, Amazon Web Services

Summary

Notice: As of April 30, 2024, VMware Cloud on AWS is no longer resold by AWS or its channel partners. The service will continue to be available through Broadcom. We encourage you to reach out to your AWS representative for details.

This pattern describes how to forward VMware Cloud on AWS events or logs to a syslog or an HTTP endpoint such as Splunk by using VMware Aria Operations for Logs.

VMware Aria Operations for Logs is a log analysis tool that offers enhanced visibility and accelerated troubleshooting in the VMware Cloud on AWS environment. You can configure this tool to send either all or a portion of logs or events in VMware Cloud on AWS to a syslog or HTTP endpoint. The endpoint can be either a software as a service (SaaS) endpoint or an on-premises endpoint such as Splunk. (This pattern provides the instructions for Splunk.) To learn more about VMware Aria Operations for Logs, see the VMware documentation.

VMware Cloud on AWS is a pay-as-you-go (on-demand) service that enables enterprises of all sizes to run workloads across VMware vSphere-based cloud environments by using a wide range of AWS services. You can start with a minimum of 2 hosts per Software-Defined Data Center (SDDC) cluster and scale up to 16 hosts per cluster in your production environment. For more information, see the VMware Cloud on AWS website. To learn more about SDDCs, see About Software-Defined Data Centers in the VMware documentation.

Prerequisites and limitations

Prerequisites

- Splunk, configured on premises

Limitations

You can sign up for a free trial subscription to VMware Aria Operations for Logs. This subscription is valid for 30 days and has the following limitations:

- Maximum size of logs you can forward: 50 GB logs per day
- Maximum number of log forwarding configurations you can create: 10
- Maximum number of log forwarding configurations you can activate: 5

To access all service features, you must upgrade to a premium subscription.

For more information about trial and premium subscriptions, see VMware Aria Operations for Logs (SaaS) Subscriptions and Billing in the VMware documentation. For more information about usage limits, see Usage Limitations for Features in the VMware documentation.

Product versions

- VMware Cloud on AWS SDDC version 1.24
- VMware Aria Operations for Logs version 8.10
- On-premises Splunk version 9.x

Architecture

Source technology stack

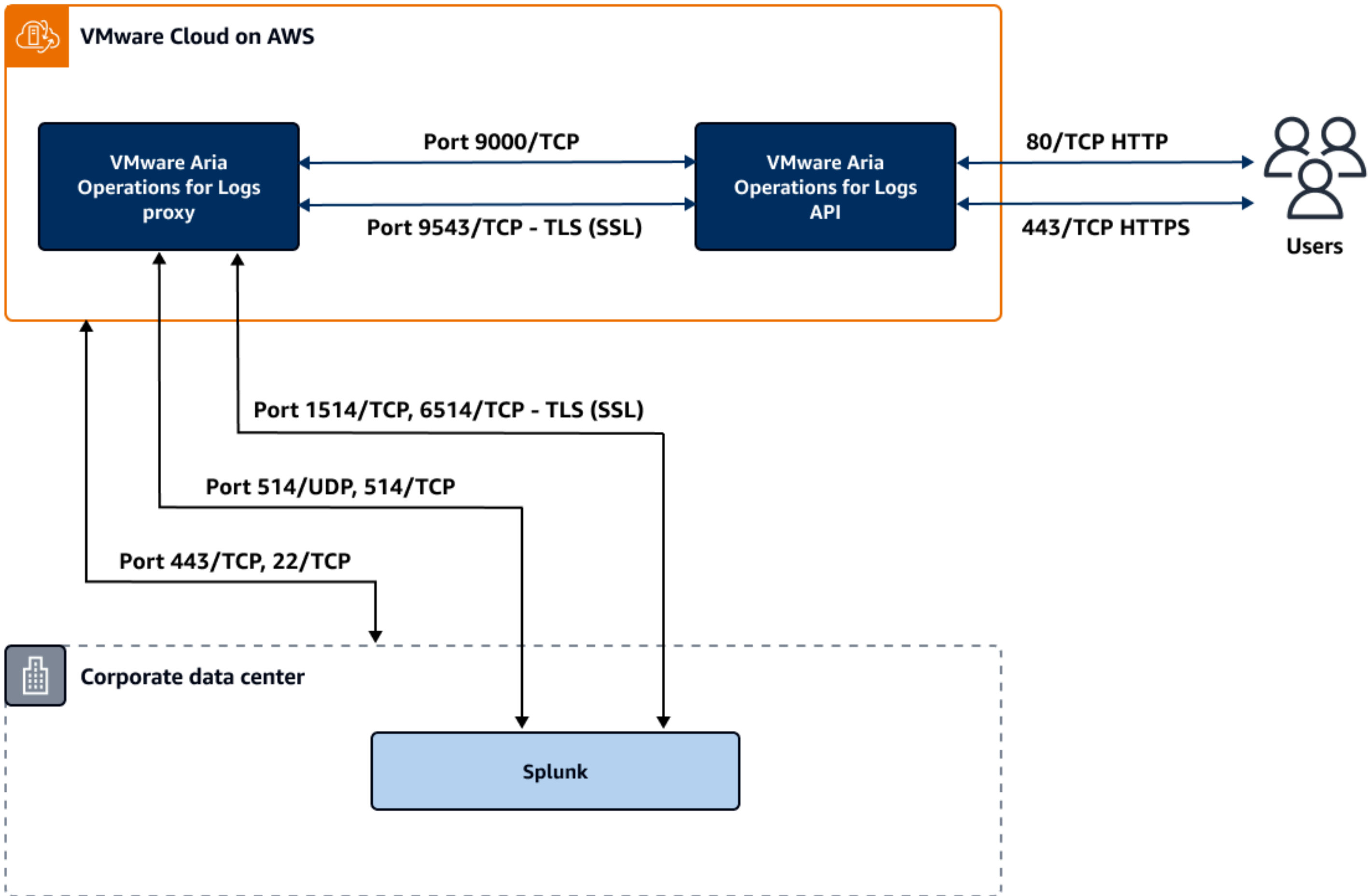
- VMware Cloud on AWS
- VMware Aria Operations for Logs

Target technology stack

- On-premises Splunk

Target architecture

The following diagram shows the connectivity between a corporate data center and VMware Aria Operations for Logs in VMware Cloud on AWS.



Tools

- VMware Cloud on AWS is an integrated cloud offering jointly developed by AWS and VMware.
- VMware Aria Operations for Logs is a log analysis and troubleshooting tool for VMware Cloud on AWS.

Epics

Deploy an SDDC and enable VMware Aria Operation for Logs

Task	Description	Skills required
Deploy a VMware Cloud on AWS SDDC.	Follow the instructions in Deploy a VMware SDDC on AWS by using VMware Cloud	Cloud architect, Cloud administrator

Task	Description	Skills required
	on AWS in AWS Prescriptive Guidance.	
Sign up for VMware Aria Operations for Logs.	For instructions, see the VMware documentation.	Cloud architect

Deploy a cloud proxy

Task	Description	Skills required
Deploy a cloud proxy.	To forward logs to an on-premises instance of Splunk, you must add a cloud proxy for VMware Aria Operations for Logs. This proxy receives information from the on-premises data center and sends it to VMware Aria Operations for Logs for analysis. To download and install the cloud proxy: 1. Make sure that ports 443, 22, and 514 are open between your on-premises environment and VMware Cloud on AWS. For additional ports, you can use 1514/TCP or 6514/TCP. For more information about ports, see VMware Aria Operations for Logs Firewall Recommendations	Cloud administrator, Cloud architect

Task	Description	Skills required
	in the VMware documentation. 2. Log in to VMware Aria Operations for Logs. 3. On the home page, choose **Add Collector** in the widget. 4. On the **Cloud Proxy Virtual Appliance** screen, copy the token key. You must use this key within 24 hours to finish the following steps. 5. Choose the download link for the OVA file. 6. Navigate to VMware vSphere web client, choose your cluster, and then select **Deploy OVF template**. 7. When you're prompted for the key, paste the token key that you copied in step 4. 8. Choose **Finish** to install the cloud proxy.	

Forward logs to an on-premises Splunk endpoint

Task	Description	Skills required
Configure log forwarding.	To forward logs to the Splunk endpoint:	

Task	Description	Skills required
	1. Log in to VMware Aria Operations for Logs. 2. Navigate to **Log Management**. 3. Choose **Log Forwarding**. 4. Choose **New Configuration**, and complete the following settings: • Provide a name for the log forwarding configuration. • For **Destination**, choose **On Premises**. • For **Cloud Proxy**, select the cloud proxy that you installed earlier. • For **Endpoint Type**, choose **TCP**. • For **Endpoint URL**, provide your on-premises Splunk URL in the format: `tcp://x.x.x.x (your Splunk IP address):514` • (Optional) For **Tags**, you can specify tag names and values to facilitate querying. • Choose **Apply to all logs** or **Apply to specific logs**. If you want to send all	

Task	Description	Skills required
	VMware Cloud on AWS logs to Splunk, choose **Apply to all logs**. 5. Choose **Verify**. 6. Choose **Save**. For more information, see Forward Logs from VMware Aria Operations for Logs in the VMware documentation.	

Related resources

- VMware Cloud on AWS website
- About Software-Defined Data Centers (VMware documentation)
- Deploy a VMware SDDC on AWS by using VMware Cloud on AWS (AWS Prescriptive Guidance)
- Migrate workloads to the VMware Cloud on AWS by using VMware HCX (AWS Prescriptive Guidance)
- Configure a data center extension to VMware Cloud on AWS using Hybrid Linked Mode (AWS Prescriptive Guidance)

Set up a CI/CD pipeline for hybrid workloads on Amazon ECS Anywhere by using AWS CDK and GitLab

Dr. Rahul Sharad Gaikwad, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more.

Amazon ECS Anywhere is an extension of the Amazon Elastic Container Service (Amazon ECS). It provides support for registering an *external instance*, such as an on-premises server or virtual machine (VM), to your Amazon ECS cluster. is feature helps reduce costs and mitigate complex local container orchestration and operations. You can use ECS Anywhere to deploy and run container applications in both on-premises and cloud environments. It removes the need for your team to learn multiple domains and skill sets, or to manage complex software on their own.

This pattern describes a step-by-step approach to provision an Amazon ECS cluster with Amazon ECS Anywhere instances by using Amazon Web Services (AWS) Cloud Development Kit (AWS CDK) stacks. You then use AWS CodePipeline to set up a continuous integration and continuous deployment (CI/CD) pipeline. Then, you replicate your GitLab code repository to AWS CodeCommit and deploy your containerized application on the Amazon ECS cluster.

This pattern is designed to help those who use on-premises infrastructure to run container applications and use GitLab to manage the application code base. You can manage those workloads by using AWS Cloud services, without disturbing your existing, on-premises infrastructure.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A container application running on on-premises infrastructure.
- A GitLab repository where you manage your application code base. For more information, see Repository (GitLab).
- AWS Command Line Interface (AWS CLI), installed and configured. For more information, see Installing or updating the latest version of the AWS CLI (AWS CLI documentation).

- AWS CDK Toolkit, installed and configured globally. For more information, see Install the AWS CDK (AWS CDK documentation).
- npm, installed and configured for the AWS CDK in TypeScript. For more information, see Downloading and installing Node.js and npm (npm documentation).

Limitations

- For limitations and considerations, see External instances (Amazon ECS Anywhere) in the Amazon ECS documentation.

Product versions

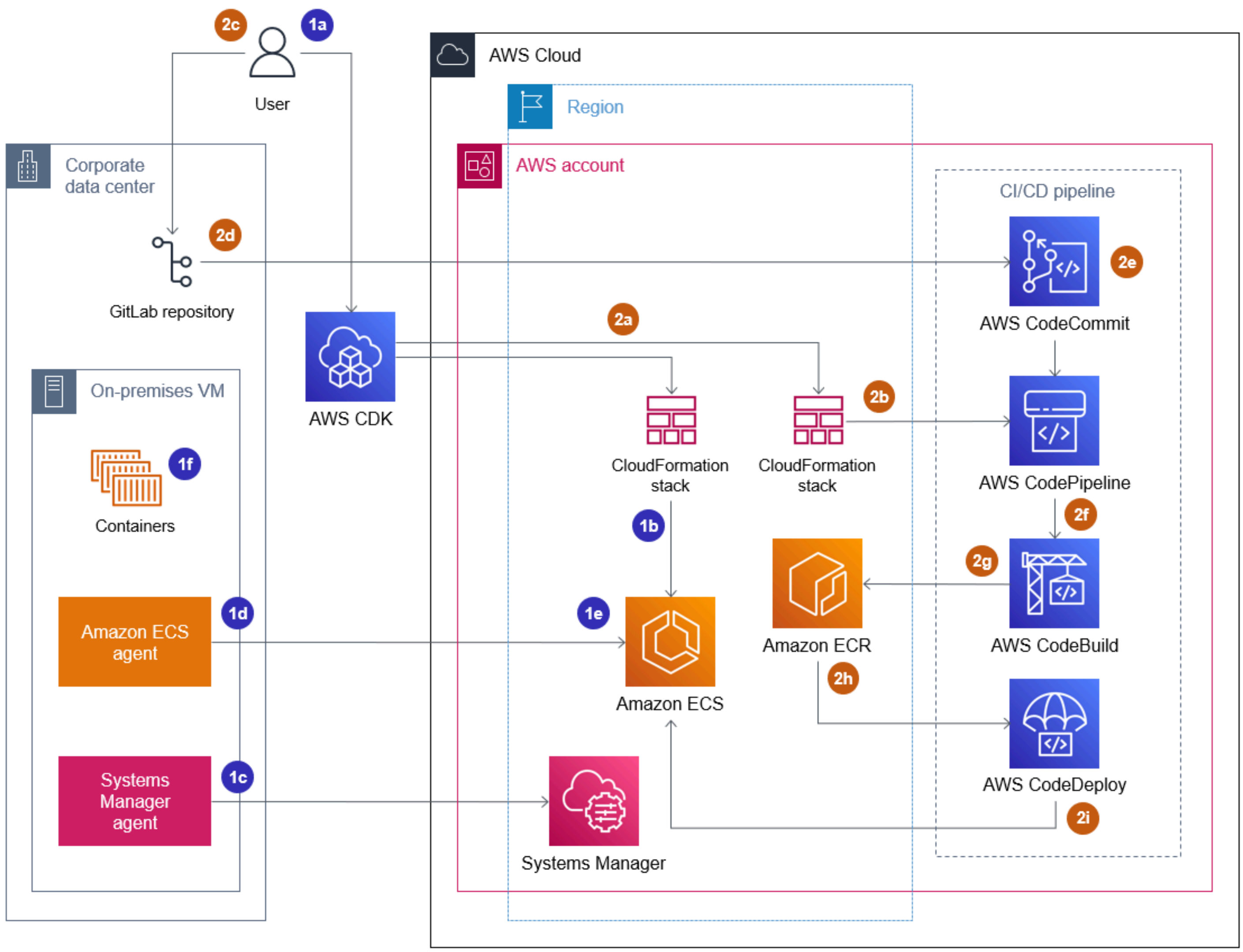
- AWS CDK Toolkit version 2.27.0 or later
- npm version 7.20.3 or later
- Node.js version 16.6.1 or later

Architecture

Target technology stack

- AWS CDK
- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- Amazon ECS Anywhere
- Amazon Elastic Container Registry (Amazon ECR)
- AWS Identity and Access Management (IAM)
- AWS System Manager
- GitLab repository

Target architecture



This diagram represents two primary workflows described in this pattern, provisioning the Amazon ECS cluster and setting up the CI/CD pipeline that sets up and deploys the CI/CD pipeline, as follows:

1. **Provisioning the Amazon ECS cluster**
   a. When you deploy the first AWS CDK stack, it creates a CloudFormation stack on AWS.
   b. This CloudFormation stack provisions an Amazon ECS cluster and related AWS resources.
   c. To register an external instance with an Amazon ECS cluster, you must install AWS Systems Manager Agent (SSM Agent) on your VM and register the VM as an AWS Systems Manager managed instance.
   d. You must also install the Amazon ECS container agent and Docker on your VM to register it as an external instance with the Amazon ECS cluster.

e. When the external instance is registered and configured with the Amazon ECS cluster, it can run multiple containers on your VM, which is registered as an external instance.

f. The Amazon ECS cluster is active and can run the application workloads through containers. The Amazon ECS Anywhere container instance runs in on-premises environment but is associated with the Amazon ECS cluster in the cloud.

2. **Setting up and deploying the CI/CD pipeline**

   a. When you deploy the second AWS CDK stack, it creates another CloudFormation stack on AWS.

   b. This CloudFormation stack provisions a pipeline in CodePipeline and related AWS resources.

   c. You push and merge application code changes to an on-premises GitLab repository.

   d. The GitLab repository is automatically replicated to the CodeCommit repository.

   e. The updates to the CodeCommit repo automatically starts CodePipeline.

   f. CodePipeline copies code from CodeCommit and creates the deployable application build in CodeBuild.

   g. CodePipeline creates a Docker image of the CodeBuild build environment and pushes it to the Amazon ECR repo.

   h. CodePipeline initiates CodeDeploy actions that pull the container image from the Amazon ECR repo.

   i. CodePipeline deploys the container image on the Amazon ECS cluster.

Automation and scale

This pattern uses the AWS CDK as an infrastructure as code (IaC) tool to configure and deploy this architecture. AWS CDK helps you orchestrate the AWS resources and set up Amazon ECS Anywhere and the CI/CD pipeline.

Tools

AWS services

- [AWS Cloud Development Kit (AWS CDK)]() is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- [AWS CodeCommit]() is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.

- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster. This pattern also uses Amazon ECS Anywhere, which provides support for registering an on-premises server or VM to your Amazon ECS cluster.

Other tools

- Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.
- Vagrant is an open-source utility for building and maintaining portable virtual software development environments. For demonstration purposes, this pattern uses Vagrant to create an on-premises VM.

Code repository

The code for this pattern is available in the GitHub CI/CD pipeline for Amazon ECS Anywhere using AWS CDK repository.

Best practices

Consider the following best practices when deploying this pattern:

- Best practices for developing and deploying cloud infrastructure with the AWS CDK
- Best practices for developing cloud applications with AWS CDK (AWS blog post)

Epics

Verify the AWS CDK configuration

Task	Description	Skills required
Verify the AWS CDK version.	Verify the version of the AWS CDK Toolkit by entering the following command. `cdk --version` This pattern requires version 2.27.0 or later. If you have an earlier version, follow the instructions in the AWS CDK documentation to update it.	DevOps engineer
Verify the npm version.	Verify the version of npm by entering the following command. `npm --version` This pattern requires version 7.20.3 or later. If you have an earlier version, follow the instructions in the npm documentation to update it.	DevOps engineer
Set up AWS credentials.	Set up AWS credentials by entering the `aws configure` command and following the prompts. `$aws configure`	DevOps engineer

Task	Description	Skills required
	```AWS Access Key ID [None]: <your-access-key-ID> AWS Secret Access Key [None]: <your-secret-access-key> Default region name [None]: <your-Region-name> Default output format [None]:```	

Bootstrap the AWS CDK environment

Task	Description	Skills required

Clone the AWS CDK code repository.	1. Clone the [CI/CD pipeline for Amazon ECS Anywhere using AWS CDK]() repository for this pattern by entering the following command.
	`git clone https://github.com/aws-samples/amazon-ecs-anywhere-cicd-pipeline-cdk-sample.git`
	2. Navigate into the cloned directory by entering the following command.
	`cd amazon-ecs-anywhere-cicd-pipeline-cdk-sample`
		DevOps engineer

Task	Description	Skills required
Bootstrap the environment.	Deploy the CloudFormation template to the account and AWS Region that you want to use by entering the following command. `cdk bootstrap <account-number>/<Region>` For more information, see [Bootstrapping](#) in the AWS CDK documentation.	DevOps engineer

Build and deploy the infrastructure for Amazon ECS Anywhere

Task	Description	Skills required

Install the package dependencies and compile the TypeScript files.	Install the package dependencies and compile the TypeScript files by entering the following commands.
	`$cd EcsAnywhereCdk`
	`$npm install`
	`$npm fund`
	These commands install all the packages from the sample repository. For more informati on, see npm ci and npm install in the npm documenta tion. If you get any errors about missing packages when
		DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>you enter these commands, see the Troubleshooting section of this pattern.</td><td></td></tr>
<tr><td>Build the project.</td><td>To build the project code, enter the following command.
<pre>npm run build</pre>For more information about building and deploying the project, see Your first AWS CDK app in the AWS CDK documentation.</td><td>DevOps engineer</td></tr>
<tr><td>Deploy the Amazon ECS Anywhere infrastructure stack.</td><td>1. List the stacks by entering the following command.
<pre>$cdk list</pre>2. Confirm that the output returns the <code>EcsAnywhereInfraStack</code> and <code>ECSAnywherePipelineStack</code> stacks.
3. Deploy the <code>EcsAnywhereInfraStack</code> stack by entering the following command.
<pre>$cdk deploy EcsAnywhereInfraStack</pre></td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
Verify stack creation and output.	1. Sign in to the AWS Management Console and open the CloudFormation console at https://console.aws.amazon.com/cloudformation/. 2. On the **Stacks** page, select the `EcsAnywhereInfraStack` stack. 3. Confirm that the stack status is either `CREATE_IN_PROGRESS` or `CREATE_COMPLETE` . Setting up the Amazon ECS cluster can take some time. Do not proceed until the stack creation is complete.	DevOps engineer

Set up an on-premises VM

Task	Description	Skills required
Set up your VM.	Create a Vagrant VM by entering the `vagrant up` command from the root directory where Vagrantfile is located. For more information, see the [Vagrant documentation](https://www.vagrantup.com/docs).	DevOps engineer
Register your VM as an external instance.	1. Log in to the Vagrant VM by using the `vagrant ssh` command. For more	DevOps engineer

Task	Description	Skills required
	information, see the [Vagrant documentation](). 2. Install AWS CLI on the VM by following [AWS CLI installation instructions]() and entering the following commands. `$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" \` `> -o "awscliv2.zip"` `$sudo apt install unzip` `$unzip awscliv2.zip` `$sudo ./aws/install` `$aws configure` `AWS Access Key ID [None]: <your-access-key-ID>` `AWS Secret Access Key [None]: <your-secret-access-key>` `Default region name [None]: <your-Region-name>` `Default output format [None]:` 1. Create an activation code and ID that you can use to register your VM with AWS Systems Manager and to activate your external instance. The output from this command includes the	

Task	Description	Skills required

	activation ID and activation code values.
	`aws ssm create-activation \ > --iam-role EcsAnywhereInstanceRole \ > | tee ssm-activation.json`
	If you receive an error when you run this command, see the Troubleshooting section.
	2. Export the activation ID and code values.
	`export ACTIVATION_ID=<activation-ID>`
	`export ACTIVATION_CODE=<activation-code>`
	3. Download the installation script to your VM.
	`curl --proto "https" -o "ecs-anywhere-install.sh" \ > "https://amazon-ecs-agent.s3.amazonaws.com/ecs-anywhere-install-latest.sh"`
	4. Run the installation script on your VM.	

Task	Description	Skills required

	<pre>sudo bash ecs-anywh
	ere-install.sh \
	--cluster EcsAnywhe
	reCluster \
	--activation-id
	 $ACTIVATION_ID \
	--activation-code
	 $ACTIVATION_CODE \
	--region <region-n
	ame></pre> This sets up your VM was an Amazon ECS Anywhere external instance and registers the instance in the Amazon ECS cluster. For more information, see [Registering an external instance to a cluster]() in the Amazon ECS documentation. If you experience any issues, see the [Troubleshooting]() section.	

Verify the status of Amazon ECS Anywhere and the external VM.	To verify whether your VM is connected to the Amazon ECS control plane and running, use the following commands. <pre>$aws ssm describe-
	instance-information
	$aws ecs list-cont
	ainer-instances --
	cluster $CLUSTER_NAME</pre>
		DevOps engineer

Deploy the CI/CD pipeline

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a branch in the CodeCommit repo.</td><td>Create a branch named <code>main</code> in the CodeCommit repo by creating the first commit for the repository. You can follow AWS documentation to Create a commit in CodeCommit. The following command is an example.<pre>aws codecommit put-file \
  --repository-name EcsAnywhereRepo \
  --branch-name main \
  --file-path README.md \
  --file-content "Test" \
  --name "Dev Ops" \
  --email "devops@example.com" \
  --commit-message "Adding README."</pre></td><td>DevOps engineer</td></tr>
<tr><td>Set up repo mirroring.</td><td>You can mirror a GitLab repository to and from external sources. You can select which repository serves as the source. Branches, tags, and commits are synced automatically. Set up a push mirror between the GitLab repository that hosts your application and the CodeCommit repository.</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	For instructions, see [Set up a push mirror from GitLab to CodeCommit](#) (GitLab documentation). **Note** By default, mirroring automatically syncs the repository. If you want to manually update the repositories, see [Update a mirror](#) (GitLab documentation).	
Deploy the CI/CD pipeline stack.	Deploy the `EcsAnywherePipelineStack` stack by entering the following command. `$cdk deploy EcsAnywherePipelineStack`	DevOps engineer

Task	Description	Skills required
Test the CI/CD pipeline.	1. Make application code changes and push it to the source, on-premises GitLab repo. For more informati on, see [Push Options](#) (GitLab documentation). For example, edit the `../application/index.html` file to update the application version value. 2. When the code is replicate d to the CodeCommit repo, this starts the CI/CD pipeline. Do one of the following: • If you are using automatic mirroring to synchronize the GitLab repo with the CodeCommit repo, continue to the next step. • If you are using manual mirroring, push the application code changes to the CodeCommit repo by following the instructi ons in [Update a mirror](#) (GitLab documentation). 3. On your local machine, in a web browser, enter http://localhost:80. This opens the NGINX web	DevOps engineer

Task	Description	Skills required
	page because port 80 is forwarded to localhost in Vagrantfile. Confirm that you can view the updated application version value. This validates the pipeline and image deployment. 4. (Optional) If you want to verify the deployment in the AWS Management Console, do the following: a. Open the Amazon ECS console at https://console.aws.amazon.com/ecs/. b. From the navigation bar, select the Region to use. c. In the navigation pane, choose **Clusters**. d. On the **Clusters** page, select the **EcsAnywhereCluster** cluster. e. Choose **Task Definitions**. f. Confirm the container is running.	

Clean up

Task	Description	Skills required
Clean up and delete the resources.	After you walk through this pattern, you should	DevOps engineer

Task	Description	Skills required

	remove the proof-of-concept resources you created. To clean up, enter the following commands.
	`$cdk destroy EcsAnywherePipelineStack`
	`$cdk destroy EcsAnywhereInfraStack`	

Troubleshooting

Issue	Solution

Errors about missing packages when installing package dependencies.	Enter one of the following commands to resolve missing packages.
	`$npm ci`
	or
	`$npm install -g @aws-cdk/<package_name>`

When you run the `aws ssm create-activation` command on the VM, you receive the following error. `An error occurred (ValidationException) when calling the CreateActivation operation: Nonexistent role or missing ssm service principal in trust policy:`	The `EcsAnywhereInfraStack` stack isn't fully deployed, and the IAM role necessary to run this command hasn't been created yet. Check the stack status in the CloudFormation console. Retry the command after the status changes to `CREATE_COMPLETE`.

Issue	Solution
`arn:aws:iam::000000000000:role/EcsAnywhereInstanceRole`	

An Amazon ECS health check returns `UNHEALTHY` , and you see the following error in the **Services** section of the cluster in the Amazon ECS console.
 	
`service EcsAnywhereService was unable to place a task because no container instance met all of its requirements. Reason: No Container Instances were found in your cluster.`
	Restart the Amazon ECS agent on your Vagrant VM by entering the following commands.
	
	`$vagrant ssh`
	`$sudo systemctl restart ecs`
	`$sudo systemctl status ecs`

Related resources

- Amazon ECS Anywhere marketing page
- Amazon ECS Anywhere documentation
- Amazon ECS Anywhere demo (video)
- Amazon ECS Anywhere workshop samples (GitHub)
- Repository mirroring (GitLab documentation)

More patterns

- Automate the setup of inter-Region peering with AWS Transit Gateway
- Manage on-premises container applications by setting up Amazon ECS Anywhere with the AWS CDK
- Migrate Hadoop data to Amazon S3 by using WANdisco LiveData Migrator
- Migrate VMware VMs with HCX Automation by using PowerCLI
- Modify HTTP headers when you migrate from F5 to an Application Load Balancer on AWS
- Rehost on-premises workloads in the AWS Cloud: migration checklist
- Use BMC Discovery queries to extract migration data for migration planning

Management & governance

Topics

- Identify and alert when Amazon Data Firehose resources are not encrypted with an AWS KMS key
- Automate adding or updating Windows registry entries using AWS Systems Manager
- Automatically create an RFC in AMS using Python
- Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows
- Centralize software package distribution in AWS Organizations by using Terraform
- Configure logging for .NET applications in Amazon CloudWatch Logs by using NLog
- Copy AWS Service Catalog products across different AWS accounts and AWS Regions
- Create a RACI or RASCI matrix for a cloud operating model
- Create alarms for custom metrics using Amazon CloudWatch anomaly detection
- Create an AWS Cloud9 IDE that uses Amazon EBS volumes with default encryption
- Create tag-based Amazon CloudWatch dashboards automatically
- Document your AWS landing zone design
- Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK
- Govern permission sets for multiple accounts by using Account Factory for Terraform
- Implement Account Factory for Terraform (AFT) by using a bootstrap pipeline
- Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions
- Migrate an AWS member account from AWS Organizations to AWS Control Tower
- Monitor SAP RHEL Pacemaker clusters by using AWS services
- Monitor application activity by using CloudWatch Logs Insights
- Monitor use of a shared Amazon Machine Image across multiple AWS accounts
- Set up alerts for programmatic account closures in AWS Organizations
- View EBS snapshot details for your AWS account or organization
- More patterns

Identify and alert when Amazon Data Firehose resources are not encrypted with an AWS KMS key

Ram Kandaswamy, Amazon Web Services

Summary

For compliance, some organizations must have encryption enabled on data delivery resources such as Amazon Data Firehose. This pattern shows a way to monitor, detect, and notify when resources are out of compliance.

To maintain the encryption requirement, this pattern can be used on AWS to provide automated monitoring and detection of Amazon Data Firehose delivery resources that aren't encrypted with an AWS Key Management Service (AWS KMS) key. The solution sends alert notifications, and it can be extended to perform automatic remediation. This solution can be applied to an individual account or a multiple-account environment, such as an environment that uses an AWS landing zone or AWS Control Tower.

Prerequisites and limitations

Prerequisites

- Amazon Data Firehose delivery stream
- Sufficient permissions and familiarity with AWS CloudFormation, which is used in this infrastructure automation

Limitations

- The solution is not real time because it uses AWS CloudTrail events for detection, and there is a delay between the time an unencrypted resource is created and the notification is sent.

Architecture

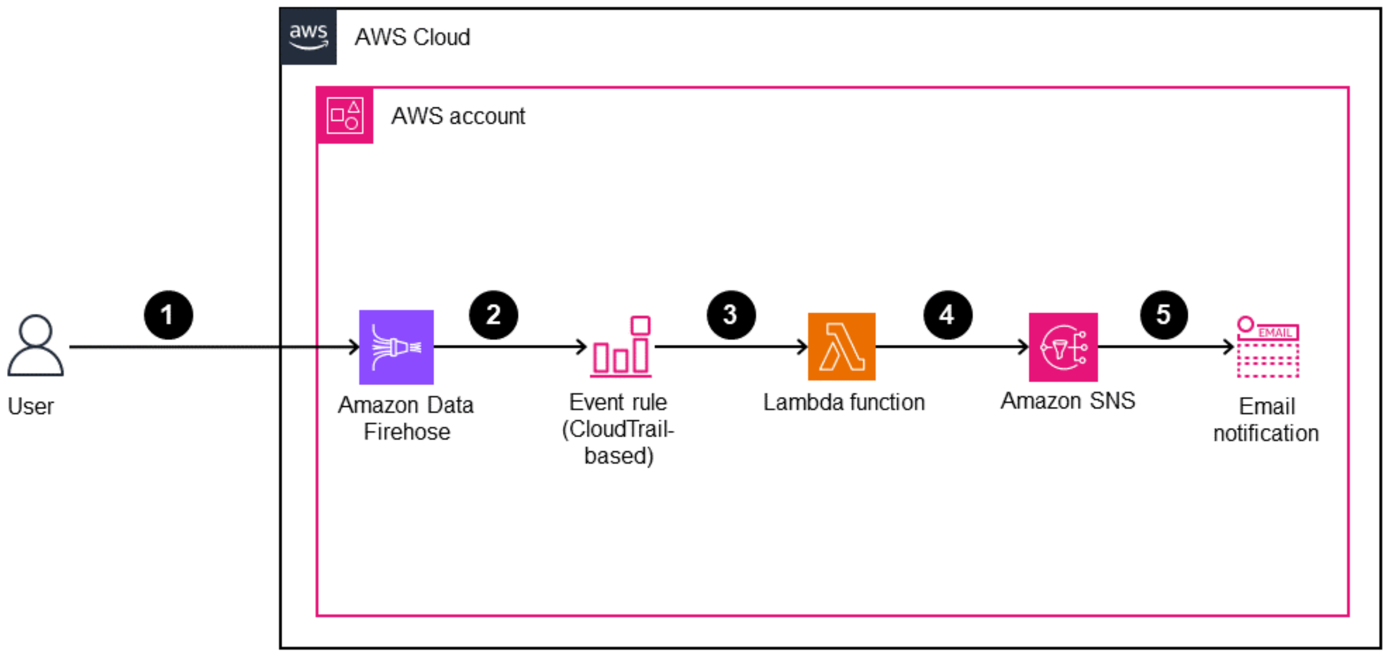
Target technology stack

The solution uses serverless technology and the following services:

- AWS CloudTrail
- Amazon CloudWatch

- AWS Command Line Interface (AWS CLI)
- AWS Identity and Access Management (IAM)
- Amazon Data Firehose
- AWS Lambda
- Amazon Simple Notification Service (Amazon SNS)

Target architecture



The diagram illustrates these steps:

1. A user creates or modifies Amazon Data Firehose.
2. A CloudTrail event is detected and matched.
3. Lambda is invoked.
4. Non-compliant resources are identified.
5. Email notification is sent.

Automation and scale

You can use AWS CloudFormation StackSets to apply this solution to multiple AWS Regions or accounts with a single command.

Tools

- AWS CloudTrail is an AWS service that helps you enable governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS CLI, AWS SDKs, and API operations.
- Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services by using commands in your command line shell.
- AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- Amazon Data Firehose is a fully managed service for delivering real-time streaming data. With Firehose, you don't have to write applications or manage resources. You configure your data producers to send data to Firehose, and it automatically delivers the data to the destination that you specified.
- AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code isn’t running.
- Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).

Epics

Enforce encryption for compliance

Task	Description	Skills required
Deploy AWS CloudFormation StackSets.	In the AWS CLI, use the `firehose-encryption-checker.yaml` template (attached) to create the stack set by running the following	Cloud architect, Systems administrator

Task	Description	Skills required
	command. Provide a valid Amazon SNS topic Amazon Resource Name (ARN) for the parameter. The deploymen t should successfully create CloudWatch Events rules, the Lambda function, and an IAM role with the necessary permissions, as described in the template. `aws cloudformation create-stack-set --stack-set-name my-stack-set --template-body file://firehose-encryption-checker.yaml`	

Task	Description	Skills required
Create stack instances.	Stacks can be created in the AWS Regions of your choice as well as in one or more accounts. To create stack instances, run the following command. Replace the stack name, account numbers, and Regions with your own. `aws cloudformation create-stack-instances --stack-set-name my-stack-set --accounts 123456789012 223456789012 --regions us-east-1 us-east-2 us-west-1 us-west-2 --operation-preferences FailureToleranceCount=1`	Cloud architect, Systems administrator

Related resources

- Working with AWS CloudFormation StackSets
- What is Amazon CloudWatch Events?

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automate adding or updating Windows registry entries using AWS Systems Manager

Appasaheb Bagali, Amazon Web Services

Summary

AWS Systems Manager is a remote management tool for Amazon Elastic Compute Cloud (Amazon EC2) instances. Systems Manager provides visibility and control over your infrastructure on Amazon Web Services. This versatile tool can be used to fix Windows registry changes that are identified as vulnerabilities by the security vulnerability scan report.

This pattern covers the steps to keep your EC2 instances that are running Windows operating system secure by automating registry changes that are recommended for the safety of your environment. The pattern uses the Run command to run a Command document. The code is attached, and a portion of it is included in the *Code* section.

Prerequisites and limitations

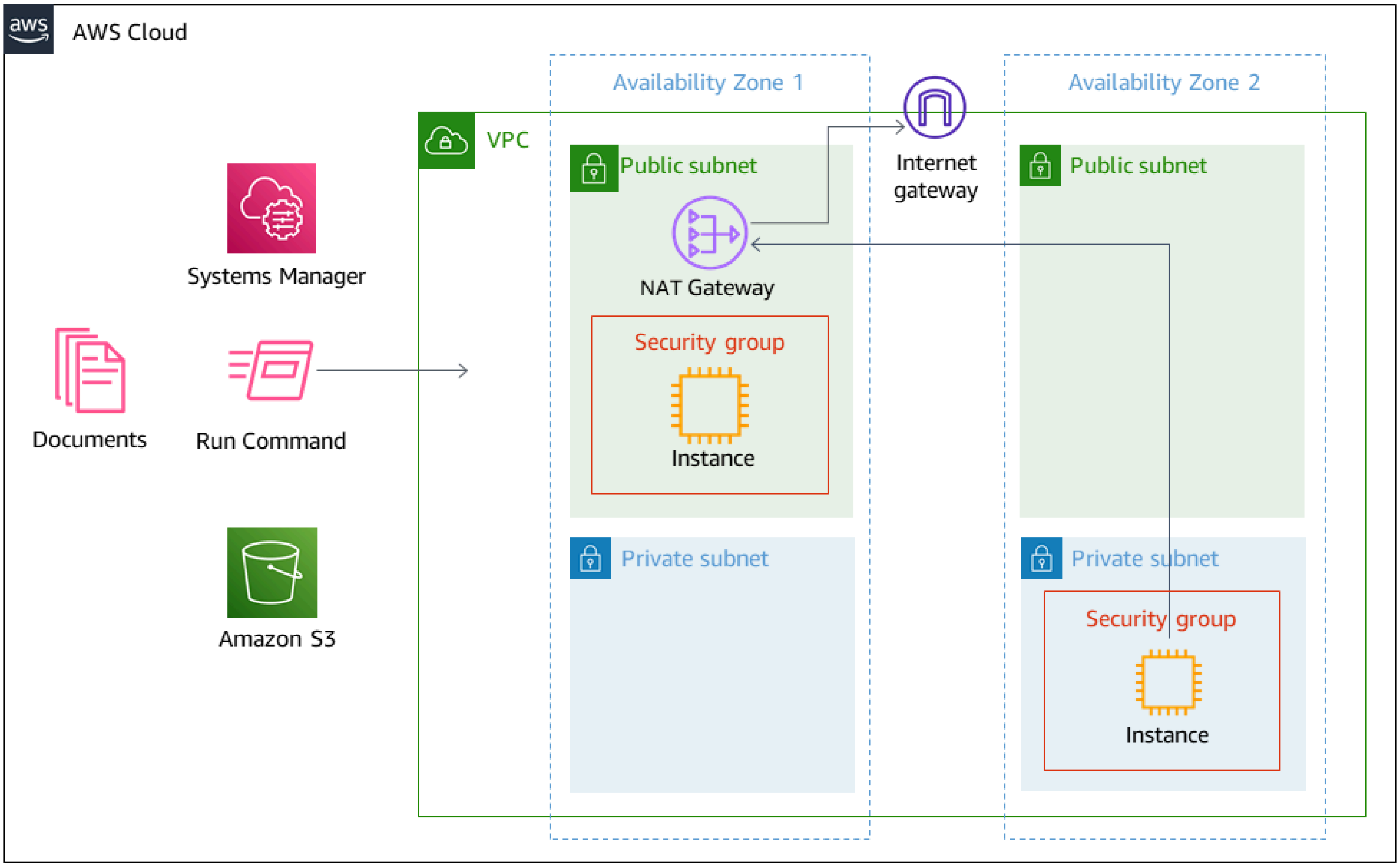
- An active AWS account
- Permissions to access the EC2 instance and Systems Manager

Architecture

Target technology stack

- A virtual private cloud (VPC), with two subnets and a network address translation (NAT) gateway
- A Systems Manager Command document to add or update the registry name and value
- Systems Manager Run Command to run the Command document on the specified EC2 instances

Target architecture



Tools

Tools

- [IAM policies and roles](#) – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- [Amazon Simple Storage Service](#) – Amazon Simple Storage Service (Amazon S3) is storage for the internet. It is designed to make web-scale computing easier for developers. In this pattern, an S3 bucket is used to store the Systems Manager logs.
- [AWS Systems Manager](#) – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Systems Manager helps you maintain security and compliance by scanning your *managed instances* and reporting (or taking corrective action on) any policy violations it detects.

- [AWS Systems Manager Command document](#) – AWS Systems Manager Command documents are used by Run Command. Most Command documents are supported on all Linux and Windows Server operating systems supported by Systems Manager.
- [AWS Systems Manager Run Command](#) – AWS Systems Manager Run Command gives you a way to manage the configuration of your managed instances remotely and securely. Using Run Command, you can automate common administrative tasks and perform one-time configuration changes at scale.

Code

You can use the following example code to add or update a Microsoft Windows registry name to `Version`, registry path to `HKCU:\Software\ScriptingGuys\Scripts`, and value to 2.

```
#Windows registry path which needs to add/update
$registryPath ='HKCU:\\Software\\ScriptingGuys\\Scripts'
#Windows registry Name  which needs to add/update
$Name = 'Version'
#Windows registry value  which needs to add/update
$value = 2
# Test-Path cmdlet to see if the registry key exists.
IF(!(Test-Path $registryPath))
        {
           New-Item -Path $registryPath -Force | Out-Null
           New-ItemProperty -Path $registryPath -Name $name -Value     $value ` -
PropertyType DWORD -                 Force | Out-        Null
        } ELSE {
                      New-ItemProperty -Path $registryPath -Name $name -Value $value `
 -PropertyType            DWORD        -Force | Out-Null
            }
echo 'Registry Path:'$registryPath
 echo 'Registry Name:'$registryPath
 echo 'Registry Value:'(Get-ItemProperty -Path $registryPath -Name $Name).version
```

The full Systems Manager Command document JavaScript Object Notation (JSON) code example is attached.

Epics

Set up a VPC

Task	Description	Skills required
Create a VPC.	On the AWS Managemen t Console, create a VPC that has public and private subnets and a NAT gateway. For more information, see the [AWS documentation]().	Cloud administrator
Create security groups.	Ensure that each security group allows access for Remote Desktop Protocol (RDP) from the source IP address.	Cloud administrator

Create an IAM policy and an IAM role

Task	Description	Skills required
Create an IAM policy.	Create an IAM policy that provides access to Amazon S3, Amazon EC2, and Systems Manager.	Cloud administrator
Create an IAM role.	Create an IAM role, and attach the IAM policy that provides access to Amazon S3, Amazon EC2, and Systems Manager.	Cloud administrator

Run the automation

Task	Description	Skills required
Create the Systems Manager Command document.	Create a Systems Manager Command document that will deploy the Microsoft Windows registry changes to add or update.	Cloud administrator
Run the Systems Manager Run Command.	Run the Systems Manager Run Command, selecting the Command document and the Systems Manager target instances. This pushes the Microsoft Windows registry change in the selected Command document to the target instances.	Cloud administrator

Related resources

- AWS Systems Manager
- AWS Systems Manager documents
- AWS Systems Manager Run Command

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automatically create an RFC in AMS using Python

Gnanasekaran Kailasam, Amazon Web Services

Summary

AWS Managed Services (AMS) helps you to operate your cloud-based infrastructure more efficiently and securely by providing ongoing management of your Amazon Web Services (AWS) infrastructure. To make a change to your managed environment, you need to create and submit a new request for change (RFC) that includes a change type (CT) ID for a particular operation or action.

However, manually creating an RFC can take around five minutes and teams in your organization might need to submit multiple RFCs every day. This pattern helps you to automate the RFC creation process, reduce the creation time for each RFC, and eliminate manual errors.

This pattern describes how to use Python code to automatically create the `Stop EC2 instance` RFC that stops Amazon Elastic Compute Cloud (Amazon EC2) instances in your AMS account. You can then apply this pattern's approach and the Python automation to other RFC types.

Prerequisites and limitations

Prerequisites

- An AMS Advanced account. For more information about this, see [AMS operations plans](#) in the AWS Managed Services documentation.
- At least one existing EC2 instance in your AMS account.
- An understanding of how to create and submit RFCs in AMS.
- Familiarity with Python.

Limitations

- You can only use RFCs for changes in your AMS account. Your AWS account uses different processes for similar changes.

Architecture

Technology stack

- AMS
- AWS Command Line Interface (AWS CLI)
- AWS SDK for Python (Boto3)
- Python and its required packages (JSON and Boto3)

Automation and scale

This pattern provides sample code to automate the `Stop EC2 instance` RFC, but you can use this pattern's sample code and approach for other RFCs.

Tools

- AWS Managed Services – AMS helps you to operate your AWS infrastructure more efficiently and securely.
- AWS CLI – AWS Command Line Interface (AWS CLI) is a unified tool to manage your AWS services. In AMS, the change management API provides operations to create and manage RFCs.
- AWS SDK for Python (Boto3) – SDK for Python makes it easy to integrate your Python application, library, or script with AWS services.

Code

The `AMS Stop EC2 Instance.zip` file (attached) contains the Python code for creating a `Stop EC2 instance` RFC. You can also configure this code to submit a single RFC for multiple EC2 instances.

Epics

Option 1 – Set up environment for macOS or Linux

Task	Description	Skills required
Install and validate Python.	1. Open a terminal window and run the `brew install python3` command. 2. Validate that Python is correctly installed by	AWS systems administrator

Task	Description	Skills required
	running the `python --version` command. 3. Validate that `pip` is correctly installed by running the `pip --version` command.	
Install AWS CLI.	Run the `pip install awscli --upgrade –user` command to install AWS CLI.	AWS systems administrator
Install Boto3.	Run the `pip install boto3` command to install Boto3.	AWS systems administrator
Install JSON.	Run the `pip install json` command to install JSON.	AWS systems administrator
Set up AMS CLI.	Sign in to the AWS Management Console, open the AMS console, and then choose **Documentation**. Download the .zip file that contains the AMS CLI, unzip it, and then install it on your local machine. After you install AMS CLI, run the `aws amscm help` command. The output provides information about the AMS change managemen t process.	AWS systems administrator

Option 2 – Set up environment for Windows

Task	Description	Skills required
Install and validate Python.	1. Open the [Python releases for Windows](#) page, download the latest version, and then install Python. 2. Validate that Python is correctly installed by running the `python --version` command. 3. Validate that `pip` is correctly installed by running the `pip --version` command.	AWS systems administrator
Install AWS CLI.	Run the `pip install awscli --upgrade –user` command to install AWS CLI.	AWS systems administrator
Install Boto3.	Run the `pip install boto3` command to install Boto3.	AWS systems administrator
Install JSON.	Run the `pip install json` command to install JSON.	AWS systems administrator
Set up AMS CLI.	Sign in to the AWS Management Console, open the AMS console, and then choose **Documentation**. Download the .zip file that contains the AMS CLI, unzip it, and then install it on your local machine.	AWS systems administrator

Task	Description	Skills required
	After you install AMS CLI, run the `aws amscm help` command. The output provides information about the AMS change managemen t process	

Extract the CT ID and execution parameters for the RFC

Task	Description	Skills required
Extract the CT ID, version, and execution parameters for the RFC.	Each RFC has a different CT ID, version, and execution parameters. You can extract this information by using one of the following options: 1. Follow the instructions from the *Finding a request for change (RFC) with the CLI* section in [RFC use examples](#) from the AWS Managed Services documentation. 2. Open an existing RFC of a similar type or create new RFC as a test through the AMS console. Use the RFC's CT ID and execution parameters. For more information about this, see [Finding an RFC with the console](#) in the	AWS systems administrator

Task	Description	Skills required
	AWS Managed Services documentation. **Note** To adapt this pattern's Python automation for other RFCs, replace the CT type and parameter values in the `ams_stop_ec2_instance` Python code file from the `AMS Stop EC2 Instance.zip` file (attached) with those that you extracted.	

Run the Python automation

Task	Description	Skills required
Run the Python automation.	1. Download the `AMS Stop EC2 Instance.zip` file (attached) to your local machine and extract the file. 2. Update `input_instances` with your EC2 instance information. 3. Open a terminal and navigate to the path for your extracted code	AWS systems administrator

Task	Description	Skills required
	4. Run the `pythonams_stop_ec2_instance.py` command.	

Related resources

- What are change types?
- CLI tutorial: High availability two-tier stack (Linux/RHEL)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automatically stop and start an Amazon RDS DB instance using AWS Systems Manager Maintenance Windows

Ashita Dsilva, Amazon Web Services

Summary

This pattern demonstrates how to automatically stop and start an Amazon Relational Database Service (Amazon RDS) DB instance on a specific schedule (for example, shutting down a DB instance outside of business hours to reduce costs) by using AWS Systems Manager Maintenance Windows. For this purpose, Systems Manager is cost-effective for typical use cases.

AWS Systems Manager Automation provides the `AWS-StopRdsInstance` and `AWS-StartRdsInstance` runbooks to stop and start Amazon RDS DB instances. This means that you don't need to write custom logic with AWS Lambda functions or create an Amazon CloudWatch Events rule.

Systems Manager provides two capabilities for scheduling tasks: [State Manager] and [Maintenance Windows]. State Manager sets and maintains the required state configuration for resources in your Amazon Web Services (AWS) account one time or on a specific schedule. Maintenance Windows runs tasks on the resources in your account during a specific time window. Although you can use this pattern's approach with State Manager or Maintenance Windows, we recommend that you use Maintenance Windows because it can run one or more tasks based on assigned priority and can also run AWS Lambda functions and AWS Step Functions tasks. For more information about State Manager and Maintenance Windows, see [Choosing between State Manager and Maintenance Windows] in the Systems Manager documentation.

This pattern provides detailed steps to configure two separate maintenance windows that use cron expressions to stop and then start an Amazon RDS DB instance.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An existing Amazon RDS DB instance that you want to stop and start on a specific schedule.
- Cron expressions for your required schedule. For example, the expression `cron(0 9 ? * MON-FRI *)` runs the task at 09:00 on every Monday, Tuesday, Wednesday, Thursday, and Friday.

For more information, see Cron and rate expressions for maintenance windows in the Systems Manager documentation.

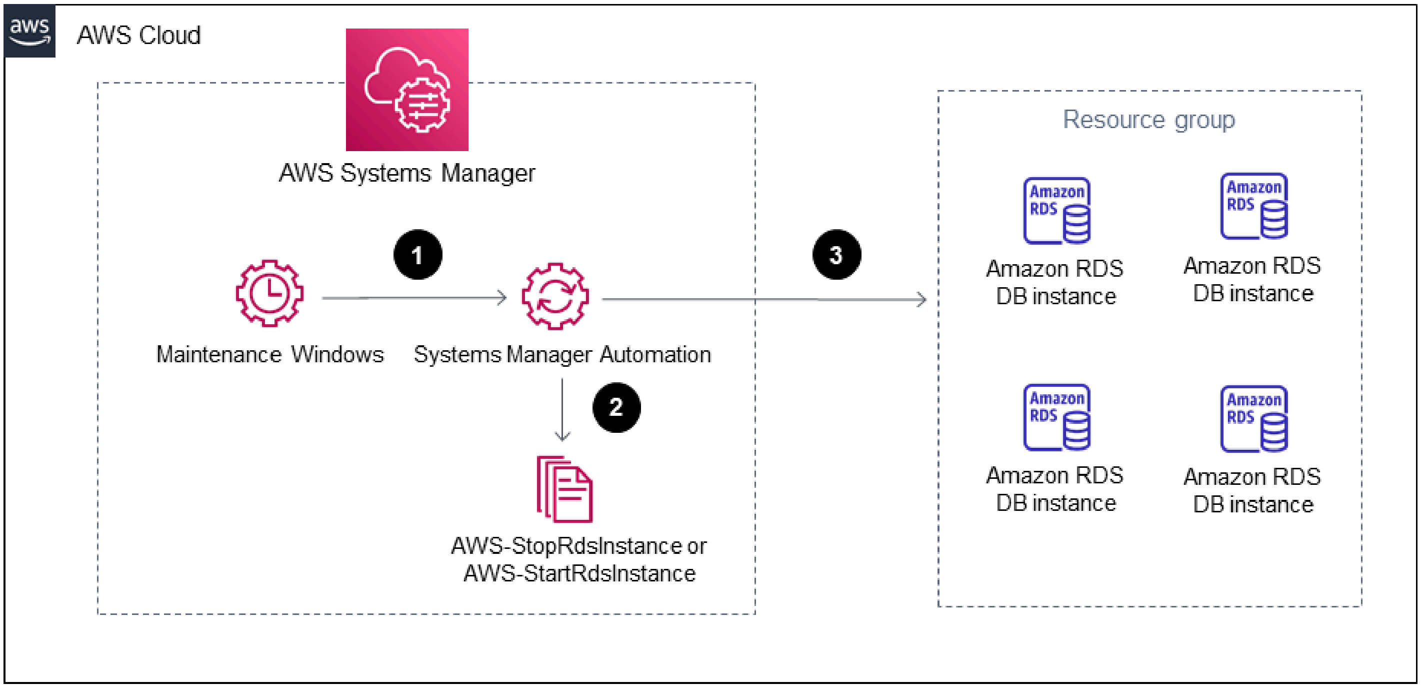
- Familiarity with Systems Manager.
- Permissions to start and stop the RDS instance. For more information, see the Epics section.

Limitations

- An Amazon RDS DB instance can be stopped for up to seven days at one time. After seven days, the DB instance automatically restarts to ensure that it receives any required maintenance updates.
- You can't stop a DB instance that is a read replica or that has a read replica.
- You can't stop an Amazon RDS for SQL Server DB instance in a Multi-AZ configuration.
- Service quotas apply to Maintenance Windows and Systems Manager Automation. For more information about service quotas, see AWS Systems Manager endpoints and quotas in the AWS General Reference documentation.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

The following diagram shows the workflow to automatically stop and start an Amazon RDS DB instance.



The workflow has the following steps:

1. Create a maintenance window and use cron expressions to define the stop and start schedule for your Amazon RDS DB instances.

2. Register a Systems Manager Automation task to the maintenance window by using the `AWS-StopRdsInstance` or `AWS-StartRdsInstance` runbook.

3. Register a target with the maintenance window by using a tag-based resource group for your Amazon RDS DB instances.

Technology stack

- AWS CloudFormation
- AWS Identity and Access Management (IAM)
- Amazon RDS
- Systems Manager

Automation and scale

You can stop and start multiple Amazon RDS DB instances at the same time by tagging the required Amazon RDS DB instances, creating a resource group that includes all the tagged DB instances, and registering this resource group as a target for the maintenance window.

Tools

- AWS CloudFormation is a service that helps you model and set up your AWS resources.
- AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources.
- Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the AWS Cloud.
- AWS Resource Groups helps you organize AWS resources into groups, tag resources, and manage, monitor, and automate tasks on grouped resources.
- AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. This pattern uses the following features of Systems Manager:
  - AWS Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon Elastic Compute Cloud (Amazon EC2) instances and other AWS resources.
  - AWS Systems Manager Maintenance Windows helps you define a schedule for when to perform potentially disruptive actions on your instances.

Epics

Create and configure the IAM service role for Systems Manager Automation

Task	Description	Skills required
Configure the IAM service role for Systems Manager Automation.	Sign in to the AWS Management Console and create a service role for Systems Manager Automatio n. You can use one of the following two methods to create this service role: • Use AWS CloudFormation to configure a service	AWS administrator

Task	Description	Skills required

	role for Systems Manager Automation
	• Use IAM to configure roles for Systems Manager Automation
	
	The Systems Manager Automation workflow invokes Amazon RDS by using a service role to perform start and stop actions on the Amazon RDS DB instance.
	
	The service role must be configured with the following inline policy that has permissions to start and stop the Amazon RDS DB instance:
	
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
	`            "Sid": "RdsStartStop",`
	`            "Effect": "Allow",`
	`            "Action": [`
	`                "rds:StopDBInstance",`
	`                "rds:StartDBInstance"`
	`            ],`
	`            "Resource": "<RDS_Instance_ARN>"`	

Task	Description	Skills required

	``
	        },
	        {
	            "Sid":
	 "RdsDescribe",
	            "Effect":
	 "Allow",
	            "Action":
	 "rds:DescribeDBIns
	tances",
	            "Resource":
	 "*"
	        }
	    ]
	}
	``
	
	Make sure that you replace `<RDS_Instance_ARN>` with the Amazon Resource Name (ARN) of your Amazon RDS DB instance.
	
	If you are unfamiliar with using IAM policies and roles, follow the instructions in the *Solution Overview* section of the [Schedule Amazon RDS stop and start using AWS Systems Manager](#) blog post.
	
	Important
	Make sure that you record the ARN of the service role.	

Create a resource group

Task	Description	Skills required
Tag the Amazon RDS DB instances.	Open the [Amazon RDS console](#) and tag the Amazon RDS DB instances that you want to add to the resource group. A tag is metadata assigned to an AWS resource and consists of a key-value pair. We recommend that you use *Action* as the **Tag key** and *StartStop* as the **Value**. For more information about this, see [Adding, listing, and removing tags](#) in the Amazon RDS documentation.	AWS administrator
Create a resource group for your tagged Amazon RDS DB instances.	Open the [AWS Resource Groups console](#) and create a resource group based on the tag that you created for your Amazon RDS instances. Under **Grouping Criteria**, make sure that you choose **AWS::RDS::DBInstance** for the resource type and then provide the tag's key-value pair (for example, "Action-StartStop"). This ensures that the service only checks for Amazon RDS DB instances and not other resources that have this tag. Make sure	AWS administrator

Task	Description	Skills required
	that you record the resource group's name. For more information and detailed steps, see [Build a tag-based query and create a group](#) in the AWS Resource Groups documentation.	

Configure a maintenance window to stop the Amazon RDS DB instances

Task	Description	Skills required
Create a maintenance window.	1. Open the [Systems Manager console](#), choose **Maintenance Windows**, and then choose **Create a maintenance window**. Provide a name for your maintenance window (for example, "StopRdsI nstance"), enter a descripti on, and then uncheck **Allow unregistered targets**. 2. Choose **CRON/Rate expression** and provide the schedule expressio n to define when the Amazon RDS DB instances should be stopped. Enter **1** for the **Duration** and **0** for **Stop initiating tasks**. By default, the **Time**	AWS administrator

Task	Description	Skills required
	zone shows UTC. You can change the time zone to initiate the maintenance window based on the timestamp defined in your cron expression. 3. Choose **Create maintenance window**. The system returns you to the maintenance window page and the state of your maintenance window is **Enabled**. **Important** The task to stop the DB instance runs almost instantly when initiated and doesn't span the entire duration of the maintenance window. This pattern provides the minimum values for **Duration** and **Stop initiating tasks** because they are the required parameters for a maintenance window.	

Task	Description	Skills required
	For more information and detailed steps, see [Create a maintenance window (console)]() in the Systems Manager documentation.	
Assign a target to the maintenance window.	1. On the [Systems Manager console](), choose **Maintenance Windows**, choose **Actions**, and then choose **Register targets**. 2. In the **Targets** area, specify **Choose a resource group** and then choose the name of an existing resource group in your account. 3. For **Resource types**, choose **AWS::RDS::DBInstance** and then choose **Register target**. For more information and detailed steps, see [Assign targets to a maintenance window (console)]() in the Systems Manager documentation.	AWS administrator

Task	Description	Skills required
Assign a task to the maintenance window.	1. On the [Systems Manager console](), choose **Maintenance Windows**, and then choose your maintenance window. Choose **Actions**, and then choose **Register Automation task**. 2. For **Document**, choose **AWS-StopRdsInstance**. 3. In the **Targets** section, choose **Selecting registered target groups**, and then choose the maintenance window target that you registered with the current maintenance window. 4. For **Rate control**, specify 100 percent for **Concurrency** and **Error threshold**. You can change the **Rate control** values according to your requirements for task concurrency and error threshold. For more information about this, see [About concurrency and error thresholds]() in the Systems Manager documentation. 5. In the **IAM service role** section, for **Service role**, leave this box blank or create your own custom role. If you leave the box	AWS administrator

Task	Description	Skills required
	blank, Systems Manager automatically creates the service-linked role **AWSServiceRoleForAmazonSSM** and then associates the role with the task. To create your own custom role, see [Create a custom service role for maintenance windows (console)](), and then associate that custom role with the task. 6. In the **Input Parameters** section, specify the following parameters for the runbook: • **InstanceId**: `{{RESOURCE_ID}}` **Note** For **InstanceId**, a pseudo parameter is used to extract the Amazon RDS DB resource ID from the ARN. To learn more about pseudo parameters, see [About pseudo parameters]() in the Systems	

Task	Description	Skills required
	Manager documentation. • **AutomationAssumeRole**: Provide the ARN of the service role that you created for Systems Manager Automation. 7. Choose **Register Automation task**. **Important** The **Service role** option defines the service role required for the maintenance window to run tasks. However, this role is not identical to the service role that you created earlier for Systems Manager Automation. For more information and detailed steps, see Assign tasks to a maintenance window (console) in the Systems Manager documentation.	

Configure a maintenance window to start the Amazon RDS DB instances

Task	Description	Skills required
Configure a maintenance window to start the Amazon RDS DB instances.	Repeat the steps from the *Configure a maintenance window to stop the Amazon RDS DB instances* epic to configure another maintenance window to start the Amazon RDS DB instances at a scheduled time. **Important** You must make the following changes when you configure the maintenance window to start the DB instances: • Use a new name for the maintenance window (for example, "StartRdsInstance"). • Replace the cron expression with the cron expression that you want to use to start the DB instances. • Replace the `AWS-StopRdsInstance` runbook with `AWS-StartRdsInstance` in **Task**.	AWS administrator

Related resources

- Use Systems Manager Automation documents to manage instances and cut costs off-hours (AWS blog post)

Centralize software package distribution in AWS Organizations by using Terraform

Pradip kumar Pandey, Chintamani Aphale, T.V.R.L.Phani Kumar Dadi, Pratap Kumar Nanda, Aarti Rajput, and Mayuri Shinde, Amazon Web Services

Summary

Enterprises often maintain multiple AWS accounts that are spread across multiple AWS Regions in order to create a strong isolation barrier between workloads. To stay secure and compliant, their administration teams install agent-based tools such as CrowdStrike, SentinelOne, or TrendMicro tools for security scanning, and the Amazon CloudWatch agent, Datadog Agent, or AppDynamics agents for monitoring. These teams often face challenges when they want to centrally automate software package management and distribution across this large landscape.

Distributor, a capability of AWS Systems Manager, automates the process of packaging and publishing software to managed Microsoft Windows and Linux instances across the cloud and on-premises servers through a single simplified interface. This pattern demonstrates how you can use Terraform to further simplify the process of managing the installation of software and to run scripts across a large number of instances and member accounts within AWS Organizations with minimal effort.

This solution works for Amazon, Linux, and Windows instances that are managed by Systems Manager.

Prerequisites and limitations

- A Distributor package that has the software to be installed
- Terraform version 0.15.0 or later
- Amazon Elastic Compute Cloud (Amazon EC2) instances that are managed by Systems Manager and have basic permissions to access Amazon Simple Storage Service (Amazon S3) in the target account
- A landing zone for your organization that's set up by using AWS Control Tower
- (Optional) Account Factory for Terraform (AFT)

Architecture

Resource details

This pattern uses [Account Factory for Terraform (AFT)](#) to create all required AWS resources and the code pipeline to deploy the resources in a deployment account. The code pipeline runs in two repositories:

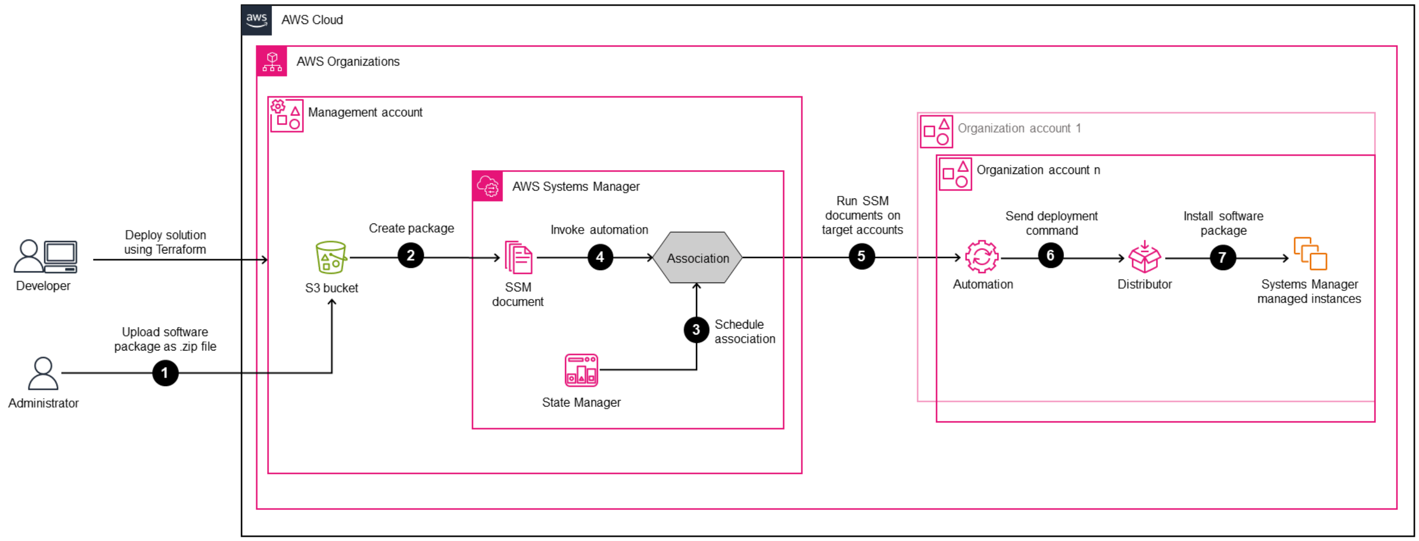
- **Global customization** contains Terraform code that will run across all accounts registered with AFT.
- **Account customizations** contains Terraform code that will run in the deployment account.

You can also deploy this solution without using AFT, by running [Terraform](#) commands in the account customizations folder.

The Terraform code deploys the following resources:

- AWS Identity and Access Management (IAM) role and policies
  - [SystemsManager-AutomationExecutionRole](#) grants the user permissions to run automations in the target accounts.
  - [SystemsManager-AutomationAdministrationRole](#) grants the user permissions to run automations in multiple accounts and organizational units (OUs).
- Compressed files and manifest.json for the package
  - In Systems Manager, a [package](#) includes at least one .zip file of software or installable assets.
  - The JSON manifest includes pointers to your package code files.
- S3 bucket
  - The distributed package that is shared across the organization is securely stored in an Amazon S3 bucket.
- AWS Systems Manager documents (SSM documents)
  - `DistributeSoftwarePackage` contains the logic to distribute the software package to every target instance in the member accounts.
  - `AddSoftwarePackageToDistributor` contains the logic to package the installable software assets and add it to Automation, a capability of AWS Systems Manager.
- Systems Manager association
  - A Systems Manager association is used to deploy the solution.

Architecture and workflow



The diagram illustrates the following steps:

1. To run the solution from a centralized account, you upload your packages or software along with deployment steps to an S3 bucket.
2. Your customized package becomes available in the Systems Manager console [Documents](#) section, in the **Owned by me** tab.
3. State Manager, a capability of Systems Manager, creates, schedules, and runs an association for the package across the organization. The association specifies that the software package must be installed and running on a managed node before it can be installed on the target node.
4. The association instructs Systems Manager to install the package on the target node.
5. For any subsequent installations or changes, users can run the same association periodically or manually from a single location to perform deployments across accounts.
6. In member accounts, Automation sends deployment commands to Distributor.
7. Distributor distributes software packages across instances.

This solution uses the management account within AWS Organizations, but you can also designate an account (delegated administrator) to manage this on behalf of the organization.

Tools

AWS services

- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data. This pattern uses Amazon S3 to centralize and securely store the distributed package.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale. This pattern uses the following Systems Manager capabilities:
  - Distributor helps you package and publish software to Systems Manager managed instances.
  - Automation simplifies common maintenance, deployment, and remediation tasks for many AWS services.
  - Documents performs actions on your Systems Manager managed instances across your organization and accounts.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The instructions and code for this pattern are available in the GitHub Centralized package distribution repository.

Best practices

- To assign tags to an association, use the AWS Command Line Interface (AWS CLI) or the AWS Tools for PowerShell. Adding tags to an association by using the Systems Manager console isn't supported. For more information, see Tagging Systems Manager resources in the Systems Manager documentation.
- To run an association by using a new version of a document shared from another account, set the document version to `default`.
- To tag only the target node, use one tag key. If you want to target your nodes by using multiple tag keys, use the resource group option.

Epics

Configure source files and accounts

Task	Description	Skills required

Clone the repository.	1. Clone the GitHub [Centralized package distribution](https://github.com/aws-samples/aws-organization-centralised-package-distribution) repository:
	`git clone https://github.com/aws-samples/aws-organization-centralised-package-distribution`
	2. The Terraform code repository requires two customization folders that are managed by AFT. Confirm that your local copy of the repository contains these folders:
	`$ cd centralised-package-distribution`
	`$ ls`
	`global-customization account-customization`
		DevOps engineer

Update global variables.	Update the following input parameters in the `global-customization/variables.tf` file. These variables apply to all accounts that are created and managed by AFT.	DevOps engineer

Task	Description	Skills required
	• `account_id` : The ID of the account where the Distributor solution will be deployed. • `aws_region` : The AWS Region where the associati on will be deployed.	
Update account variables.	Update the following input parameters in the `account-customizat ion/variables.tf` file. These variables apply only to specific accounts that are created and managed by AFT. • `package_bucket_nam e` : The name of the S3 bucket that contains the package distribution file. • `package_name` : The name of the package distribution file. • `package_version` : The package version of the installer.	DevOps engineer

Customize parameters and deployment files

Task	Description	Skills required
Update input parameters for the State Manager associati on.	Update the following input parameters in the `account- customization/assoc`	DevOps engineer

Task	Description	Skills required

	`iation.tf` file to define the state you want to maintain on your instances . You can use the default parameter values if they support your use case.
	
	• `targetAccounts` : The organizational unit (OU) IDs within AWS Organizations that represent accounts with the target instances for distribution. OU IDs start with "ou".
	• `targetRegions` : The AWS Regions (for example, "us-east-1” or "ap-south east-2") where the target instances are running.
	• `action`: Specify whether to install or uninstall the package.
	• `installationType` : One of the following installation types:
	  • `uninstall` : The package is uninstalled.
	  • `reinstall` : The application is taken offline until the reinstall ation process is complete.
	  • `In-place update`: The application is available while new or updated	

Task	Description	Skills required
	files are added to the installation. • `name`: The name of the package to install or uninstall. • `version`: The version of the package to install or uninstall. If no version of the package is installed, the system returns an error. • `bucketName` : The S3 bucket name the package has been deployed to. This bucket should consist of the packages and the manifest file only. • `bucketPrefix` : The S3 prefix where the package assets are stored. • `AutomationAssumeRole` : The Amazon Resource Name (ARN) of `SystemsManager-AutomationAdministrationRole` .	

Task	Description	Skills required
Prepare compressed files and the `manifest.json` file for the package.	This pattern provides sample PowerShell installable files (.msi for Windows and .rpm for Linux) with install and uninstall scripts in the `account-customization/package` folder. 1. Replace the PowerShell installable files with your own files, or provide your installable file, install and uninstall scripts, and manifest file to create a package in the `account-customization` folder in your account. 2. Customize the default `manifest.json` file that Terraform generates in the `account-customization` folder according to your requirements.	DevOps engineer

Run Terraform commands to provision resources

Task	Description	Skills required
Initialize the Terraform configuration.	To deploy the solution automatically with AFT, push the code to AWS CodeCommit: `$ git add *`	DevOps engineer

Task	Description	Skills required

	`$ git commit -m "message"`
	`$ git push`
	
	You can also deploy this solution without using AFT by running a Terraform command from the `account-customization` folder. To initialize the working directory that contains the Terraform files, run:
	
	`$ terraform init`	

Preview changes.	To preview the changes that Terraform will make to the infrastructre, run the command: `$ terraform plan` This command evaluates the Terraform configuration to determine the desired state of the resources that have been declared. It also compares the desired state with the actual infrastructure to provision within the workspace.	DevOps engineer

Task	Description	Skills required
Apply changes.	Run the following command to implement the changes that you made to the `variables.tf` files: `$ terraform apply`	DevOps engineer

Validate resources

Task	Description	Skills required
Validate the creation of SSM documents.	1. On the [Systems Manager console](https://console.aws.amazon.com/systems-manager/), in the left navigation pane, choose **Documents**. 2. Choose the **Owned by me** tab. You should see the `DistributeSoftwarePackage` and `AddSoftwarePackageToDistributor` packages.	DevOps engineer
Validate the successful deployment of automations.	1. On the Systems Manager console, in the left navigation pane, choose **Automation**. 2. In the **Automation executions** list, you should see the most recent `DistributeSoftwarePackage` and `AddSoftwa`	DevOps engineer

Task	Description	Skills required
	`rePackageToDistributor` deployments. 3. Choose **Execution ID** to validate that they completed successfully.	
Validate that the package deployed to the targeted member account instances.	1. On the Systems Manager console, in the navigation pane, choose **Run Command**. 2. In **Command history**, you will see each invocation and its status. 3. Choose any **Command ID** to see the deployment history for each target instance. 4. Choose the **Instance ID** and check the **Output** section for the distribution.	DevOps engineer

Troubleshooting

Issue	Solution
The State Manager association failed or is stuck in pending status.	See the troubleshooting information in the AWS Knowledge Center.
A scheduled association failed to run.	Your schedule specification might be invalid. State Manager doesn't currently support specifying months in cron expressions for associations. Use cron or rate expressions to confirm the schedule.

Related resources

- Centralized package distribution (GitHub repository)
- Account Factory for Terraform (AFT)
- Use cases and best practices (AWS Systems Manager documentation)

Configure logging for .NET applications in Amazon CloudWatch Logs by using NLog

Bibhuti Sahu and Rob Hill (AWS), Amazon Web Services

Summary

This pattern describes how to use the NLog open-source logging framework to log .NET application usage and events in Amazon CloudWatch Logs. In the CloudWatch console, you can view the application's log messages in near real time. You can also set up metrics and configure alarms to notify you if a metric threshold is exceeded. Using CloudWatch Application Insights, you can view automated or custom dashboards that show potential problems for the monitored applications. CloudWatch Application Insights is designed to help you quickly isolate ongoing issues with your applications and infrastructure.

To write log messages to CloudWatch Logs, you add the `AWS.Logger.NLog` NuGet package to the .NET project. Then, you update the `NLog.config` file to use CloudWatch Logs as a target.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A .NET web or console application that:
  - Uses supported .NET Framework or .NET Core versions. For more information, see *Product versions*.
  - Uses NLog to send log data to Application Insights.
- Permissions to create an IAM role for an AWS service. For more information, see Service role permissions.
- Permissions to pass a role to an AWS service. For more information, see Granting a user permissions to pass a role to an AWS service.

Product versions

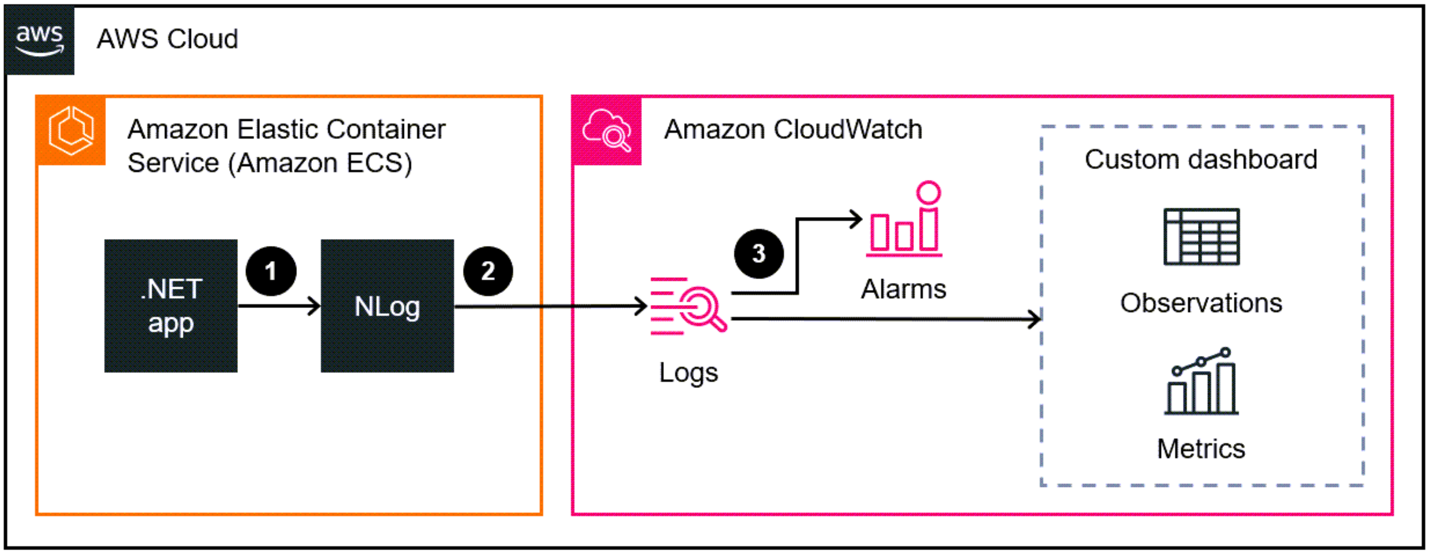
- .NET Framework version 3.5 or later
- .NET Core versions 1.0.1, 2.0.0, or later

Architecture

Target technology stack

- NLog
- Amazon CloudWatch Logs

Target architecture



1. The .NET application writes log data to the NLog logging framework.
2. NLog writes the log data to CloudWatch Logs.
3. You use CloudWatch alarms and custom dashboards to monitor the .NET application.

Tools

AWS services

- [Amazon CloudWatch Application Insights](#) helps you observe the health of your applications and underlying AWS resources.
- [Amazon CloudWatch Logs](#) helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- AWS Tools for PowerShell are a set of PowerShell modules that help you script operations on your AWS resources from the PowerShell command line.

Other tools

- Logger.NLog is an NLog target that records log data to CloudWatch Logs.
- NLog is an open-source logging framework for .NET platforms that helps you write log data to targets, such as databases, log files, or consoles.
- PowerShell is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS.
- Visual Studio is an integrated development environment (IDE) that includes compilers, code completion tools, graphical designers, and other features that support software development.

Best practices

- Set a retention policy for the target log group. This must be done outside of the NLog configuration. By default, log data is stored in CloudWatch Logs indefinitely.
- Adhere to the Best practices for managing AWS access keys.

Epics

Set up access and tools

Task	Description	Skills required
Create an IAM policy.	Follow the instructions in Creating policies using the JSON editor in the IAM documentation. Enter the following JSON policy, which has the least-privilege permissions necessary to allow CloudWatch Logs to read and write logs. `{`	AWS administrator, AWS DevOps

Task	Description	Skills required

	``
	    "Version":
	 "2012-10-17",
	    "Statement": [
	        {
	            "Effect":
	 "Allow",
	            "Action": [
	
	 "logs:CreateLogGro
	up",
	
	 "logs:CreateLogStr
	eam",
	
	 "logs:GetLogEvents",
	
	 "logs:PutLogEvents",
	
	 "logs:DescribeLogG
	roups",
	
	 "logs:DescribeLogS
	treams",
	
	 "logs:PutRetention
	Policy"
	            ],
	            "Resource":
	 [
	                "*"
	            ]
	        }
	    ]
	}
	``	

Task	Description	Skills required
Create an IAM role.	Follow the instructions in [Creating a role to delegate permissions to an AWS service](#) in the IAM documentation. Select the policy that you created previously. This is the role CloudWatch Logs assumes to perform logging actions.	AWS administrator, AWS DevOps
Set up AWS Tools for PowerShell.	1. Follow the instructions for your operating system in [Installing the AWS Tools for PowerShell](#). 2. Use the AWS Tools for PowerShell cmdlets to store your access key and secret key in a profile. For instructions, see [Managing Profiles](#) in the AWS Tools for PowerShell documenta tion.	General AWS

Configure NLog

Task	Description	Skills required
Install the NuGet package.	1. In Visual Studio, choose **File**, and then choose **Open a project or solution**. 2. Choose the project where you want to install NLog.	App developer

Task	Description	Skills required
	3. In Visual Studio, choose **Tools**, **NuGet Package Manager**, **Package Manager Console**. 4. Install the `AWS.Logger.NLog` NuGet package by entering the following command. `Install-Package AWS.Logger.NLog -Version 3.1.0`	

Task	Description	Skills required
Configure the logging target.	1. Open the `NLog.config` file. 2. For the target `type`, enter `AWSTarget` . 3. For the target `logGroup`, enter the name of the [log group](https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/Working-with-log-groups-and-streams.html) that you want to use. If the log group doesn't already exist, a new log group with the provided name is automatically created. 4. For the target `region`, enter the AWS Region where CloudWatch Logs is configured. 5. For the target `profile`, enter the name of the profile that you created previously to store to store the access key and secret key. 6. Save and close the `NLog.config` file. For a sample configuration file, see the Additional information section of this pattern. When you run your application, NLog will write the log messages and send them to CloudWatch Logs.	App developer

Validate and monitor logs

Task	Description	Skills required
Validate logging.	Follow the instructions in View log data sent to CloudWatch Logs in the CloudWatch Logs documenta tion. Validate that log events are being recorded for the .NET application. If log events are not being recorded, see the Troublesh ooting section in this pattern.	General AWS
Monitor the .NET application stack.	Configure monitoring in CloudWatch as needed for your use case. You can use CloudWatch Logs Insights, CloudWatch Metrics Insights, and CloudWatch Application Insights to monitor your .NET workload. You can also configure alarms so that you can receive alerts, and you can create a custom dashboard for monitoring the workload from a single view.	General AWS

Troubleshooting

Issue	Solution
Log data doesn't appear in CloudWatch Logs.	Make sure that the IAM policy is attached to the IAM role that CloudWatch Logs assumes.

Issue	Solution
	For instructions, see the *Set up access and tools* section in the Epics section.

Related resources

- Working with log groups and log streams (CloudWatch Logs documentation)
- Amazon CloudWatch Logs and .NET Logging Frameworks (AWS blog post)

Additional information

The following is a sample `NLog.config` file.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="nlog" type="NLog.Config.ConfigSectionHandler, NLog" />
  </configSections>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <nlog>
    <extensions>
      <add assembly="NLog.AWS.Logger" />
    </extensions>
    <targets>
      <target name="aws" type="AWSTarget" logGroup="NLog.TestGroup" region="us-east-1"
 profile="demo"/>
    </targets>
    <rules>
      <logger name="*" minlevel="Info" writeTo="aws" />
    </rules>
  </nlog>
</configuration>
```

Copy AWS Service Catalog products across different AWS accounts and AWS Regions

Sachin Vighe and Santosh Kale, Amazon Web Services

Summary

AWS Service Catalog is a Regional service and this means that AWS Service Catalog portfolios and products are only visible in the AWS Region where they are created. If you set up an AWS Service Catalog hub in a new Region, you must recreate your existing products and this can be a time-consuming process.

This pattern's approach helps simplify this process by describing how to copy products from an AWS Service Catalog hub in a source AWS account or Region to a new hub in a destination account or Region. For more information about the AWS Service Catalog hub and spoke model, see AWS Service Catalog hub and spoke model: How to automate the deployment and management of AWS Service Catalog to many accounts on the AWS Management and Governance Blog.

The pattern also provides the separate code packages required to copy AWS Service Catalog products across accounts or to other Regions. By using this pattern, your organization can save time, make existing and previous product versions available in a new AWS Service Catalog hub, minimize the risk of manual errors, and scale the approach across multiple accounts or Regions.

> **Note**
>
> This pattern's *Epics* section provides two options for copying products. You can use Option 1 to copy products across accounts or choose Option 2 to copy products across Regions.

Prerequisites and limitations

Prerequisites

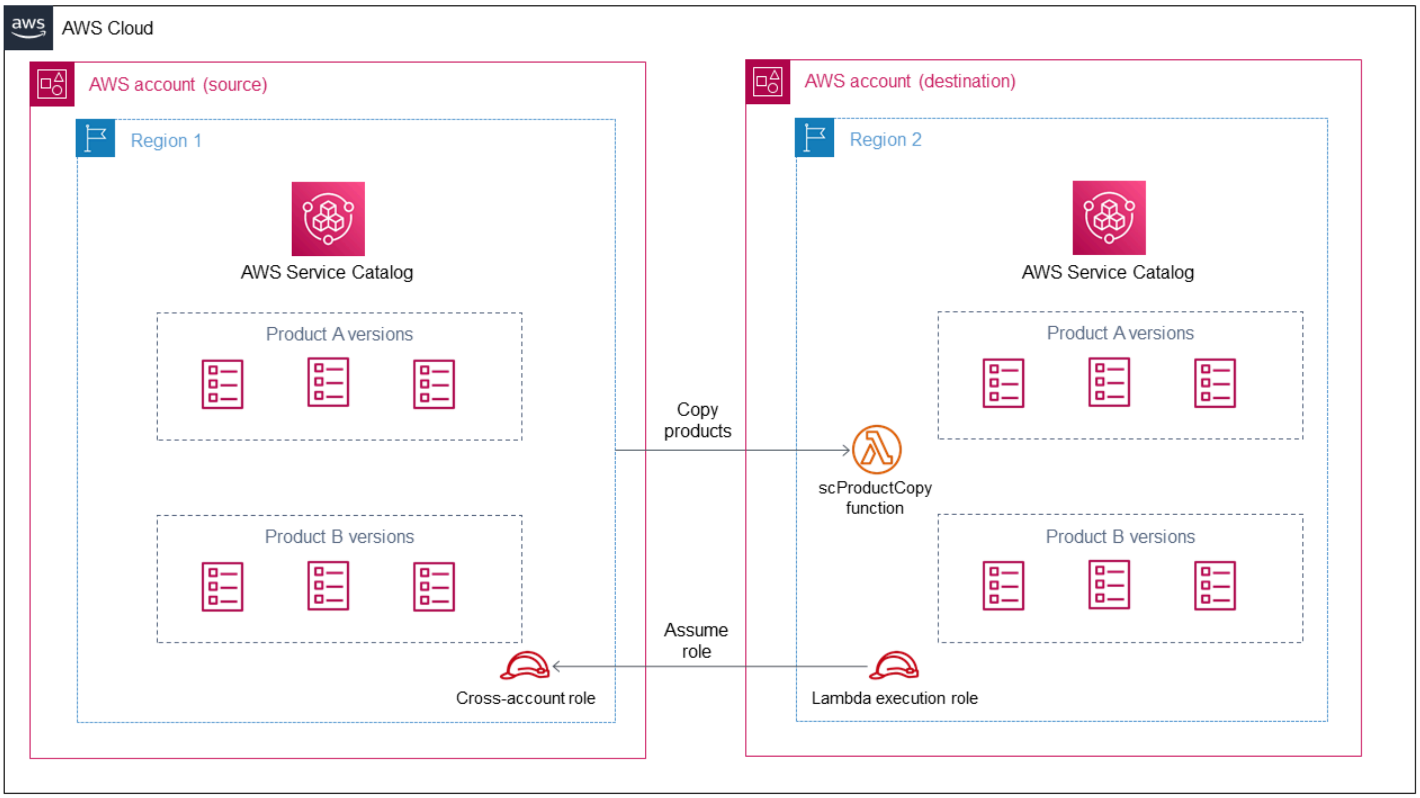
- An active AWS account.
- Existing AWS Service Catalog products in a source account or Region.
- An existing AWS Service Catalog hub in a destination account or Region.
- If you want to copy products across accounts, you must share and then import the AWS Service Catalog portfolio containing the products into your destination account. For more information about this, see Sharing and importing portfolios in the AWS Service Catalog documentation.

Limitations

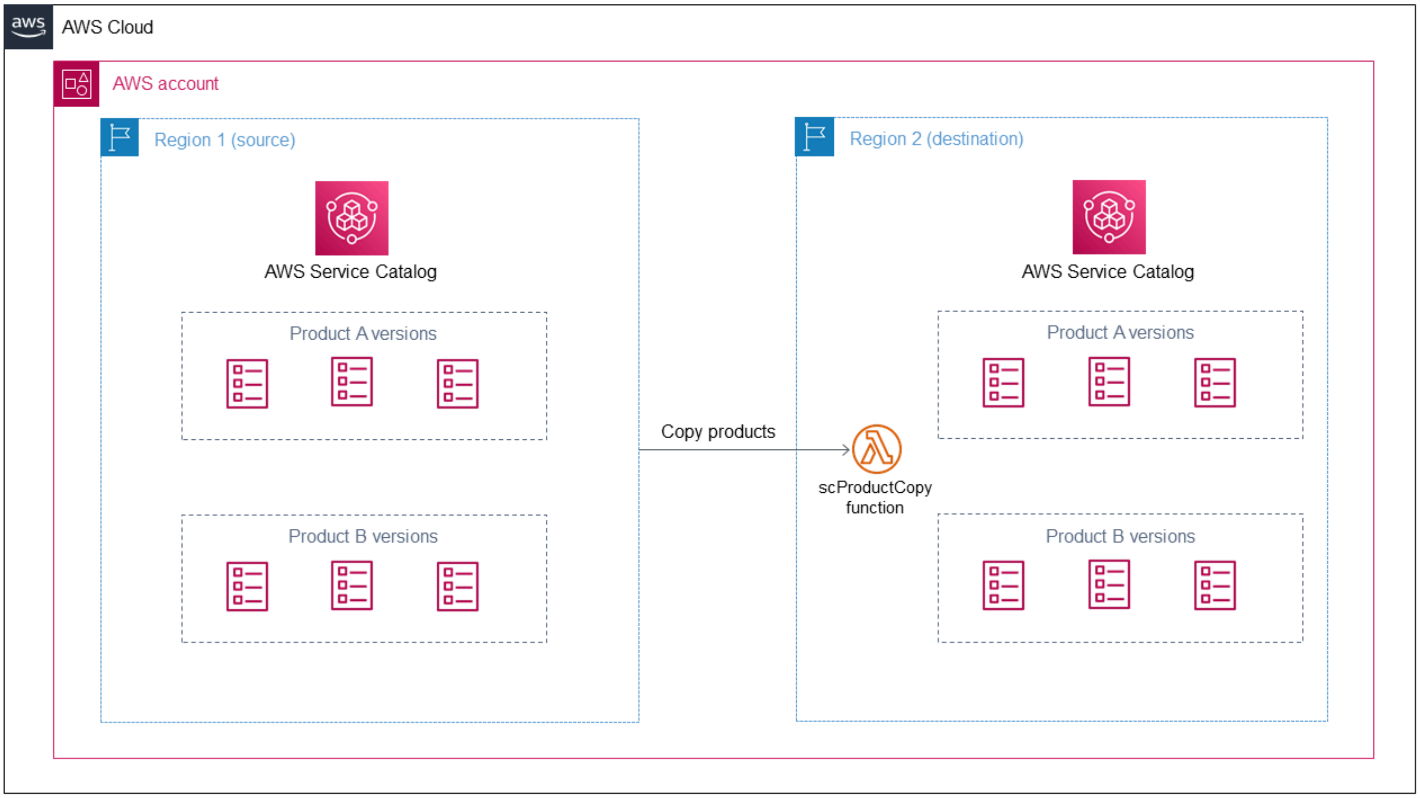
- AWS Service Catalog products that you want to copy across Regions or accounts cannot belong to more than one portfolio.

Architecture

The following diagram shows the copying of AWS Service Catalog products from a source account to a destination account.



The following diagram shows the copying of AWS Service Catalog products from a source Region to a destination Region.



Technology stack

- Amazon CloudWatch
- AWS Identity and Access Management (IAM)
- AWS Lambda
- AWS Service Catalog

Automation and scale

You can scale this pattern's approach by using a Lambda function that can be scaled depending on the number of requests received or how many AWS Service Catalog products you need to copy. For more information about this, see [Lambda function scaling](#) in the AWS Lambda documentation.

Tools

- [AWS Command Line Interface (AWS CLI)](#) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [AWS Service Catalog](#) helps you centrally manage catalogs of IT services that are approved for AWS. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.

Code

You can use the `cross-account-copy` package (attached) to copy AWS Service Catalog products across accounts or the `cross-region-copy` package (attached) to copy products across Regions.

The `cross-account-copy` package contains the following files:

- `copyconf.properties` – The configuration file that contains the Region and AWS account ID parameters for copying products across accounts.
- `scProductCopyLambda.py` – The Python function for copying products across accounts.
- `createDestAccountRole.sh` – The script to create an IAM role in the destination account.
- `createSrcAccountRole.sh` – The script to create an IAM role in the source account.
- `copyProduct.sh` – The script to create and invoke the Lambda function for copying products across accounts.

The `cross-region-copy` package contains the following files:

- `copyconf.properties` – The configuration file that contains the Region and AWS account ID parameters for copying products across Regions.
- `scProductCopyLambda.py` – The Python function for copying products across Regions.
- `copyProduct.sh` – The script to create an IAM role and create and invoke the Lambda function for copying products across Regions.

Epics

Option 1 – Copy AWS Service Catalog products across accounts

Task	Description	Skills required
Update the configuration file.	1. Download the `cross-account-copy` package (attached) to your local machine. 2. Update the `copyconf.properties` configuration file with the following values: • `srcRegion` – Provide the source Region that contains the products. • `destRegion` – Provide the destination Region for the products. • `sourceAccountId` – Provide the AWS account ID for your source account. • `destAccountId` – Provide the AWS account ID for your destination account.	AWS administrator, AWS systems administrator, Cloud administrator
Configure your credentials for AWS CLI in the destination account.	Configure your credentials to access AWS CLI in your destination account by running the `aws configure` command and providing the following values :	AWS administrator, AWS systems administrator, Cloud administrator

Task	Description	Skills required

	`$aws configure`
	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: <your_secret_access_key>`
	`Default region name [None]: Region`
	`Default output format [None]:`
	
	For more information about this, see Configuration basics in the AWS Command Line Interface documentation.	

Task	Description	Skills required

Configure your credentials for AWS CLI in the source account.	Configure your credentials to access AWS CLI in your source account by running the `aws configure` command and providing the following values:
	`$aws configure`
	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: <your_secret_access_key>`
	`Default region name [None]: Region`
	`Default output format [None]:`
	For more information about this, see [Configuration basics] in the AWS Command Line Interface documentation.
		AWS administrator, AWS systems administrator, Cloud administrator

Create a Lambda execution role in your destination account.	Run the `createDestAccountRole.sh` script in your destination account. The script implements the following actions: • Creates a Lambda execution role in your destination account • Creates and attaches the IAM policy for the Lambda execution role	AWS administrator, AWS systems administrator, Cloud administrator

Task	Description	Skills required
Create the cross-account IAM role in your source account.	Run the `createSrcAccountRole.sh` script in your source account. The script implements the following actions: • Creates a cross-account IAM role in your source account that is assumed by the Lambda execution role in the destination account to copy products • Creates and attaches an IAM policy for the cross-account role in your source account	AWS administrator, AWS systems administrator, Cloud administrator
Run the copyProduct script in the destination account.	Run the `copyProduct.sh` script in your destination account. The script implements the following actions: • Creates and invokes the Lambda function to copy products from the source account to the destination account	AWS administrator, AWS systems administrator, Cloud administrator

Option 2 – Copy AWS Service Catalog products from a source Region to a destination Region

Task	Description	Skills required
Update the configuration file.	1. Download the `cross-region-copy` package (attached) to your local machine. 2. Update the `copyconf.properties` configuration file with the following values: • `srcRegion` – Provide the source Region that contains the products. • `destRegion` – Provide the destination Region for the products. • `accountId` – Provide your AWS account ID.	AWS systems administrator, Cloud administrator, AWS administrator

Configure your credentials for AWS CLI.	Configure your credentials to access AWS CLI in your environment by running the `aws configure` command and providing the following values:
	
	`$aws configure`
	`AWS Access Key ID [None]: <your_access_key_id>`
	`AWS Secret Access Key [None]: <your_secret_access_key>`
	`Default region name [None]: Region`
		AWS administrator, AWS systems administrator, Cloud administrator

Task	Description	Skills required
	`Default output format [None]:` For more information about this, see Configuration basics in the AWS Command Line Interface documentation.	
Run the copyProduct script.	Run the `copyProduct.sh` script in your destination Region. The script implements the following actions: • Creates a Lambda execution role • Creates and attaches the IAM policy for the Lambda execution role • Creates and invokes the Lambda function to copy products from the source Region to the destination Region	AWS administrator, AWS systems administrator, Cloud administrator

Related resources

- Create a Lambda execution role (AWS Lambda documentation)
- Create a Lambda function (AWS Lambda documentation)
- AWS Service Catalog API reference
- AWS Service Catalog documentation

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Create a RACI or RASCI matrix for a cloud operating model

Teddy Germade, Jerome Descreux, Florian Leroux, and Josselin LE MINEUR, Amazon Web Services

Summary

The Cloud Center of Excellence (CCoE) or CEE (Cloud Enablement Engine) is an empowered and accountable team that is focused on operational readiness for the cloud. Their key focus is to transform the information IT organization from an on-premises operating model to a cloud operating model. The CCoE should be a cross-functional team that includes representation from infrastructure, applications, operations, and security.

One of the key components of a cloud operating model is a *RACI matrix* or *RASCI matrix*. This is used to define the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), support (S), consulted (C), and informed (I). The support type is optional. If you include it, it's called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

By starting with the attached template, your CCoE team can create a RACI or RASCI matrix for your organization. The template contains teams, roles, and tasks that are common in cloud operating models. The foundation of this matrix is the tasks related to operations integration and CCoE capabilities. However, you can customize this template to meet the needs of your organization's structure and use case.

There are no limits to the implementation of a RACI matrix. This approach works for large organizations, start-ups, and everything in between. For small organizations, the same resource can fill several roles.

Epics

Create the matrix

Task	Description	Skills required
Identify key stakeholders.	Identify key service and team managers that are linked to the strategic objectives of your cloud operating model.	Project manager

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Customize the matrix template.</td><td>Download the template in the Attachments section, and then update the RACI or RASCI matrix as follows:
On the Cloud Teams worksheet, update the CCoE stream names, team names, and team descripti ons as needed for your organization.
On the Cloud Roles worksheet, update the roles, team names, and role descriptions as needed for your organization.
On the RASCI worksheet , update the following as needed for your organizat ion:
In row 1 and column A, update the CCoE streams.
In row 2, update the team names.
In row 3, update the role names.
In columns D and E, update the general fields and activities that you want to include in your RASCI chart.</td><td>Project manager</td></tr>
</table>

Task	Description	Skills required
Plan meetings.	1. Communicate the RASCI objectives to all stakehold ers. 2. Plan one or more meetings so that an empowered representative from each team can attend.	Project manager

Task	Description	Skills required

Complete the matrix.	In the meeting with all stakeholders, do the following:
	
	1. Confirm that a represent ative from each team is present. Team participa tion is mandatory so that you can accurately assign responsibility types for each task.
	2. Review the what a RASCI matrix is and the objectives with the participants.
	3. Review the [shared responsibility model](https://aws.amazon.com/compliance/shared-responsibility-model/) with the participants so that they understand the scope of their organization’s responsibilities for security in the cloud.
	4. On the **RASCI** worksheet , for each task or activity, complete columns F through AN to assign the following responsibility types:
	    • **Responsible (R)** – This role is responsible for performing the work to complete the task.
	    • **Accountable (A)** – This role is held accountab le for making sure the
		Project manager

Task	Description	Skills required
	task is completed. This role is also responsible for ensuring the prerequisites are met and delegating the task to those who are responsible. • **Support (S)** – This role helps those who are responsible complete the task. This responsibility type is optional, and you can choose to exclude it in order to create a more traditional RACI matrix. • **Consulted (C)** – This role should be consulted for opinions or expertise on the task. Depending on the task, this responsibility type might not be required. • **Informed (I)** – This role should be kept up to date on the progress of the task and notified when the task is completed. • **Blank** – This role is not involved in the activity or task.	

Task	Description	Skills required
Share the RASCI matrix.	When the RACI or RASCI matrix is complete, have it approved by leadership. Save it in a shared repositor y or central location where all stakeholders can access it. We recommend that you use standard document control processes to record and approve revisions to the matrix.	Project manager

Related resources

- AWS shared responsibility model

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Create alarms for custom metrics using Amazon CloudWatch anomaly detection

Ram Kandaswamy and Raheem Jiwani, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, you can use Amazon CloudWatch to create alarms that monitor metrics and send notifications or automatically make changes if a threshold is breached.

To avoid being limited by static thresholds, you can create alarms based on past patterns and that notify you if specific metrics are outside the normal operating window. For example, you could monitor your API's response times from Amazon API Gateway and receive notifications about anomalies that prevent you from meeting a service-level agreement (SLA).

This pattern describes how to use CloudWatch anomaly detection for custom metrics. The pattern shows you how to create a custom metric in Amazon CloudWatch Logs Insights or publish a custom metric with an AWS Lambda function, and then set up anomaly detection and create notifications using Amazon Simple Notification Service (Amazon SNS).

Prerequisites and limitations

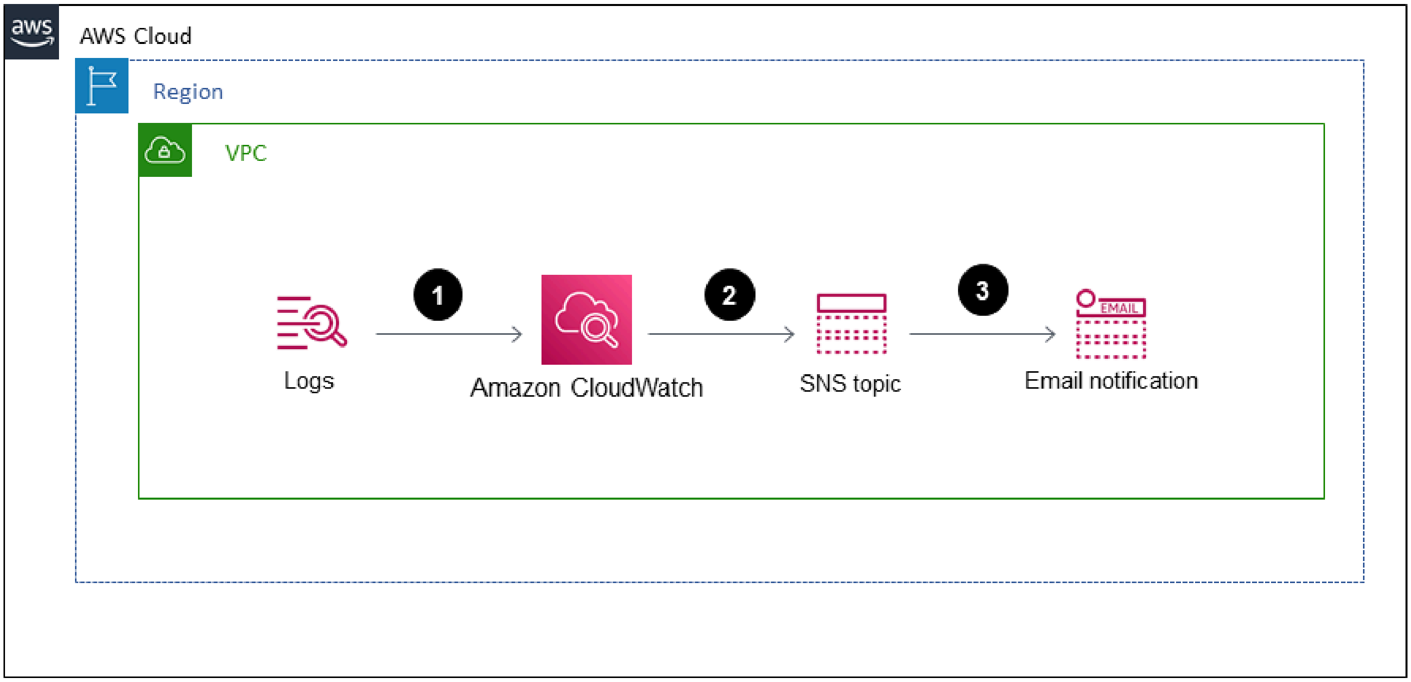
Prerequisites

- An active AWS account.
- An existing SNS topic, configured to send email notifications. For more information about this, see Getting started with Amazon SNS in the Amazon SNS documentation.
- An existing application, configured with CloudWatch Logs.

Limitations

- CloudWatch metrics don't support millisecond time intervals. For more information about the granularity of regular and custom metrics, see the Amazon CloudWatch FAQs.

Architecture



The diagram shows the following workflow:

1. Logs that use metrics created and updated by CloudWatch Logs are streamed to CloudWatch.
2. An alarm initiates based on thresholds and sends an alert to an SNS topic.
3. Amazon SNS sends you an email notification.

Technology stack

- CloudWatch
- AWS Lambda
- Amazon SNS

Tools

- Amazon CloudWatch provides a reliable, scalable, and flexible monitoring solution.
- AWS Lambda is a compute service that helps you run code without provisioning or managing servers.
- Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers.

Epics

Set up anomaly detection for a custom metric

Task	Description	Skills required
Option 1 - Create a custom metric with a Lambda function.	Download the `lambda_function.py` file (attached) and then replace the sample `lambda_function.py` file in the [aws-lambda-developer-guide](https://github.com/awsdocs/aws-lambda-developer-guide) repository on the AWS Documentation GitHub. This provides you with a sample Lambda function that sends custom metrics to CloudWatch Logs. The Lambda function uses the Boto3 API to integrate with CloudWatch. After you run the Lambda function, you can sign in to the AWS Management Console, open the CloudWatch console, and the published metric is available under your published namespace.	DevOps engineer, AWS DevOps
Option 2 – Create custom metrics from CloudWatch log groups.	Sign in to the AWS Management Console, open the CloudWatch console, and then choose **Log groups**. Choose the log group that you want to create a metric for.	DevOps engineer, AWS DevOps

Task	Description	Skills required
	Choose **Actions** and then choose **Create metric filter**. For **Filter pattern**, enter the filter pattern that you want to use. For more informati on, see [Filter and pattern syntax](#) in the CloudWatch documentation. To test your filter pattern, enter one or more log events under **Test Pattern**. Each log event must be within one line, because line breaks are used to separate log events in the **Log event** messages box. After you test the pattern, you can enter a name and value for your metric under **Metric details**. For more information and steps to create a custom metric, see [Create a metric filter for a log group](#) in the CloudWatch documentation.	

Task	Description	Skills required
Create an alarm for your custom metric.	On the CloudWatch console, choose **Alarms** and then choose **Create Alarm**. Choose **Select metric** and enter the name of the metric that you created earlier into the search box. Choose the **Graphed metrics** tab and configure the options according to your requirements. Under **Conditions**, choose **Anomaly detection** instead of **Static thresholds**. This shows you a band based on two standard default deviations. You can set up thresholds and adjust them according to your requireme nts. Choose **Next.** **Note** The band is dynamic and depends on the quality of the datapoints. When you begin aggregati ng more data, the band and threshold s are automatically updated.	DevOps engineer, AWS DevOps

Task	Description	Skills required
Set up SNS notifications.	Under **Notification**, choose the SNS topic to notify when the alarm is in `ALARM` state, `OK` state, or `INSUFFICIENT_DATA` state. To have the alarm send multiple notifications for the same alarm state or for different alarm states, choose **Add notification**. Choose **Next**. Enter a name and description for the alarm. The name must only contain ASCII characters. Then choose **Next**. Under **Preview and create**, confirm that the information and conditions are correct, and then choose **Create alarm**.	DevOps engineer, AWS DevOps

Related resources

- Publishing custom metrics to CloudWatch
- Using CloudWatch anomaly detection
- Alarm events and Amazon EventBridge
- What are the best practices to follow while pushing custom metrics to Cloud Watch? (video)
- Introduction to CloudWatch Application Insights (video)
- Detect anomalies with CloudWatch (video)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Create an AWS Cloud9 IDE that uses Amazon EBS volumes with default encryption

Janardhan Malyala and Dhrubajyoti Mukherjee, Amazon Web Services

Summary

Notice: AWS Cloud9 is no longer available to new customers. Existing customers of AWS Cloud9 can continue to use the service as normal. Learn more

You can use encryption by default to enforce the encryption of your Amazon Elastic Block Store (Amazon EBS) volumes and snapshot copies on the Amazon Web Services (AWS) Cloud.

You can create an AWS Cloud9 integrated development environment (IDE) that uses EBS volumes encrypted by default. However, the AWS Identity and Access Management (IAM) service-linked role for AWS Cloud9 requires access to the AWS Key Management Service (AWS KMS) key for these EBS volumes. If access is not provided, the AWS Cloud9 IDE might fail to launch and debugging might be difficult.

This pattern provides the steps to add the service-linked role for AWS Cloud9 to the AWS KMS key that is used by your EBS volumes. The setup described by this pattern helps you successfully create and launch an IDE that uses EBS volumes with encryption by default.

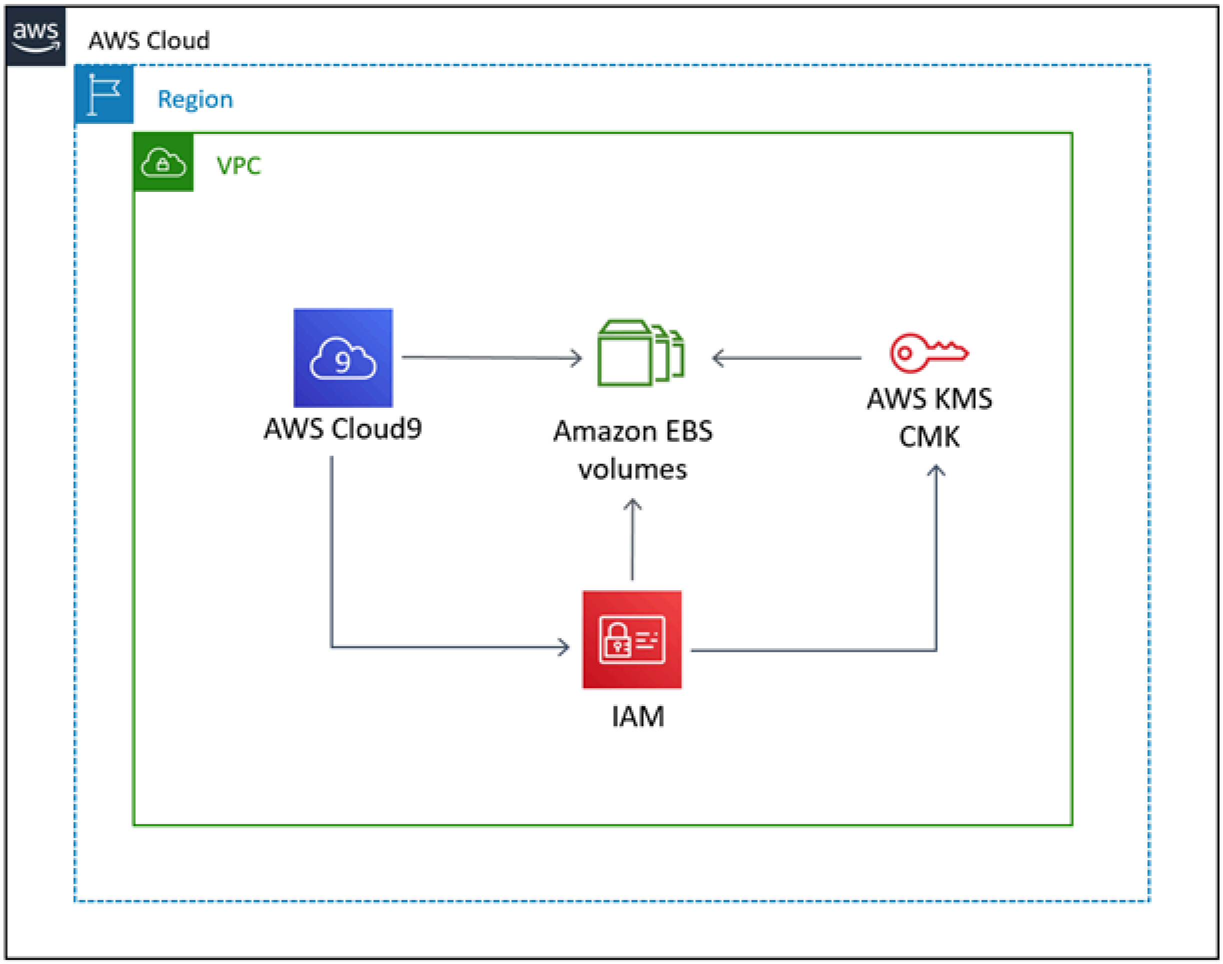
Prerequisites and limitations

Prerequisites

- An active AWS account.
- Default encryption turned on for EBS volumes. For more information about encryption by default, see Amazon EBS encryption in the Amazon Elastic Compute Cloud (Amazon EC2) documentation.
- An existing customer managed KMS key for encrypting your EBS volumes.

> **Note**
>
> You don't need to create the service-linked role for AWS Cloud9. When you create an AWS Cloud9 development environment, AWS Cloud9 creates the service-linked role for you.

Architecture



Technology stack

- AWS Cloud9
- IAM
- AWS KMS

Tools

- [AWS Cloud9](https://docs.aws.amazon.com/cloud9/latest/user-guide/welcome.html) is an integrated development environment (IDE) that helps you code, build, run, test, and debug software. It also helps you release software to the AWS Cloud.
- [Amazon Elastic Block Store (Amazon EBS)](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html) provides block-level storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances.

- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Key Management Service (AWS KMS)](#) helps you create and control cryptographic keys to help protect your data.

Epics

Find the default encryption key value

Task	Description	Skills required
Record the default encryptio n key value for the EBS volumes.	Sign in to the AWS Management Console and open the Amazon EC2 console. Choose **EC2 dashboard**, and then choose **Data protection and security** in **Account attributes**. In **EBS encryption** section, copy and record the value in **Default encryption key**.	Cloud architect, DevOps engineer

Provide access to the AWS KMS key

Task	Description	Skills required
Provide AWS Cloud9 with access to the KMS key for EBS volumes.	1. Open the AWS KMS console, and then choose **Customer managed keys**. Select the AWS KMS key used for Amazon EBS encryption, and then choose **View key**. 2. On the **Key policy** tab, confirm that you can see the text form of the key	Cloud architect, DevOps engineer

Task	Description	Skills required
	policy. If you cannot see the text form, choose **Switch to policy view**. 3. Choose **Edit**. Add the code in the [Additional information](#) section to the policy, and then choose **Save changes**. The policy changes permit the service-linked role for AWS Cloud9, `AWSServiceRoleForAWSCloud9` , to access the key. For more information about updating a key policy, see [How to change a key policy](#) (AWS KMS documentation). **Important** The service-linked role for AWS Cloud9 is automatically created when you launch your first IDE. For more information, see [Creating a service-linked role](#) in the AWS Cloud9 documentation.	

Create and launch the IDE

Task	Description	Skills required
Create and launch the AWS Cloud9 IDE.	Open the AWS Cloud9 console and choose **Create environment**. Configure IDE according to your requirements by following the steps from [Creating an EC2 environment](#) in the AWS Cloud9 documentation.	Cloud architect, DevOps engineer

Related resources

- [Encrypt EBS volumes used by AWS Cloud9](#)
- [Create a service-linked role for AWS Cloud9](#)
- [Create an EC2 environment in AWS Cloud9](#)

Additional information

AWS KMS key policy updates

Replace <aws_accountid> with your AWS account ID.

```
{
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<aws_accountid>:role/aws-service-role/
cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<aws_accountid>:role/aws-service-role/
cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
```

Using a cross-account key

If you want to use a cross-account KMS key, you must use a grant in combination with the KMS key policy. This enables cross-account access to the key. In the same account that you used to create the Cloud9 environment, run the following command in the terminal.

```
aws kms create-grant \
 --region <Region where Cloud9 environment is created> \
 --key-id <The cross-account KMS key ARN> \
 --grantee-principal arn:aws:iam::<The account where Cloud9 environment is
 created>:role/aws-service-role/cloud9.amazonaws.com/AWSServiceRoleForAWSCloud9 \
 --operations "Encrypt" "Decrypt" "ReEncryptFrom" "ReEncryptTo" "GenerateDataKey"
 "GenerateDataKeyWithoutPlaintext" "DescribeKey" "CreateGrant"
```

After you run this command, you can create Cloud9 environments by using EBS encryption with a key in a different account.

Create tag-based Amazon CloudWatch dashboards automatically

Janak Vadaria, Vinodkumar Mandalapu, and RAJNEESH TYAGI, Amazon Web Services

Summary

Creating different Amazon CloudWatch dashboards manually can be time-consuming, particularly when you have to create and update multiple resources to automatically scale your environment. A solution that creates and updates your CloudWatch dashboards automatically can save you time. This pattern helps you deploy a fully automated AWS Cloud Development Kit (AWS CDK) pipeline that creates and updates CloudWatch dashboards for your AWS resources based on tag change events, to display Golden Signals metrics.

In site reliability engineering (SRE), Golden Signals refers to a comprehensive set of metrics that offer a broad view of a service from a user or consumer perspective. These metrics consist of latency, traffic, errors, and saturation. For more information, see [What is Site Reliability Engineering (SRE)?](#) on the AWS website.

The solution provided by this pattern is event-driven. After it's deployed, it continuously monitors the tag change events and automatically updates the CloudWatch dashboards and alarms.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Command Line Interface (AWS CLI), installed and configured
- Prerequisites for the AWS CDK v2
- A bootstrapped environment on AWS
- Python version 3
- AWS SDK for Python (Boto3), installed
- Node.js version 18 or later
- Node package manager (npm), installed and configured for the AWS CDK
- Moderate (level 200) familiarity with the AWS CDK and AWS CodePipeline

Limitations

This solution currently creates automated dashboards for the following AWS services only:

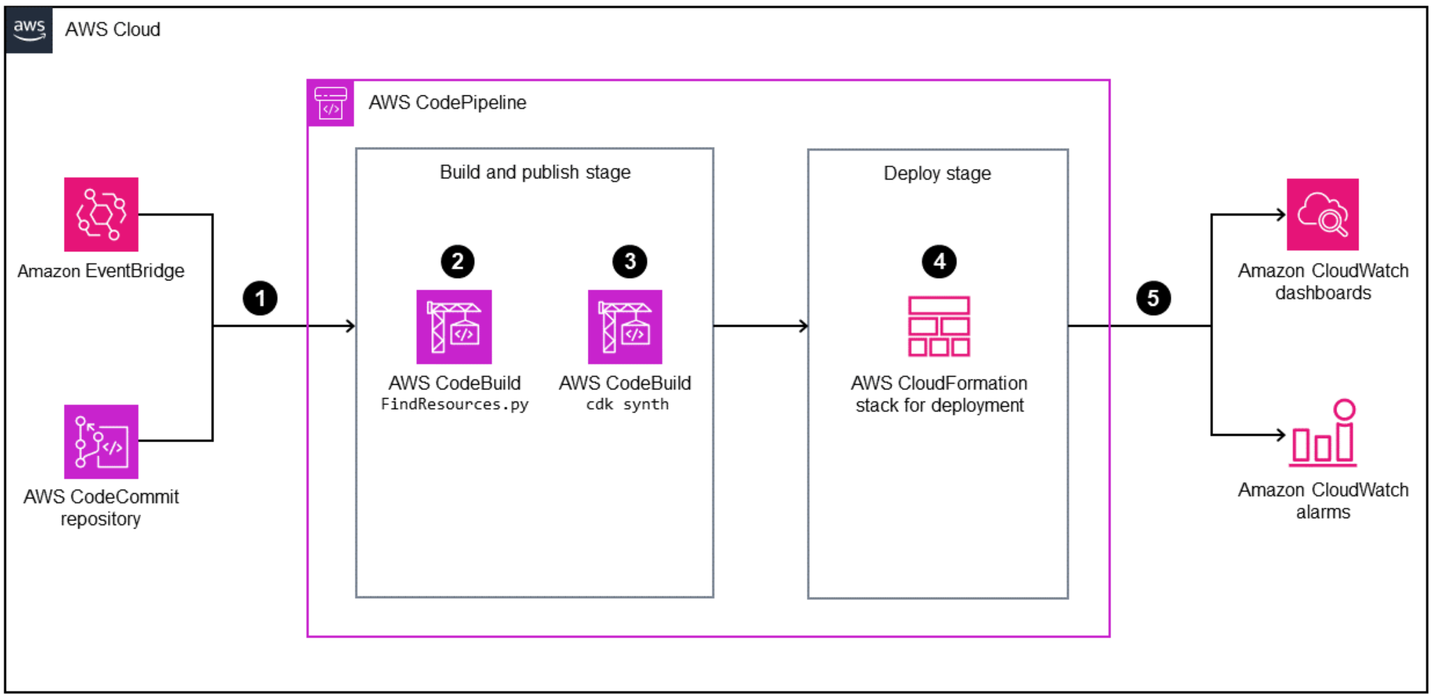
- Amazon Relational Database Service (Amazon RDS)
- AWS Auto Scaling
- Amazon Simple Notification Service (Amazon SNS)
- Amazon DynamoDB
- AWS Lambda

Architecture

Target technology stack

- CloudWatch dashboards
- CloudWatch alarms

Target architecture



1. An AWS tag change event for the configured application tags or code changes initiates a pipeline in AWS CodePipeline to build and deploy updated CloudWatch dashboards.
2. AWS CodeBuild runs a Python script to find the resources that have configured tags and stores the resource IDs in a local file in a CodeBuild environment.

3. CodeBuild runs **cdk synth** to generate AWS CloudFormation templates that deploy CloudWatch dashboards and alarms.
4. CodePipeline deploys the AWS CloudFormation templates to the specified AWS account and Region.
5. When the AWS CloudFormation stack has been deployed successfully, you can view the CloudWatch dashboards and alarms.

Automation and scale

This solution has been automated by using the AWS CDK. You can find the code in the GitHub Golden Signals Dashboards on Amazon CloudWatch repository. For additional scaling and to create custom dashboards, you can configure multiple tag keys and values.

Tools

Amazon services

- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources, including AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS Command Line Interface (AWS CLI) is an open source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Best practices

As a security best practice, you can use encryption and authentication for the source repositories that connect to your pipelines. For additional best practices, see CodePipeline best practices and use cases in the CodePipeline documentation.

Epics

Configure and deploy the sample application

Task	Description	Skills required
Configure and deploy the sample application.	1. Clone the GitHub [sample code repository](#) by using the command: `git clone https://github.com/aws-samples/golden-signals-dashboards-sample-app` 2. Navigate to the cloned repository on your computer and open the `src/project-settings.ts` file with the editor of your choice. 3. Change the `projectSettings` constant value according to your AWS resource tags and application mappings. 4. Set the `AWS_ACCOUNT`, `AWS_REGION`, and `GS_DASHBOARD_INSTANCE` environment variables: • Set `AWS_ACCOUNT` to the account ID of your AWS account. • Set `AWS_REGION` to the Region where you want	AWS DevOps

Task	Description	Skills required
	to deploy the sample application. • Set `GS_DASHBOARD_INSTANCE` to `dev`, `test`, or `prod`, depending on your development environment. (We recommend `test` for the testing procedure described in this pattern.) 5. Set up the AWS CLI with your AWS credentials. For more information, see [Set and view configuration settings using commands](#) in the AWS CLI documentation. 6. Run the following command to deploy the Golden Signals dashboard sample application: `sh deploy.sh`	

Task	Description	Skills required
Automatically create dashboards and alarms.	After you deploy the sample application, you can create any of the resources that this solution supports with expected tag values, which will automatically create the specified dashboards and alarms. To test this solution, create an AWS Lambda function: 1. Sign in to the AWS Management Console in the AWS Region where you deployed the sample application. 2. Open the Lambda console at https://console.aws.amazon.com/lambda/. 3. Choose **Create a function**, and then enter a function name. 4. In the **Advanced settings** pane, select **Enable tags**, and then choose **Add new tag**. Enter the following key and value: • Key: `AutoDashboard` • Value: `True` 5. Choose **Create function**. The Lambda function immediately starts a code	AWS DevOps

Task	Description	Skills required
	pipeline, which creates the dashboards and alarms for that particular Lambda function automatically. 6. To view the automated dashboards and alarms, open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/. You can view the custom dashboards and alarms for the function you specified in the `projectSettings` constant (**APP1-lambda** by default). 7. Select the dashboard for the Lambda function to view additional automated dashboards that were created as part of this solution. 8. Repeat these steps for other services, such as Amazon RDS, Amazon SNS, AWS Auto Scaling, and DynamoDB, to generate the associated dashboards. For an example for Amazon RDS, see the [Additional information](#) section.	

Remove the sample application

Task	Description	Skills required
Remove the `golden-signals-dashboard` construct.	1. To remove all the AWS CloudFormation stacks created by the sample application, you have to reconfigure the `AWS_ACCOUNT` , `AWS_REGION` , and `GS_DASHBOARD_INSTANCE` environment variables. The `destroy.sh` command requires these configurations. • `AWS_ACCOUNT` is the account ID of your AWS account. • `AWS_REGION` is the Region where you deployed your sample application. • `GS_DASHBOARD_INSTANCE` is `dev`, `test`, or `prod`, based on your previous settings. 2. Set up AWS CLI with your AWS credentials. 3. Run the following command to remove the sample application and all associated AWS CloudFormation stacks:	AWS DevOps

Task	Description	Skills required
	`sh destroy.sh`	

Troubleshooting

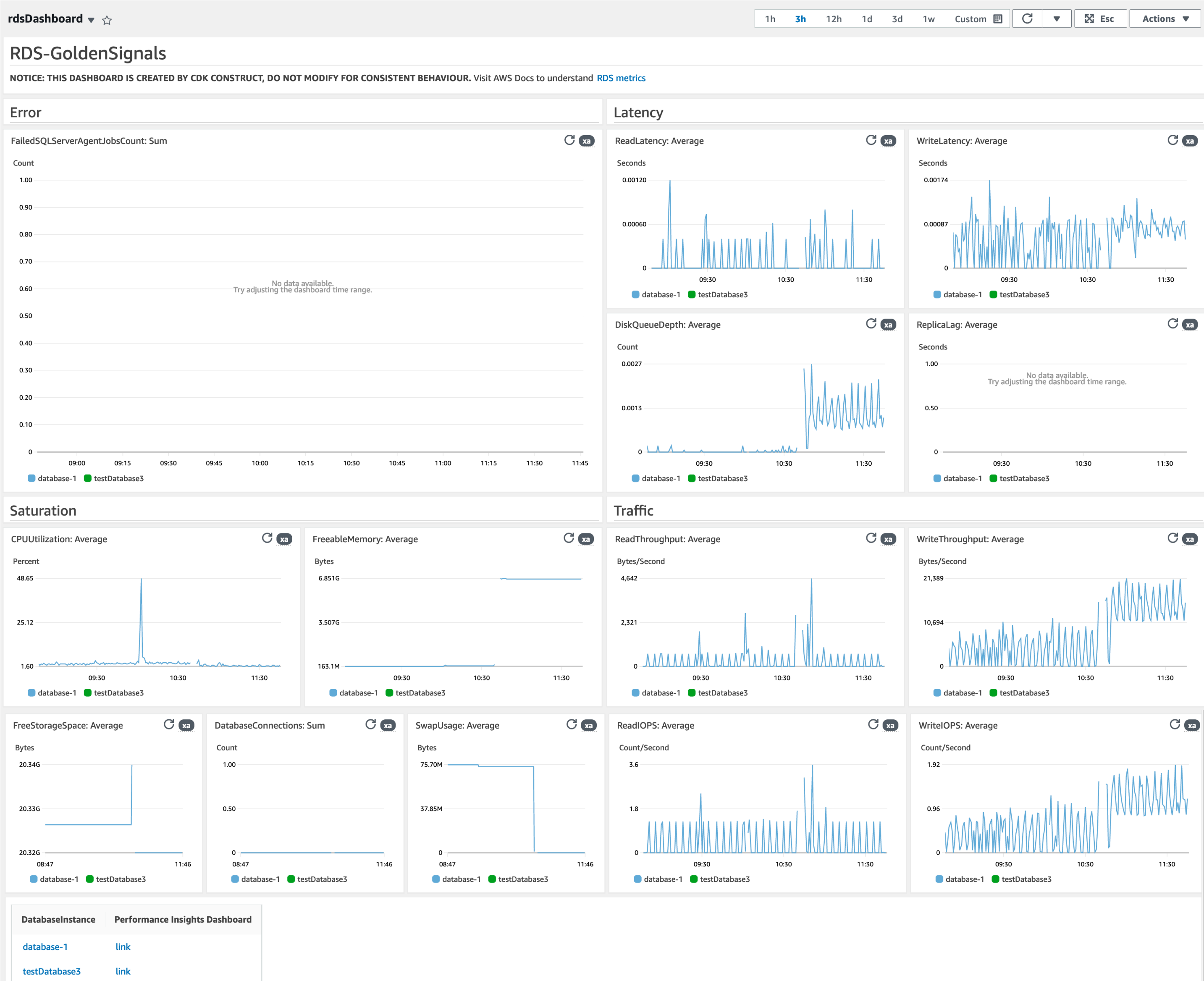
Issue	Solution
Python command not found (referring to `findresources.sh` , line 8).	Check the version of your Python installat ion. If you have installed Python version 3, replace `python` with `python3` on line 8 of the `resources.sh` file, and run the `sh deploy.sh` command again to deploy the solution.

Related resources

- Bootstrapping (AWS CDK documentation)
- Using named profiles (AWS CLI documentation)
- AWS CDK Workshop

Additional information

The following illustration shows a sample dashboard for Amazon RDS that is created as part of this solution.


rdsDashboard
1h 3h 12h 1d 3d 1w Custom
Esc
Actions
RDS-GoldenSignals
NOTICE: THIS DASHBOARD IS CREATED BY CDK CONSTRUCT, DO NOT MODIFY FOR CONSISTENT BEHAVIOUR. Visit AWS Docs to understand RDS metrics
Error
FailedSQLServerAgentJobsCount: Sum
Count
No data available.
Try adjusting the dashboard time range.
database-1 testDatabase3
Latency
ReadLatency: Average
Seconds
WriteLatency: Average
Seconds
DiskQueueDepth: Average
Count
ReplicaLag: Average
Seconds
No data available.
Try adjusting the dashboard time range.
Saturation
CPUUtilization: Average
Percent
FreeableMemory: Average
Bytes
Traffic
ReadThroughput: Average
Bytes/Second
WriteThroughput: Average
Bytes/Second
FreeStorageSpace: Average
Bytes
DatabaseConnections: Sum
Count
SwapUsage: Average
Bytes
ReadIOPS: Average
Count/Second
WriteIOPS: Average
Count/Second
DatabaseInstance
Performance Insights Dashboard
database-1
link
testDatabase3
link

Document your AWS landing zone design

Michael Daehnert, Florian Langer, and Michael Lodemann, Amazon Web Services

Summary

A *landing zone* is a well-architected, multi-account environment that's based on security and compliance best practices. It is the enterprise-wide container that holds all of your organizational units (OUs), AWS accounts, users, and other resources. A landing zone can scale to fit the needs of an enterprise of any size. AWS has two options for creating your landing zone: a service-based landing zone using AWS Control Tower or a customized landing zone that you build. Each option requires a different level of AWS knowledge.

AWS created AWS Control Tower to help you save time by automating the setup of a landing zone. AWS Control Tower is managed by AWS and uses best practices and guidelines to help you create your foundational environment. AWS Control Tower uses integrated services, such as AWS Service Catalog and AWS Organizations, to provision accounts in your landing zone and manage access to those accounts.

AWS landing zone projects vary in requirements, implementation details, and operational action items. There are customization aspects that need to be handled with every landing zone implementation. This includes (but is not limited to) how access management is handled, which technology stack is used, and what the monitoring requirements are for operational excellence. This pattern provides a template that helps you document your landing zone project. By using the template, you can document your project more quickly and help your development and operations teams understand your landing zone.

Prerequisites and limitations

Limitations

This pattern does not describe what a landing zone is or how to implement one. For more information about these topics, see the Related resources section.

Epics

Create the design document

Task	Description	Skills required
Identify key stakeholders.	Identify key service and team managers that are linked to your landing zone.	Project manager
Customize the template.	Download the template in the [Attachments](#) section, and then update the template as follows: 1. Remove any sections that don't apply to your organization's landing zone or processes. 2. Add any sections that are unique to your organizat ion.	Project manager
Complete the template.	In meetings with the stakeholders or by using a write-and-review process, complete the template as follows: 1. Use the guidance and information in the blue boxes to complete each section. 2. Replace or remove any yellow fields with custom values for your organizat ion.	Project manager

Task	Description	Skills required
	3. Replace or remove any image fields with your custom architecture or flow diagrams. 4. Complete the *Revision history* and *Contributors* section of the template.	
Share the design document.	When your landing zone design documentation is complete, save it in a shared repository or central location where all stakeholders can access it. We recommend that you use standard document control processes to record and approve revisions to the design document.	Project manager

Related resources

- AWS Control Tower documentation
  - Plan your AWS Control Tower landing zone
  - AWS multi-account strategy for your AWS Control Tower landing zone
  - Administrative tips for landing zone setup
  - Expectations for landing zone configuration
- Customizations for AWS Control Tower (AWS Solutions Library)
- Setting up a secure and scalable multi-account AWS environment (AWS Prescriptive Guidance)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK

Dr. Rahul Sharad Gaikwad, Amazon Web Services

Summary

This pattern demonstrates the steps to enable the Amazon DevOps Guru service across multiple Amazon Web Services (AWS) Regions, accounts, and organizational units (OUs) by using the AWS Cloud Development Kit (AWS CDK) in TypeScript. You can use AWS CDK stacks to deploy AWS CloudFormation StackSets from the administrator (primary) AWS account to enable Amazon DevOps Guru across multiple accounts, instead of logging into each account and enabling DevOps Guru individually for each account.

Amazon DevOps Guru provides artificial intelligence operations (AIOps) features to help you improve the availability of your applications and resolve operational issues faster. DevOps Guru reduces your manual effort by applying machine learning (ML) powered recommendations, without requiring any ML expertise. DevOps Guru analyzes your resources and operational data. If it detects any anomalies, it provides metrics, events, and recommendations to help you address the issue.

This pattern describes three deployment options for enabling Amazon DevOps Guru:

- For all stack resources across multiple accounts and Regions
- For all stack resources across OUs
- For specific stack resources across multiple accounts and Regions

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI), installed and configured. (See Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.)
- AWS CDK Toolkit, installed and configured. (See AWS CDK Toolkit in the AWS CDK documentation.)
- Node Package Manager (npm), installed and configured for the AWS CDK in TypeScript. (See Downloading and installing Node.js and npm in the npm documentation.)

- Python3 installed and configured, for running a Python script to inject traffic into the sample serverless application. (See Python Setup and Usage in the Python documentation.)
- Pip, installed and configured to install the Python requests library. (See the pip installation instructions on the PyPl website.)

Product versions

- AWS CDK Toolkit version 1.107.0 or later
- npm version 7.9.0 or later
- Node.js version 15.3.0 or later

Architecture

Technologies

The architecture for this pattern includes the following services:

- Amazon DevOps Guru
- AWS CloudFormation
- Amazon API Gateway
- AWS Lambda
- Amazon DynamoDB
- Amazon CloudWatch
- AWS CloudTrail

AWS CDK stacks

The pattern uses the following AWS CDK stacks:

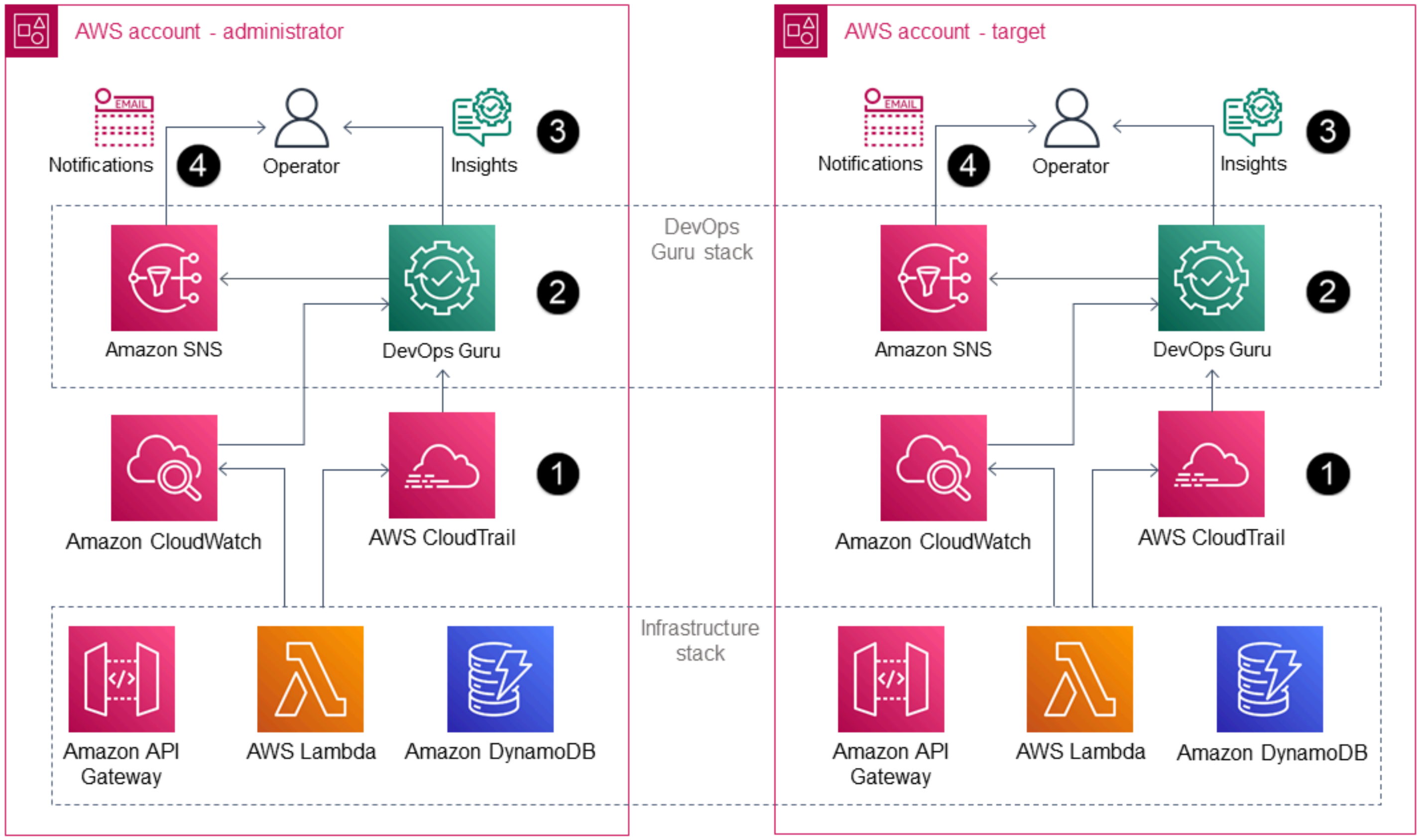
- `CdkStackSetAdminRole` – Creates an AWS Identity and Access management (IAM) administrator role to establish a trust relationship between the administrator and target accounts.
- `CdkStackSetExecRole` – Creates an IAM role to trust the administrator account.

- `CdkDevopsGuruStackMultiAccReg` – Enables DevOps Guru across multiple AWS Regions and accounts for all stacks, and sets up Amazon Simple Notification Service (Amazon SNS) notifications.
- `CdkDevopsGuruStackMultiAccRegSpecStacks` – Enables DevOps Guru across multiple AWS Regions and accounts for specific stacks, and sets up Amazon SNS notifications.
- `CdkDevopsguruStackOrgUnit` – Enables DevOps Guru across OUs, and sets up Amazon SNS notifications.
- `CdkInfrastructureStack` – Deploys sample serverless application components such as API Gateway, Lambda, and DynamoDB in the administrator account to demonstrate fault injection and insights generation.

Sample application architecture

The following diagram illustrates the architecture of a sample serverless application that has been deployed across multiple accounts and Regions. The pattern uses the administrator account to deploy all the AWS CDK stacks. It also uses the administrator account as one of the target accounts for setting up DevOps Guru.

1. When DevOps Guru is enabled, it first baselines each resource's behavior and then ingests operational data from CloudWatch vended metrics.
2. If it detects an anomaly, it correlates it with the events from CloudTrail, and generates an insight.
3. The insight provides a correlated sequence of events along with prescribed recommendations to enable the operator to identify the culprit resource.
4. Amazon SNS sends notification messages to the operator.



Automation and scale

The [GitHub repository](#) provided with this pattern uses the AWS CDK as an infrastructure as code (IaC) tool to create the configuration for this architecture. AWS CDK helps you orchestrate resources and enable DevOps Guru across multiple AWS accounts, Regions, and OUs.

Tools

AWS services

- [AWS CDK](#) – AWS Cloud Development Kit (AWS CDK) helps you define your cloud infrastructure as code in one of five supported programming languages: TypeScript, JavaScript, Python, Java, and C#.
- [AWS CLI](#) – AWS Command Line Interface (AWS CLI) is a unified tool that provides a consistent command-line interface for interacting with AWS services and resources.

Code

The source code for this pattern is available on GitHub, in the [Amazon DevOps Guru CDK Samples] repository. The AWS CDK code is written in TypeScript. To clone and use the repository, follow the instructions in the next section.

> **Important**
>
> Some of the stories in this pattern include AWS CDK and AWS CLI command examples that are formatted for Unix, Linux, and macOS. For Windows, replace the backslash (\) continuation character at the end of each line with a caret (^).

Epics

Prepare the AWS resources for deployment

Task	Description	Skills required

Configure AWS named profiles.	Set up your AWS named profiles as follows to deploy stacks in a multi-account environment.
	
	For the administrator account:
	
	`$aws configure --profile administrator`
	`AWS Access Key ID [****]: <your-administrator-access-key-ID>`
	`AWS Secret Access Key [****]: <your-administrator-secret-access-key>`
	`Default region name [None]: <your-administrator-region>`
	`Default output format [None]: json`
		DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>For the target account:
<pre>$aws configure --profile
 target
AWS Access Key ID
 [****: <your-target-
access-key-ID>
AWS Secret Access Key
 [****]: <your-target-
secret-access-key>
Default region name
 [None]: <your-target-
region>
Default output format
 [None]: json</pre>For more information, see Using named profiles in the AWS CLI documentation.</td><td></td></tr>
<tr><td>Verify AWS profile configurations.</td><td>(Optional) You can verify your AWS profile configurations in the <code>credentials</code> and <code>config</code> files by following the instructions in Set and view configuration settings in the AWS CLI documentation.</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
Verify the AWS CDK version.	Verify the version of the AWS CDK Toolkit by running the following command: `$cdk --version` This pattern requires version 1.107.0 or later. If you have an earlier version of the AWS CDK, follow the instructions in the [AWS CDK documentation](https://docs.aws.amazon.com/cdk/) to update it.	DevOps engineer
Clone the project code.	Clone the GitHub repository for this pattern by using the command: `$git clone https://github.com/aws-samples/amazon-devopsguru-cdk-samples.git`	DevOps engineer

Task	Description	Skills required

Install package dependencies and compile the TypeScript files.	Install the package dependencies and compile the TypeScript files by running the following commands:
	`$cd amazon-devopsguru-cdk-samples`
	`$npm install`
	`$npm fund`
	These commands install all the packages from the sample repository.
	Important
	If you get any errors about missing packages, use one of the following commands:
	`$npm ci`
	—or—
	`$npm install -g @aws-cdk/<package-name>`
	You can find the list of package names and versions in the `Dependencies` section of the `/amazon-d`
		DevOps engineer

Task	Description	Skills required
	`evopsguru-cdk-samples/package.json` file. For more information, see [npm ci] and [npm install] in the npm documentation.	

Build (synthesize) the AWS CDK stacks

Task	Description	Skills required
Configure an email address for Amazon SNS notifications.	Follow these steps to provide an email address for Amazon SNS notifications: 1. Edit the files `/amazon-devopsguru-cdk-samples/lib/cdk-devopsguru-multi-acc-reg-stack.ts` and `/amazon-devopsguru-cdk-samples/lib/cdk-devopsguru-org-uni-stack.ts` . 2. In the `DevOpsGuruTopic` , `Subscription` section, update the `Endpoint` parameter with your email address. 3. Save and close the files.	DevOps engineer
Build the project code.	Build the project code and synthesize the stacks by running the command:	DevOps engineer

Task	Description	Skills required
	```npm run build && cdk synth``` You should see output similar to the following: ```$npm run build && cdk synth > cdk-devopsguru@0.1.0 build > tsc Successfully synthesized to ~/amazon-devopsguru-cdk-samples/cdk.out Supply a stack id (CdkDevopsGuruStackMultiAccReg,CdkDevopsGuruStackMultiAccRegSpecStacks, CdkDevopsguruStackOrgUnit, CdkInfrastructureStack, CdkStackSetAdminRole, CdkStackSetExecRole) to display its template.``` For more information and steps, see [Your first AWS CDK app](#) in the AWS CDK documentation.	

Task	Description	Skills required

List the AWS CDK stacks.	Run the following command to list all AWS CDK stacks:
	`$cdk list`
	The command displays the following list:
	`CdkDevopsGuruStackMultiAccReg`
	`CdkDevopsGuruStackMultiAccRegSpecStacks`
	`CdkDevopsguruStackOrgUnit`
	`CdkInfrastructureStack`
	`CdkStackSetAdminRole`
	`CdkStackSetExecRole`
		DevOps engineer

Option 1 - Enable DevOps Guru for all stack resources across multiple accounts

Task	Description	Skills required
Deploy the AWS CDK stacks for creating IAM roles.	This pattern uses [AWS CloudFormation StackSets](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html) to perform stack operations across multiple accounts. If you are creating your first stack set, you must create the following IAM roles to get the required permissions set up in your AWS accounts:	DevOps engineer

Task	Description	Skills required
	• `AWSCloudFormationStackSetAdministrationRole` • `AWSCloudFormationStackSetExecutionRole` **Note** The roles must have these exact names. 1. Create the IAM `AWSCloudFormationStackSetAdministrationRole` role in the administrator (primary) account by running the following CLI command: `$cdk deploy CdkStackSetAdminRole --profile administrator` 2. Create the IAM `AWSCloudFormationStackSetExecutionRole` role in all target accounts where you want to run the stack instances. To create this role, run these CLI commands:	

Task	Description	Skills required

	<pre>$cdk deploy CdkStackSetExecRole \
	  --parameters AdministratorAccountId=<administrator-account-ID> \
	  --profile administrator
	
	$cdk deploy CdkStackSetExecRole \
	  --parameters AdministratorAccountId=<administrator-account-ID> \
	  --profile target</pre>
	For more information, see Grant self-managed permissions in the AWS CloudFormation documentation.	

Task	Description	Skills required
Deploy the AWS CDK stack for enabling DevOps Guru across multiple accounts.	The AWS CDK `CdkDevopsGuruStackMultiAccReg` stack creates stack sets to deploy stack instances across multiple accounts and Regions. To deploy the stack, run the following CLI command with the specified parameters: `$cdk deploy CdkDevopsGuruStackMultiAccReg \ --profile administrator \ --parameters AdministratorAccountId=<administrator-account-ID> \ --parameters TargetAccountId=<target-account-ID> \ --parameters RegionIds="<region-1>,<region-2>"` Currently Amazon DevOps Guru is available in the AWS Regions listed in the [DevOps Guru FAQ]().	DevOps engineer

Option 2 - Enable DevOps Guru for all stack resources across OUs

Task	Description	Skills required
Extract OU IDs.	On the AWS Organizations console, identify the IDs of the organizational units where you want to enable DevOps Guru.	DevOps engineer
Enable service-managed permissions for OUs.	If you're using AWS Organizations for account management, you must grant service-managed permissions to enable DevOps Guru. Instead of creating the IAM roles manually, use organization-based trusted access and service-linked roles (SLRs).	DevOps engineer
Deploy the AWS CDK stack for enabling DevOps Guru across OUs.	The AWS CDK `CdkDevopsguruStackOrgUnit` stack enables DevOps Guru service across OUs. To deploy the stack, run the following command with the specified parameters: `$cdk deploy CdkDevopsguruStackOrgUnit \ --profile administrator \ --parameters RegionIds="<region-1>,<region-2>" \ --parameters OrganizationalUnitIds="<OU-1>,<OU-2>"`	DevOps engineer

Option 3 - Enable DevOps Guru for specific stack resources across multiple accounts

Task	Description	Skills required

Deploy the AWS CDK stacks for creating IAM roles.	If you haven't already created the required IAM roles shown in the first option, do that first:
	
	1. Create the IAM `AWSCloudFormationStackSetAdministrationRole` role in the administrator (primary) account by running the following CLI command:
	
	`$cdk deploy CdkStackSetAdminRole --profile administrator`
	
	2. Create the IAM `AWSCloudFormationStackSetExecutionRole` role in all target accounts where you want to run the stack instances. To create this role, run the CLI commands:
	
	`$cdk deploy CdkStackSetExecRole \ --parameters AdministratorAccountId=<administrator-account-ID> \ --profile administrator`
		DevOps engineer

Task	Description	Skills required
	```$cdk deploy CdkStackSetExecRole \ --parameters AdministratorAccountId=<administrator-account-ID> \ --profile target``` For more information, see [Grant self-managed permissions](#) in the AWS CloudFormation documentation.	
Delete existing stacks.	If you already used the first option to enable DevOps Guru for all stack resources, you can delete the old stack by using the following command: ```$cdk destroy CdkDevopsGuruStackMultiAccReg --profile administrator``` Or, you can change the `RegionIds` parameter when you redeploy the stack to avoid a *Stacks already exist* error.	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Update the AWS CDK stack with a stack list.</td><td>1. Edit the file <code>/amazon-devopsguru-cdk-samples/lib/cdk-devopsguru-multi-acc-reg-spec-stack.ts</code> .
2. Under <code>Resources</code> , <code>CloudFormation</code> , <code>StackNames</code> , list the stacks for which you want to enable DevOps Guru. For demonstration purposes, the parameter specifies the <code>CdkInfrastructureStack</code> stack, but you can edit this entry based on your requirements.
3. Save and close the file.
4. To synthesize and update the stack template, run:
<pre>$cdk synth</pre></td><td>Data engineer</td></tr>
</table>

Task	Description	Skills required
Deploy the AWS CDK stack for enabling DevOps Guru for specific stack resources across multiple accounts.	The AWS CDK `CdkDevopsGuruStackMultiAccRegSpecStacks` stack enables DevOps Guru for specific stack resources across multiple accounts. To deploy the stack, run the following command: `$cdk deploy CdkDevopsGuruStackMultiAccRegSpecStacks \ --profile administrator \ --parameters AdministratorAccountId=<administrator-account-ID> \ --parameters TargetAccountId=<target-account-ID> \ --parameters RegionIds="<region-1>,<region-2>"` **Note** If you previously deployed this stack for option 1, change the `RegionIds` parameter (making sure to choose from [available Regions](https://)) to avoid a *Stacks already exist* error.	DevOps engineer

Deploy the AWS CDK infrastructure stack

Task	Description	Skills required
Deploy the sample serverless infrastructure stack.	The AWS CDK `CdkInfrastructureStack` stack deploys serverless components such as API Gateway, Lambda, and a DynamoDB table to demonstrate DevOps Guru insights. To deploy the stack, run the following command: `$cdk deploy CdkInfrastructureStack --profile administrator`	DevOps engineer
Insert sample records in DynamoDB.	Run the following command to populate the DynamoDB table with sample records. Provide the correct path for the `populate-shops-dynamodb-table.json` script. `$aws dynamodb batch-write-item \ --request-items file://scripts/populate-shops-dynamodb-table.json \ --profile administrator` The command displays the following output:	DevOps engineer

Task	Description	Skills required
	`{ "UnprocessedItems" : {} }`	

Task	Description	Skills required

Verify inserted records in DynamoDB.	To verify that the DynamoDB table includes the sample records from the `populate-shops-dynamodb-table.json` file, access the URL for the `ListRestApiEndpointMonitorOperator` API, which is published as an output of the AWS CDK stack. You can also find this URL in the **Outputs** tab of the AWS CloudFormation console for the `CdkInfrastructureStack` stack. The AWS CDK output would look similar to the following:
	`CdkInfrastructureStack.CreateRestApiMonitorOperatorEndpointD1D00045 = https://oure17c5vob.execute-api.<your-region>.amazonaws.com/prod/`
	
	`CdkInfrastructureStack.ListRestApiMonitorOperatorEndpointABBDB8D8 = https://cdff8icfrn4.execute-api.<your-region>.amazonaws.com/prod/`
		DevOps engineer

Task	Description	Skills required
Wait for resources to complete baselining.	This serverless stack has a few resources. We recommend that you wait for 2 hours before you carry out the next steps. If you deployed this stack in a production environment, it might take up to 24 hours to complete baselining, depending on the number of resources you selected to monitor in DevOps Guru.	DevOps engineer

Generate DevOps Guru insights

Task	Description	Skills required
Update the AWS CDK infrastructure stack.	To try out DevOps Guru insights, you can make some configuration changes to reproduce a typical operational issue. 1. Edit the file `/amazon-devopsguru-cdk-samples/lib/infrastructure-stack.ts` . 2. In the `DDB Table` section, change the read capacity for the DynamoDB table from 5 to 1. 3. Save and close the file. 4. Run the following commands to synthesize	DevOps engineer

Task	Description	Skills required

	and deploy the updated AWS CDK infrastructure stack:
	`$cdk synth`
	`$cdk deploy CdkInfras tructureStack -- profile administrator`	

Task	Description	Skills required
Inject HTTP requests on the API.	Inject ingress traffic in the form of HTTP requests on the `ListRestApiMonitorOperatorEndpointxxxx` API: 1. Edit the Python script `/amazon-devopsguru-cdk-samples/scripts/sendAPIRequest.py`. 2. Update the `url` variable with the API link for `ListRestApiMonitorOperatorEndpointxxxx`. You can find this URL in the output of the AWS CDK **deploy** command or on the AWS Cloudformation console, in the **Outputs** tab for the stack. 3. Save and close the file. 4. Run the Python script by using the command: `$python sendAPIRequest.py` 5. Make sure that you get a 200 status code. 6. You might need to run the script through multiple (preferably four) terminals	DevOps engineer

Task	Description	Skills required
	to inject traffic at a high rate. 7. After the script runs approximately 10 minutes in a loop, you can see an operational insight on the [DevOps Guru console]().	
Review DevOps Guru insights.	Under standard conditions, the DevOps Guru dashboard displays zero in the ongoing insights counter. If it detects an anomaly, it raises an alert in the form of an insight. In the navigation pane, choose **Insights** to see the details of the anomaly, including an overview, aggregated metrics, relevant events, and recommendations. For more information about reviewing insights, see the [Gaining operational insights with AIOps using Amazon DevOps Guru]() blog post.	DevOps engineer

Clean up

Task	Description	Skills required
Clean up and delete resources .	After you walk through this pattern, you should remove the resources you created to avoid incurring any	DevOps engineer

Task	Description	Skills required

	further charges. Run these commands:
	`$cdk destroy CdkDevopsGuruStackMultiAccReg --profile administrator`
	`$cdk destroy CdkDevopsguruStackOrgUnit --profile administrator`
	`$cdk destroy CdkDevopsGuruStackMultiAccRegSpecStacks --profile administrator`
	`$cdk destroy CdkInfrastructureStack --profile administrator`
	`$cdk destroy CdkStackSetAdminRole --profile administrator`
	`$cdk destroy CdkStackSetExecRole --profile administrator`
	`$cdk destroy CdkStackSetExecRole --profile target`	

Related resources

- Gaining operational insights with AIOps using Amazon DevOps Guru
- Easily configure Amazon DevOps Guru across multiple accounts and Regions using AWS CloudFormation StackSets
- DevOps Guru Workshop

Govern permission sets for multiple accounts by using Account Factory for Terraform

Anand Krishna Varanasi and Siamak Heshmati, Amazon Web Services

Summary

This pattern helps you integrate AWS Control Tower Account Factory Terraform (AFT) with AWS IAM Identity Center in order to configure permissions for multiple AWS accounts at scale. This approach uses custom AWS Lambda functions to automate permission set assignments to AWS accounts that are managed as an organization. This streamlines the process because it doesn't require manual intervention from your platform engineering team. This solution can enhance operational efficiency, security and consistency. It promotes a secure and standardized onboarding process for AWS Control Tower, making it indispensable for enterprises that prioritize agility and reliability for their cloud infrastructure.

Prerequisites and limitations

Prerequisites

- AWS accounts, managed through AWS Control Tower. For more information, see Getting started with AWS Control Tower.
- Account Factory for Terraform, deployed in a dedicated account in your environment. For more information, see Deploy AWS Control Tower Account Factory for Terraform.
- An IAM Identity Center instance, set up in your environment. For more information, see Getting started with IAM Identity Center.
- An active IAM Identity Center group, configured. For more information, see Add groups to your IAM Identity Center directory.
- Python version 3.9 or later, installed

Limitations

- This solution can be used only with accounts that are managed through AWS Control Tower. This solution is deployed by using Account Factory for Terraform.
- This pattern does not include instructions for setting up identity federation with an identity source. For more information about how to complete this set up, see IAM Identity Center identity source tutorials in the IAM Identity Center documentation.

Architecture

AFT overview

AFT sets up a Terraform pipeline that helps you provision and customize your accounts in AWS Control Tower. AFT follows a GitOps model that automates the processes of account provisioning in AWS Control Tower. You create an *account request Terraform file* and commit it to repository. This initiates the AFT workflow for account provisioning. After account provisioning is complete, AFT can automatically run additional customization steps. For more information, see AFT architecture in the AWS Control Tower documentation.

AFT provides the following main repositories:

- `aft-account-request` – This repository contains Terraform code to create or update AWS accounts.
- `aft-account-customizations` – This repository contains Terraform code to create or customize resources on a per-account basis.
- `aft-global-customizations` – This repository contains Terraform code to create or customize resources for all accounts, at scale.
- `aft-account-provisioning-customizations` – This repository manages customizations that are applied only to specific accounts created by and managed with AFT. For example, you might use this repository to customize user or groups assignments in IAM Identity Center or to automate account closures.

Solution overview

This custom solution includes an AWS Step Functions state machine and an AWS Lambda function that assign permission sets to users and groups for multiple accounts. The state machine deployed through this pattern operates in conjunction with pre-existing AFT `aft_account_provisioning_customizations` state machine. A user submits a request to update IAM Identity Center user and group assignments either when a new AWS account is created or after the account is created. They do this by pushing a change to the `aft-account-request` repository. The request to create or update an account initiates a stream in Amazon DynamoDB Streams. This starts the Lambda function, which updates IAM Identity Center users and groups for the target AWS accounts.

The following is an example of the parameters you can provide in the Lambda function for permission set assignments to target users and groups:

```
custom_fields = {
    "InstanceArn"          = "<Organization ID>",
    "PermissionSetArn"     = "<Permission set ARN>",
    "PrincipalId"          = "<Principal ID>",
  }
```

The following are the parameters in this statement:

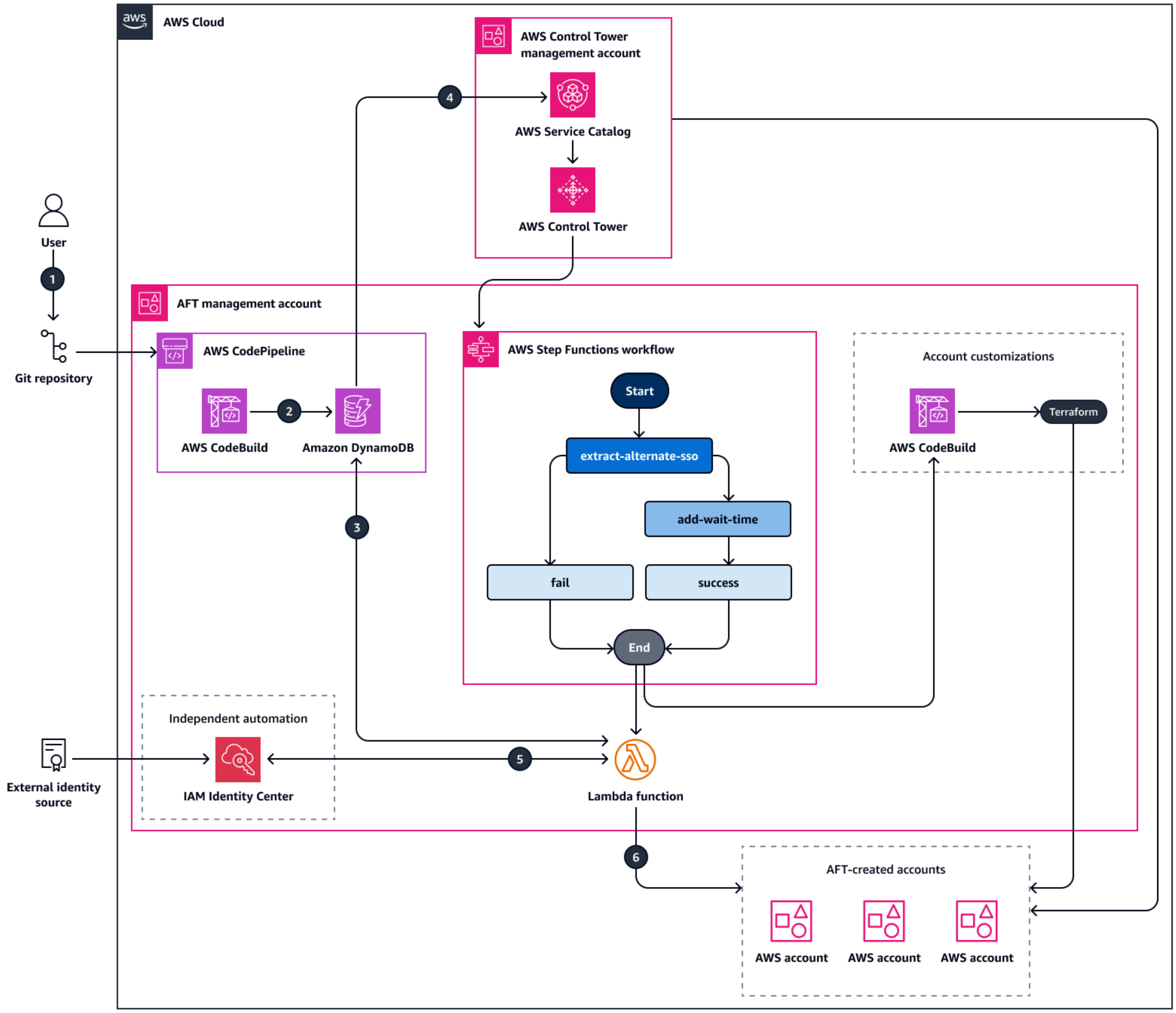
- `InstanceArn` – The Amazon Resource Name (ARN) of the organization
- `PermissionSetArn` – The ARN of the permission set
- `PrincipalId` – The identifier of a user or group in IAM Identity Center to which the permission set will be applied

> **Note**
>
> You must create the target permission set, users, and groups before running this solution.

While the `InstanceArn` value must remain consistent, you can modify the Lambda function to assign multiple permission sets to multiple target identities. The parameters for permission sets must end in `PermissionSetArn`, and the parameters for users and groups must end in `PrincipalId`. You must define both attributes. The following is an example of how to define multiple permission sets and target users and groups:

```
custom_fields = {
    "InstanceArn"                      = "<Organization ID>",
    "AdminAccessPermissionSetArn"      = "<Admin privileges permission set ARN>",
    "AdminAccessPrincipalId"           = "<Admin principal ID>",
    "ReadOnlyAccessPermissionSetArn"  = "<Read-only privileges permission set ARN>",
    "ReadOnlyAccessPrincipalId"        = "<Read-only principal ID>",
  }
```

The following diagram shows a step-by-step workflow of how the solution updates permissions sets for users and groups in the target AWS accounts at scale. When the user initiates an account creation request, AFT initiates the `aft-account-provisioning-framework` Step Functions state machine. This state machine starts the `extract-alternate-sso` Lambda function. The Lambda function assigns permissions sets to users and groups in the target AWS accounts. These users or groups can be from any configured identity source in IAM Identity Center. Examples of identity sources include Okta, Active Directory, or Ping Identity.



The diagram shows the following workflow when new accounts are created:

1. A user pushes a `custom_fields` change to the `aft-account-request` repository.
2. AWS CodePipeline starts an AWS CodeBuild job that records the user-defined metadata into the `aft-request-audit` Amazon DynamoDB table. This table has attributes to record user-defined metadata. The `ddb_event_name` attribute defines the type of AFT operation:

  - If the value is `INSERT`, then the solution assigns the permissions set to the target identities when the new AWS account is created.
  - If the value is `UPDATE`, then the solution assigns the permissions set to the target identities after the AWS account is created.
3. Amazon DynamoDB Streams initiates the `aft_alternate_sso_extract` Lambda function.
4. The `aft_alternate_sso_extract` Lambda function assumes an AWS Identity and Access Management (IAM) role in the AWS Control Tower management account.
5. The Lambda function assigns the permissions sets to the target users and groups by making an AWS SDK for Python (Boto3) [create_account_assignment](#) API call to IAM Identity Center. It retrieves the permission set and identity assignments from the `aft-request-audit` Amazon DynamoDB table.
6. When the Step Functions workflow completes, the permission sets are assigned to the target identities.

Automation and scale

AFT operates at scale by using AWS services such as CodePipeline, AWS CodeBuild, DynamoDB, and Lambda, which are highly scalable. For additional automation, you can integrate this solution with a ticket or issue management system, such as Jira. For more information, see the [Additional information](#) section of this pattern.

Tools

AWS services

- [Account Factory for Terraform (AFT)](#) is the main tool in this solution. The `aft-account-provisioning-customizations` repository contains the Terraform code for creating customizations for AWS accounts, such as custom IAM Identity Center user or group assignments.
- [Amazon DynamoDB](#) is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [AWS Step Functions](#) is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

Other tools

- Python is a general-purpose computer programming language.
- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code repository for AFT is available in the GitHub AWS Control Tower Account Factory for Terraform repository. The code for this pattern is available in the Govern SSO Assignments for AWS accounts using Account Factory for Terraform (AFT) repository.

Best practices

- Understand the AWS shared responsibility model.
- Follow the security recommendations for AWS Control Tower. For more information, see Security in AWS Control Tower.
- Follow the principle of least privilege. For more information, see Apply least-privilege permissions.
- Build specific and focused permission sets and IAM roles for groups and business units.

Epics

Deploy the solution

Task	Description	Skills required
Create an IAM role.	In the AWS Control Tower management account, use Terraform to create an IAM role. This role has cross-acc ount access and a trust policy that allows federated access from the identity provider. It also has permissions to grant access to other accounts through AWS Control Tower.	AWS DevOps, Cloud architect

Task	Description	Skills required

	The Lambda function will assume this role. Do the following:
	
	1. Download the `AFTCrossAccountRole.tf` file from the GitHub code repository.
	2. Modify the `AFTCrossAccountRole.tf` file as necessary for your AWS environment.
	3. In Terraform, enter the following commands to create this IAM role:
	
	`terraform init`
	`terraform plan`
	`terraform apply`
	
	4. Validate that the role was successfully deployed and has the expected cross-account access.	

Task	Description	Skills required
Customize the solution for your environment.	1. Enter the following command to clone the [Govern SSO Assignments for AWS accounts using Account Factory for Terraform (AFT)](https://github.com/aws-samples/aft-custom-sso-assignment) repository to your local workstation. `git clone https://github.com/aws-samples/aft-custom-sso-assignment.git` 2. In the `aft-account-provisioning-customizations/terraform` folder, open the `variables.tf` file. 3. Modify the variables as needed for your environment. 4. Save and close the `variables.tf` file. 5. Open the `account-request.tf` file in your `aft-account-request` repository. 6. Modify the `custom_fields` parameters to define the permission sets and target users and groups. For more information, see the [Architecture](#) section of this pattern.	AWS DevOps, Cloud architect

Task	Description	Skills required
	7. Save and close the `account-request.tf` file.	
Deploy the solution.	1. In the cloned repository, copy the contents of `terraform` folder, and then paste them into the `terraform` folder in the `aft-account-provisioning-customizations` repository. 2. In the AFT management account, start the `ct-aft-account-provisioning-customizations` pipeline. This deploys the custom solution. For instructions, see [Start a pipeline in CodePipeline](#). 3. Validate that the resources were successfully deployed in the AFT management account.	AWS DevOps, Cloud architect
Set up a code repository connection.	Set up a connection between the code repository where you will store the configuration files and your AWS account. For instructions, see the [Add third-party source providers to pipelines using CodeConnections](#) in the AWS CodePipeline documentation.	AWS DevOps, Cloud architect

Use the solution

Task	Description	Skills required
Start AFT pipeline to deploy a new account.	Follow the instructions in [Provision a new account with AFT](#) in order to start the pipeline that creates a new AWS account in your AWS Control Tower environment. Wait for the account creation process to complete.	AWS DevOps, Cloud architect
Validate the changes.	1. Open the [AWS IAM Identity Center console](#). 2. In the list of accounts, choose the newly created account. 3. Validate that permission sets have been assigned to grant access to the target users and groups.	AWS DevOps, Cloud architect

Troubleshooting

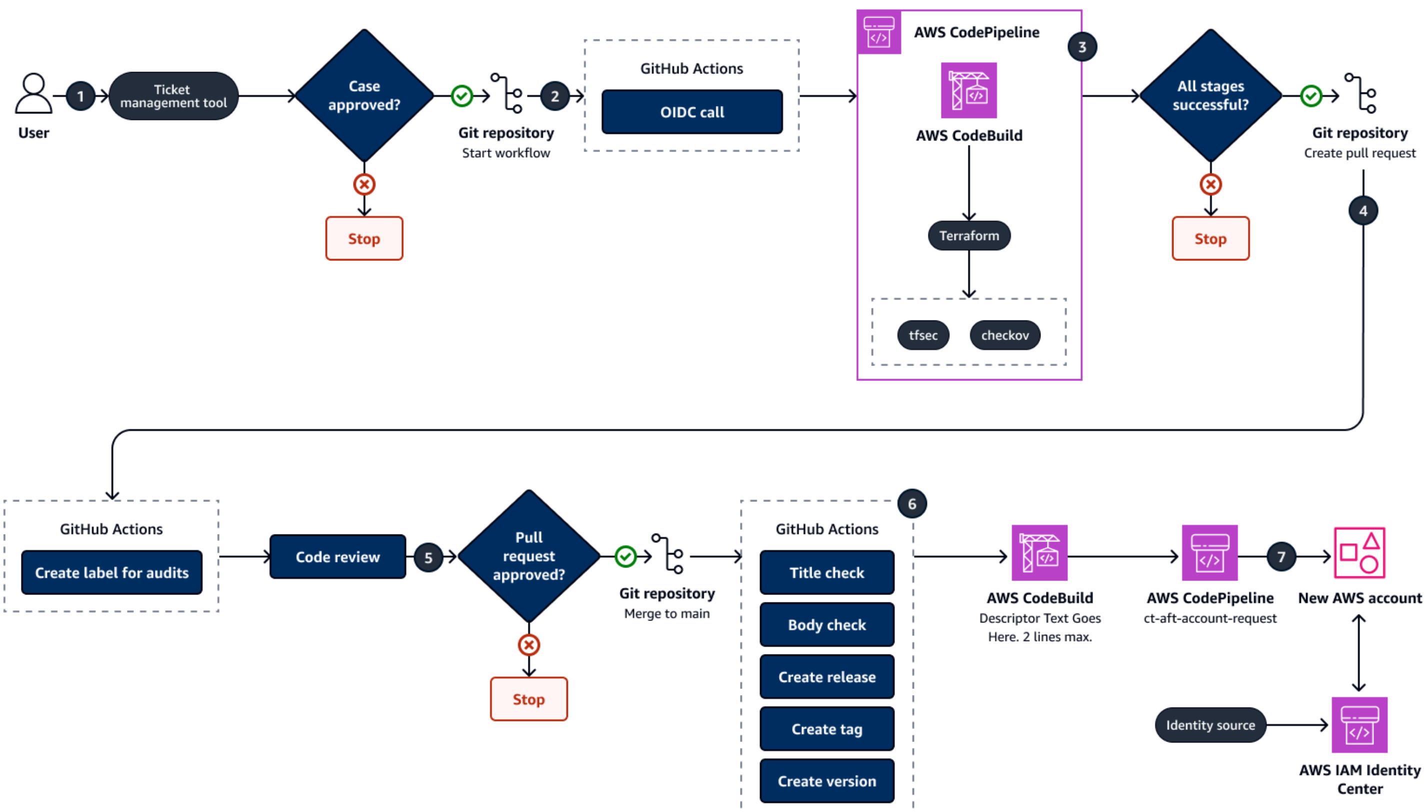
Issue	Solution
Permission set assignment is not working.	Make sure the group ARN, organization id, and Lambda parameters are correct. For examples, see the *Solution overview* section of this pattern.
Updating code in the repository does not start the pipeline.	This issue is related to the connectivity between your AWS account and the repository. In the AWS Management Console, validate that the connection is active. For more

Issue	Solution
	information, see GitHub connections in the AWS CodePipeline documentation.

Additional information

Integrating with a ticket management tool

You can choose to integrate this solution with a ticket or issue management tool, such as Jira or ServiceNow. The following diagram shows an example workflow for this option. You can integrate the ticket management tool with the AFT solution repositories by using your tool's connectors. For Jira connectors, see Integrate Jira with GitHub. For ServiceNow connectors, see Integrating with GitHub. You can even build custom solutions that require users to provide a ticket ID as part of the pull request approval. If a request to create a new AWS account by using AFT is approved, that event could initiate a workflow that adds custom fields to the `aft-account-request` GitHub repository. You can design any custom workflow that meets the requirements of your use case.



The diagram shows the following workflow:

1. Users request a custom permission set assignment in a ticket management tool, such as Jira.
2. After the case is approved, a workflow begins to update the permission set assignment. (Optional) You can use plugins for custom automation of this step.
3. Operators send the Terraform code with the updated permission set parameters to the `aft-account-request` repository into a development or feature branch.
4. GitHub Actions initiates AWS CodeBuild by using an OpenID Connect (OIDC) call. CodeBuild performs infrastructure as code (IaC) security scans by using tools such as [tfsec] and [checkov]. It warns the operators of any security violations.
5. If no violations are found, GitHub Actions creates an automated pull request and assigns a code review to the code owners. It also creates a tag for the pull request.
6. If the code owner approves the code review, another GitHub Actions workflow starts. It checks pull request standards, including:
   - If the pull request title meets requirements.
   - If pull request body contains approved case numbers.
   - If the pull request is properly tagged.
7. If the pull requests meets standards, GitHub Actions starts the AFT product workflow. It uses starts the `ct-aft-account-request` pipeline in AWS CodePipeline. This pipeline starts the `aft-account-provisioning-framework` custom state machine in Step Functions. This state machine works as previously described in the *Solution overview* section of this pattern.

Implement Account Factory for Terraform (AFT) by using a bootstrap pipeline

Vinicius Elias and Edgar Costa Filho, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

This pattern provides a simple and secure method for deploying AWS Control Tower Account Factory for Terraform (AFT) from the management account of AWS Organizations. The core of the solution is an AWS CloudFormation template that automates the AFT configuration by creating a Terraform pipeline, which is structured to be easily adaptable for initial deployment or subsequent updates.

Security and data integrity are top priorities at AWS, so the Terraform state file, which is a critical component that tracks the state of the managed infrastructure and configurations, is securely stored in an Amazon Simple Storage Service (Amazon S3) bucket. This bucket is configured with several security measures, including server-side encryption and policies to block public access, to help ensure that your Terraform state is safeguarded against unauthorized access and data breaches.

The management account orchestrates and oversees the entire environment, so it is a critical resource in AWS Control Tower. This pattern follows AWS best practices and ensures that the deployment process is not only efficient but also aligns with security and governance standards, to offer a comprehensive, secure, and efficient way to deploy AFT in your AWS environment.

For more information about AFT, see the AWS Control Tower documentation.

Prerequisites and limitations

Prerequisites

- A basic AWS multi-account environment with the following accounts at the minimum: management account, Log Archive account, Audit account, and one additional account for AFT management.
- An established AWS Control Tower environment. The management account should be properly configured, because the CloudFormation template will be deployed within it.

- The necessary permissions in the AWS management account. You'll need sufficient permissions to create and manage resources such as S3 buckets, AWS Lambda functions, AWS Identity and Access Management (IAM) roles, and AWS CodePipeline projects.
- Familiarity with Terraform. Understanding Terraform's core concepts and workflow is important because the deployment involves generating and managing Terraform configurations.

Limitations

- Be aware of the AWS resource quotas in your account. The deployment might create multiple resources, and encountering service quotas could impede the deployment process.
- The template is designed for specific versions of Terraform and AWS services. Upgrading or changing versions might require template modifications.
- The template doesn't support self-managed version control system (VCS) services such as GitHub Enterprise.

Product versions

- Terraform version 1.6.6 or later
- AFT version 1.11 or later

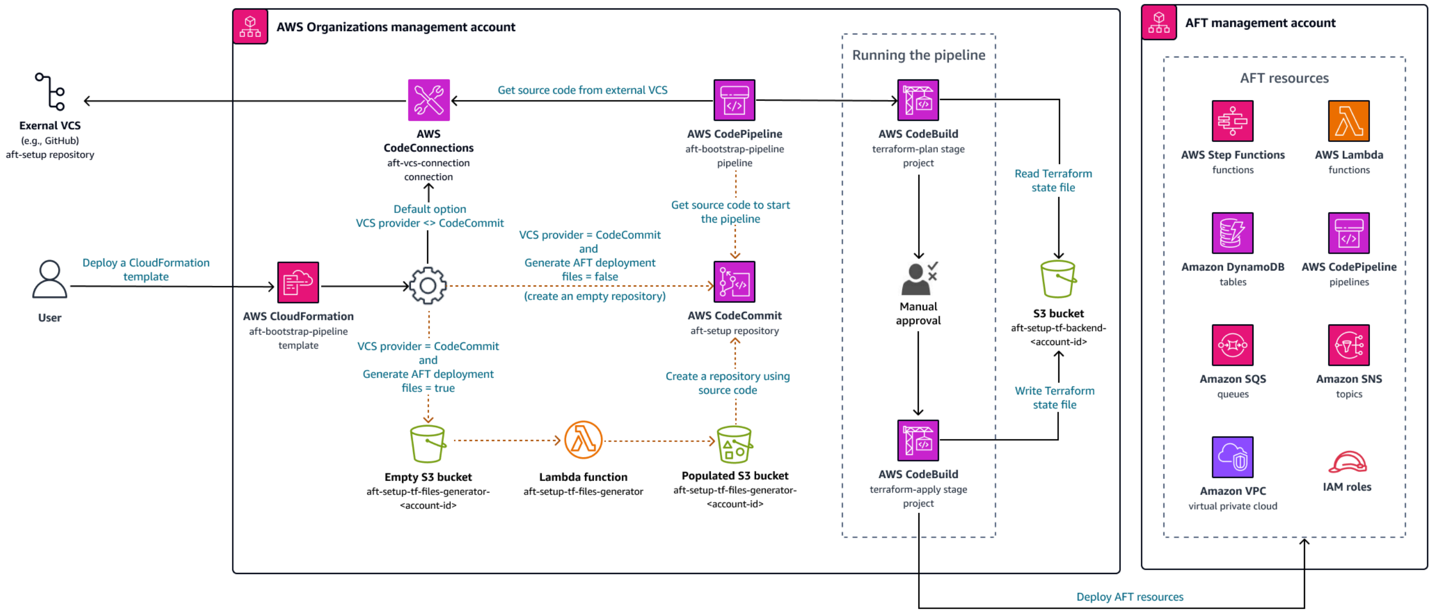
Architecture

Target technology stack

- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- AWS CodePipeline
- Amazon EventBridge
- IAM
- AWS Lambda
- Amazon S3

Target architecture

The following diagram illustrates the implementation discussed in this pattern.



The workflow consists of three main tasks: creating the resources, generating the content, and running the pipeline.

Creating the resources

The [CloudFormation template that's provided with this pattern] creates and sets up all required resources, depending on the parameters you select when you deploy the template. At the minimum, the template creates the following resources:

- A CodePipeline pipeline to implement the AFT
- An S3 bucket to store the Terraform state file that's associated with the AFT implementation
- Two CodeBuild projects to implement the Terraform plan and apply commands in different stages of the pipeline
- IAM roles for CodeBuild and CodePipeline services
- A second S3 bucket to store pipeline runtime artifacts

Depending on the VCS provider you select (CodeCommit or external VCS), the template creates the following resources.

- For **CodeCommit**:
  - A CodeCommit repository to store the AFT Terraform bootstrap code

  - An EventBridge rule to capture CodeCommit repository changes on the `main` branch
  - Another IAM role for the EventBridge rule
- For any other **external VCS provider**, such as GitHub:
  - An AWS CodeConnections connection

Additionally, when you select CodeCommit as the VCS provider, if you set the `Generate AFT Files` parameter to `true`, the template creates these additional resources to generate the content:

- An S3 bucket to store the generated content and to be used as the source of the CodeCommit repository
- A Lambda function to process the given parameters and generate the appropriate content
- An IAM function to run the Lambda function
- A CloudFormation custom resource that runs the Lambda function when the template is deployed

Generating the content

To generate the AFT bootstrap files and their content, the solution uses a Lambda function and an S3 bucket. The function creates a folder in the bucket, and then creates two files inside the folder: `main.tf` and `backend.tf`. The function also processes the provided CloudFormation parameters and populates these files with predefined code, replacing the respective parameter values.

To view the code that's used as a template to generate the files, see the solution's [GitHub repository](). Basically, the files are generated as follows.

main.tf

```
module "aft" {
  source = "github.com/aws-ia/terraform-aws-control_tower_account_factory?
ref=<aft_version>"

  # Required variables
  ct_management_account_id  = "<ct_management_account_id>"
  log_archive_account_id    = "<log_archive_account_id>"
  audit_account_id          = "<audit_account_id>"
  aft_management_account_id = "<aft_management_account_id>"
  ct_home_region            = "<ct_home_region>"
```

```
  # Optional variables
  tf_backend_secondary_region = "<tf_backend_secondary_region>"
  aft_metrics_reporting       = "<false|true>"

  # AFT Feature flags
  aft_feature_cloudtrail_data_events      = "<false|true>"
  aft_feature_enterprise_support          = "<false|true>"
  aft_feature_delete_default_vpcs_enabled = "<false|true>"

  # Terraform variables
  terraform_version      = "<terraform_version>"
  terraform_distribution = "<terraform_distribution>"

  # VCS variables (if you have chosen an external VCS)
  vcs_provider                                  = "<github|githubenterprise|gitlab|
gitlabselfmanaged|bitbucket>"
  account_request_repo_name                     = "<org-name>/aft-account-request"
  account_customizations_repo_name              = "<org-name>/aft-account-
customizations"
  account_provisioning_customizations_repo_name = "<org-name>/aft-account-provisioning-
customizations"
  global_customizations_repo_name               = "<org-name>/aft-global-
customizations"

}
```

backend.tf

```
terraform {
  backend "s3" {
    region = "<aft-main-region>"
    bucket = "<s3-bucket-name>"
    key    = "aft-setup.tfstate"
  }
}
```

During the CodeCommit repository creation, if you set the `Generate AFT Files` parameter to `true`, the template uses the S3 bucket with the generated content as the source of the `main` branch to automatically populate the repository.

Running the pipeline

After the resources have been created and the bootstrap files have been configured, the pipeline runs. The first stage (*Source*) fetches the source code from the main branch of the repository, and the second stage (*Build*) runs the Terraform plan command and generates the results to be reviewed. In the third stage (*Approval*), the pipeline waits for a manual action to approve or reject the last stage (*Deploy*). At the last stage, the pipeline runs the Terraform `apply` command by using the result of the previous Terraform `plan` command as input. Finally, a cross-account role and the permissions in the management account are used to create the AFT resources in the AFT management account.

> **Note**
>
> If you select an external VCS provider, you will need to authorize the connection with your VCS provider credentials. To complete the setup, follow the steps in [Update a pending connection](#) in the AWS Developer Tools console documentation.

Tools

AWS services

- [AWS CloudFormation](#) helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- [AWS CodeBuild](#) is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- [AWS CodeCommit](#) is a version control service that helps you privately store and manage Git repositories without needing to manage your own source control system.
- [AWS CodePipeline](#) helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- [AWS CodeConnections](#) enables AWS resources and services, such as CodePipeline, to connect to external code repositories, such as GitHub.
- [AWS Lambda](#) is a compute service that runs your code in response to events and automatically manages compute resources, providing a fast way to create a modern, serverless application for production.
- [AWS SDK for Python (Boto3)](#) is a software development kit that helps you integrate your Python application, library, or script with AWS services.

Other tools

- Terraform is an infrastructure as code (IaC) tool that lets you build, change, and version infrastructure safely and efficiently. This includes low-level components such as compute instances, storage, and networking; and high-level components such as DNS entries and SaaS features.
- Python is an easy to learn, powerful programming language. It has efficient high-level data structures and provides a simple but effective approach to object-oriented programming.

Code repository

The code for this pattern is available in the GitHub AFT bootstrap pipeline repository.

For the official AFT repository, see AWS Control Tower Account Factory for Terraform in GitHub.

Best practices

When you deploy AFT by using the provided CloudFormation template, we recommend that you follow best practices to help ensure a secure, efficient, and successful implementation. Key guidelines and recommendations for implementing and operating the AFT include the following.

- **Thorough review of parameters**: Carefully review and understand each parameter in the CloudFormation template. Accurate parameter configuration is crucial for the correct setup and functioning of AFT.
- **Regular template updates**: Keep the template updated with the latest AWS features and Terraform versions. Regular updates help you take advantage of new functionality and maintain security.
- **Versioning**: Pin your AFT module version and use a separate AFT deployment for testing if possible.
- **Scope**: Use AFT only to deploy infrastructure guardrails and customizations. Do not use it to deploy your application.
- **Linting and validation**: The AFT pipeline requires a linted and validated Terraform configuration. Run lint, validate, and test before pushing the configuration to AFT repositories.
- **Terraform modules**: Build reusable Terraform code as modules, and always specify the Terraform and AWS provider versions to match your organization's requirements.

Epics

Set up and configure the AWS environment

Task	Description	Skills required
Prepare the AWS Control Tower environment.	Set up and configure AWS Control Tower in your AWS environment to ensure centralized management and governance for your AWS accounts. For more information, see [Getting started with AWS Control Tower](#) in the AWS Control Tower documentation.	Cloud administrator
Launch the AFT management account.	Use the AWS Control Tower Account Factory to launch a new AWS account to serve as your AFT management account. For more information, see [Provision accounts with AWS Service Catalog Account Factory](#) in the AWS Control Tower documentation.	Cloud administrator

Deploy the CloudFormation template in the management account

Task	Description	Skills required
Launch the CloudFormation template.	In this epic, you deploy the CloudFormation template provided with this solution to set up the AFT bootstrap pipeline in your AWS	Cloud administrator

Task	Description	Skills required
	management account. The pipeline deploys the AFT solution in the AFT management account that you set up in the previous epic. **Step 1: Open the AWS CloudFormation console** • Sign in to the AWS Management Console and open the [AWS CloudFormation console](). Make sure that you are operating within the correct AWS Control Tower main Region. **Step 2: Create a new stack** 1. Choose to create a new stack. 2. Select the option to upload a template file, and upload the [CloudFormation template]() that's provided with this pattern. **Step 3: Configure stack parameters** • `VCS Provider`: Select the version control system (VCS) provider to use. You can select an external	

Task	Description	Skills required
	VCS such as GitHub, or use CodeCommit if your account is allowed to use the service. • `Repository Name` : Specify the repository name for storing the AFT bootstrap module. For external VCS providers, use the full path, including the organization name (for example, `my-github-org/my-repo`). • `Branch Name`: Specify the source repository branch. • `CodeBuild Docker Image`: Choose the file to use as the CodeBuild Docker base image. • If you set the VCS provider to an option other than CodeCommit, go to step 8. **Step 4: Decide on file generation** • If you selected CodeCommit as your VCS provider, you can use the `Generate AFT Files` parameter to control the generation of default AFT deployment files. Set this parameter to:	

Task	Description	Skills required
	`true` to automatically create and store AFT deployment files in the specified repository. `false` if you want to manually handle the file creation or already have the files in place. - If you selected `false`, go to step 8; otherwise, follow steps 5–7 first. **Step 5: Fill in AWS Control Tower and AFT account details** - If you set the `Generate AFT Files` parameter to `true`, provide the following AWS Control Tower and AFT account-specific information. `Log Archive Account ID`: The ID of the Log Archive account ID in AWS Control Tower. `Audit Account ID`: The ID of the Audit account in AWS Control Tower. `AFT Management Account ID`: The ID of the AFT management	

Task	Description	Skills required

	account that you created in the first epic.
	• `AFT Main Region` and `AFT Secondary Region`: The main and secondary AWS Regions for AFT deployment.
	
	Step 6: Configure AFT options
	
	• Set up metrics reporting:
	  • `AFT Enable Metrics Reporting` : Enable or disable AFT metrics reporting. For more information, see [Operational metrics](#) in the AWS Control Tower documentation.
	
	• Set AFT feature options:
	  • `Enable AFT CloudTrail Data Events`: Enable CloudTrail data events in all AFT managed accounts. For more information, see [AWS CloudTrail data events](#) in the AWS Control Tower documentation.
	  • `Enable AFT Enterprise Support` :	

Task	Description	Skills required
	Enable Enterprise Support in all AFT managed accounts. For more information, see [AWS Enterprise Support plan](#) in the AWS Control Tower documentation. • `Enable AFT Delete Default VPC`: Delete all VPCs in the AFT management account only. For more information, see [Delete the AWS default VPC](#) in the AWS Control Tower documentation. **Step 7: Specify versions** • `AFT Terraform Version`: Choose the version of Terraform to use in AFT pipelines. • `AFT Version`: Define the AFT version for deployment. Keep the default setting (`latest`) to use the most current AFT version. **Step 8: Review and create the stack** • Review all the parameters and settings. If everythin	

Task	Description	Skills required
	g is in order, proceed to create the stack. **Step 9: Monitor stack creation** • AWS CloudFormation provisions and configures the resources you defined. Monitor the stack creation process on the CloudFormation console. This process might take several minutes. **Step 10: Verify the deployment** • When the stack status shows **CREATE_COMPLETE**, verify that all resources have been correctly created. • In the **Outputs** section, note the `TerraformBackendBucketName` value.	

Populate and validate the AFT bootstrap repository and pipeline

Task	Description	Skills required
Option 1: Populate the AFT bootstrap repository for an external VCS.	If you set the VCS provider to an external VCS (not to	Cloud administrator

Task	Description	Skills required
	CodeCommit), follow these steps. (Optional) After you deploy the CloudFormation template, you can populate or validate the content in the newly created AFT bootstrap repository, and test whether the pipeline has run successfully. **Step 1: Update the connection** 1. On the [CodePipeline console](https://console.aws.amazon.com/codesuite/codepipeline/home), in the navigation pane, choose **Settings**, **Connections**. 2. Select the `aft-vcs-connection` connection. It should be in `Pending` status. 3. Choose **Update pending connection** and follow the instructions in [Update a pending connection](https://docs.aws.amazon.com/dtconsole/latest/userguide/connections-update.html) in the Developer Tools console documentation. 4. When the connection is in `Available` status, move to the next step. **Step 2: Populate the repository**	

Task	Description	Skills required

	1. Use your external VCS credentials to clone the repository that you specified in the template to your local machine. If you kept the default name, the repository is called `aft-setup` .
	2. In the repository, create a folder called `terraform` with two empty files inside it: `backend.tf` and `main.tf`.
	3. Open the `backend.tf` file and add this code snippet:
	
	`terraform {`
	`  backend "s3" {`
	`    region = "<aft-main-region>"`
	`    bucket = "<s3-bucket-name>"`
	`    key    = "aft-setup"`
	`  }`
	`}`
	
	In the file:
	• Replace `<aft-main-region>` with the main AFT Region. This should match the AWS Control Tower main Region.	

Task	Description	Skills required
	- Replace `<s3-bucket-name>` with the name of the Terraform backend bucket. You can find this in the `TerraformBackendBucketName` output generated by the CloudFormation template you deployed earlier. 4. Open the `main.tf` file and use one of the examples available in the [AFT repository] to deploy AFT. For example, you can work with your preferred VCS provider (CodeCommit, GitHub, or Bitbucket) or customize the AFT VPC. For more AFT input options, see the [README file] in the AFT repository. **Step 2: Commit and push your changes** - After you have created and populated the folder and files, confirm your changes, and upload the code to the repository. The pipeline starts automatically, runs through the Source and Build stages, and then waits	

Task	Description	Skills required
	for an Approval action before the Deploy stage.	

Task	Description	Skills required
Option 2: Populate the AFT bootstrap repository for CodeCommit.	If you set the VCS provider to CodeCommit, follow these steps. (Optional) After you deploy the CloudFormation template, you can populate or validate the content in the newly created AFT bootstrap repository, and test whether the pipeline has run successfully. If you set the `Generate AFT Files` parameter to `true`, skip to the next story (validating the pipeline). **Step 1: Populate the repository** 1. Open the [AWS CodeCommit console](#) and select the newly created repository. If you kept the default name, the repository name should be `aft-setup` . 2. Clone the repository to your local machine by using SSH, HTTPS, or HTTPS (GRC), and open it in an editor. 3. Create a folder called `terraform` and two empty files inside it:	Cloud administrator

Task	Description	Skills required

	`backend.tf` and `main.tf`.
	4. Open the `backend.tf` file and add this code snippet:
	`terraform {`
	`  backend "s3" {`
	`    region = "<aft-main-region>"`
	`    bucket = "<s3-bucket-name>"`
	`    key    = "aft-setup"`
	`  }`
	`}`
	In the file:
	• Replace `<aft-main-region>` with the main AFT Region. This should match the AWS Control Tower main Region.
	• Replace `<s3-bucket-name>` with the name of the Terraform backend bucket. You can find this in the `TerraformBackendBucketName` output generated by the CloudFormation template you deployed earlier.
	5. Open the `main.tf` file and use one of the examples	

Task	Description	Skills required
	available in the AFT repository to deploy AFT. For example, you can work with your preferred version control system (VCS) provider (CodeCommit, GitHub, or Bitbucket) or customize the AFT VPC. For more AFT input options, see the README file in the AFT repository. **Step 2: Commit and push your changes** • After you have created and populated the folder and files, confirm your changes, and upload the code to the repository. The pipeline starts automatically, runs through the Source and Build stages, and then waits for an Approval action before the Deploy stage.	

Task	Description	Skills required
Validate the AFT bootstrap pipeline.	**Step 1: View the pipeline** • Open the [CodePipeline console](https://console.aws.amazon.com/codepipeline/) and check whether the `aft-bootstrap-pipeline` pipeline was started successfully. It should be running a Terraform plan or waiting for a manual approval action. **Step 2: Approve the Terraform plan results** • You can review the results of the Terraform plan by looking at the execution logs of the Build stage, and then approve or reject the execution on the Approval stage. If you approve, the pipeline starts deploying AFT resources in the provided AFT management account. **Step 3: Wait for the deployment** • Wait for the pipeline to run successfully. This should take around 30 minutes. Any failures you might encounter are often caused	Cloud administrator

Task	Description	Skills required
	by API quotas. In these cases, you can rerun the pipeline to continue the deployment. **Step 4: Check created resources** • Access the AFT managemen t account and confirm that the resources have been created.	

Troubleshooting

Issue	Solution
The custom Lambda function included in the CloudFormation template fails during deployment.	Check the Amazon CloudWatch logs for the Lambda function to identify the error. The logs provide detailed information and can help pinpoint the specific issue. Confirm that the Lambda function has the necessary permissions and that the environment variables have been set correctly.
You encounter failures in resource creation or management caused by inadequate permissio ns.	Review the IAM roles and policies that are attached to the Lambda function, CodeBuild, and other services involved in the deploymen t. Confirm that they have the necessary permissions. If there are permission issues, adjust the IAM policies to grant the required access.

Issue	Solution
You're using an outdated version of the CloudFormation template with newer AWS services or Terraform versions.	Regularly update the CloudFormation template to be compatible with the latest AWS and Terraform releases. Check the release notes or documentation for any version-s pecific changes or requirements.
You reach AWS service quotas during deployment.	Before you deploy the pipeline, check AWS service quotas for resources such as S3 buckets, IAM roles, and Lambda functions . Request increases if necessary. For more information, see [AWS service quotas](#) on the AWS website.
You encounter errors due to incorrect input parameters in the CloudFormation template.	Double-check all input parameters for typos or incorrect values. Confirm that resource identifiers, such as account IDs and Region names, are accurate.

Related resources

To implement this pattern successfully, review the following resources. These resources provide additional information and guidance that can be invaluable in setting up and managing AFT by using AWS CloudFormation.

AWS documentation:

- AWS Control Tower User Guide offers detailed information on setting up and managing AWS Control Tower.
- AWS CloudFormation documentation provides insights into CloudFormation templates, stacks, and resource management.

IAM policies and best practices:

- Security best practices in IAM explains how to help secure AWS resources by using IAM roles and policies.

Terraform on AWS:

- Terraform AWS Provider documentation provides comprehensive information about using Terraform with AWS.

AWS service quotas:

- AWS service quotas provides information about how to view AWS service quotas and how to request increases.

Manage AWS Service Catalog products in multiple AWS accounts and AWS Regions

Ram Kandaswamy, Amazon Web Services

Summary

Amazon Web Services (AWS) Service Catalog simplifies and accelerates the governance and distribution of infrastructure as code (IaC) templates for enterprises. You use AWS CloudFormation templates to define a collection of AWS resources (*stacks*) required for a product. AWS CloudFormation StackSets extends this functionality by enabling you to create, update, or delete stacks across multiple accounts and AWS Regions with a single operation.

AWS Service Catalog administrators create products by using CloudFormation templates that are authored by developers, and publish them. These products are then associated with a portfolio, and constraints are applied for governance. To make your products available to users in other AWS accounts or organizational units (OUs), you typically share your portfolio with them. This pattern describes an alternative approach for managing AWS Service Catalog product offerings that is based on AWS CloudFormation StackSets. Instead of sharing portfolios, you use stack set constraints to set AWS Regions and accounts where your product can be deployed and used. By using this approach, you can provision your AWS Service Catalog products in multiple accounts, OUs, and AWS Regions, and manage them from a central location, while meeting your governance requirements.

Benefits of this approach:

- The product is provisioned and managed from the primary account, and not shared with other accounts.
- This approach provides a consolidated view of all provisioned products (stacks) that are based on a specific product.
- Configuration with AWS Service Management Connector is easier, because it targets only one account.
- It's easier to query and use products from AWS Service Catalog.

Prerequisites and limitations

Prerequisites

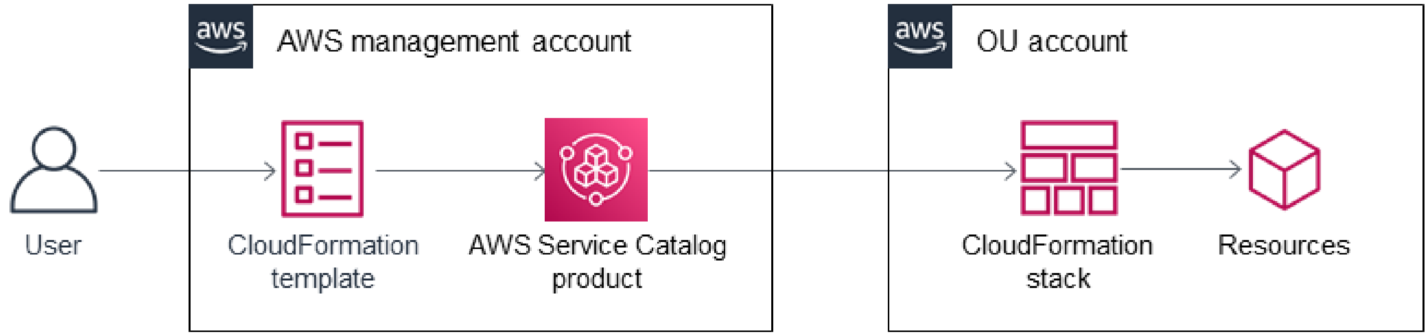
- AWS CloudFormation templates for IaC and versioning
- Multi-account setup and AWS Service Catalog for provisioning and managing AWS resources

Limitations

- This approach uses AWS CloudFormation StackSets, and the limitations of StackSets apply:
  - StackSets doesn't support CloudFormation template deployment through macros. If you're using a macro to preprocess the template, you won't be able to use a StackSets-based deployment.
  - StackSets provides the ability to disassociate a stack from the stack set, so you can target a specific stack to fix an issue. However, a disassociated stack cannot be re-associated with the stack set.
- AWS Service Catalog autogenerates StackSet names. Customization isn't currently supported.

Architecture

Target architecture



1. The user creates an AWS CloudFormation template to provision AWS resources, in JSON or YAML format.
2. The CloudFormation template creates a product in AWS Service Catalog, which is added to a portfolio.
3. The user creates a provisioned product, which creates CloudFormation stacks in the target accounts.
4. Each stack provisions the resources specified in the CloudFormation templates.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Service Catalog helps you centrally manage catalogs of IT services that are approved for AWS. End users can quickly deploy only the approved IT services they need, following the constraints set by your organization.

Epics

Provision products across accounts

Task	Description	Skills required
Create a portfolio.	A portfolio is a container that includes one or more products that are grouped together based on specific criteria. Using a portfolio for your products helps you apply common constraints across your product set. To create a portfolio, follow the instructions in the AWS Service Catalog documentation. If you're using the AWS CLI, here's an example command:	AWS Service Catalog, IAM

Task	Description	Skills required
	`aws servicecatalog create-portfolio --provider-name my-provider --display-name my-portfolio` For more information, see the [AWS CLI documentation]().	
Create a CloudFormation template.	Create a CloudFormation template that describes the resources. Resource property values should be parameterized where applicable.	AWS CloudFormation, JSON/ YAML

Task	Description	Skills required
Create a product with version information.	The CloudFormation template becomes a product when you publish it in the AWS Service Catalog. Provide values for the optional version detail parameters, such as version title and description; this will be helpful for querying for the product later. To create a product, follow the instructions in the [AWS Service Catalog documentation](). If you're using the AWS CLI, an example command is: `aws servicecatalog create-product --cli-input-json file://create-product-input.json` where `create-product-input.json` is the file that passes the parameters for the product. For an example of this file, see the *Additional information* section. For more information, see the [AWS CLI documentation]().	AWS Service Catalog

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Apply constraints.</td><td>Apply stack set constraints to the portfolio, to configure product deployment options such as multiple AWS accounts, Regions, and permissions. For instructions, see the AWS Service Catalog documentation.</td><td>AWS Service Catalog</td></tr>
<tr><td>Add permissions.</td><td>Provide permissions to users so that they can launch the products in the portfolio. For console instructions, see the AWS Service Catalog documentation. If you're using the AWS CLI, here's an example command:

<pre><code>aws servicecatalog
 associate-principal-
with-portfolio \
    --portfolio-id
 port-2s6abcdefwdh4 \
    --principal-arn
 arn:aws:iam::44445
5556666:role/Admin \
    --principal-type
 IAM</code></pre>

For more information, see the AWS CLI documentation.</td><td>AWS Service Catalog, IAM</td></tr>
</table>

Task	Description	Skills required
Provision the product.	A provisioned product is a resourced instance of a product. Provisioning a product based on a CloudFormation template launches a CloudFormation stack and its underlying resources. Provision the product by targeting the applicable AWS Regions and accounts, based on stack set constraints. In the AWS CLI, here's an example command: `aws servicecatalog provision-product \ --product-id prod-abcdfz3syn2rg \ --provisioning-artifact-id pa-abc347pcsccfm \ --provisioned-product-name "mytestppname3"` For more information, see the [AWS CLI documentation]().	AWS Service Catalog

Related resources

References

- [Overview of AWS Service Catalog]()
- [Using AWS CloudFormation StackSets]()

Tutorials and videos

- [AWS re:Invent 2019: Automate everything: Options and best practices](#) (video)

Additional information

When you use the `create-product` command, the `cli-input-json` parameter points to a file that specifies information such as product owner, support email, and CloudFormation template details. Here's an example of such a file:

```
{
   "Owner": "Test admin",
      "SupportDescription": "Testing",
         "Name": "SNS",
            "SupportEmail": "example@example.com",
            "ProductType": "CLOUD_FORMATION_TEMPLATE",
               "AcceptLanguage": "en",
                  "ProvisioningArtifactParameters": {
                     "Description": "SNS product",
                        "DisableTemplateValidation": true,
                           "Info": {
                              "LoadTemplateFromURL": "<url>"
                     },
                           "Name": "version 1"
}
```

Migrate an AWS member account from AWS Organizations to AWS Control Tower

Rodolfo Jr. Cerrada, Amazon Web Services

Summary

This pattern describes how to migrate an Amazon Web Services (AWS) account from AWS Organizations, where it is a member account that's governed by a management account, to AWS Control Tower. By enrolling the account in AWS Control Tower, you can take advantage of preventive and detective guardrails and features that streamline your account governance. You might also want to migrate your member account if your AWS Organizations management account has been compromised, and you want to move member accounts to a new organization that is governed by AWS Control Tower.

AWS Control Tower provides a framework that combines and integrates the capabilities of several other AWS services, including AWS Organizations, and ensures consistent compliance and governance across your multi-account environment. With AWS Control Tower, you can follow a set of prescribed rules and definitions that extend the capabilities of AWS Organizations. For example, you can use guardrails to ensure that security logs and necessary cross-account access permissions are created, and not altered.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Control Tower set up in your target organization in AWS Organizations (for instructions, see Setting up in the AWS Control Tower documentation)
- Administrator credentials for AWS Control Tower (member of the **AWSControlTowerAdmins** group)
- Administrator credentials for the source AWS account

Limitations

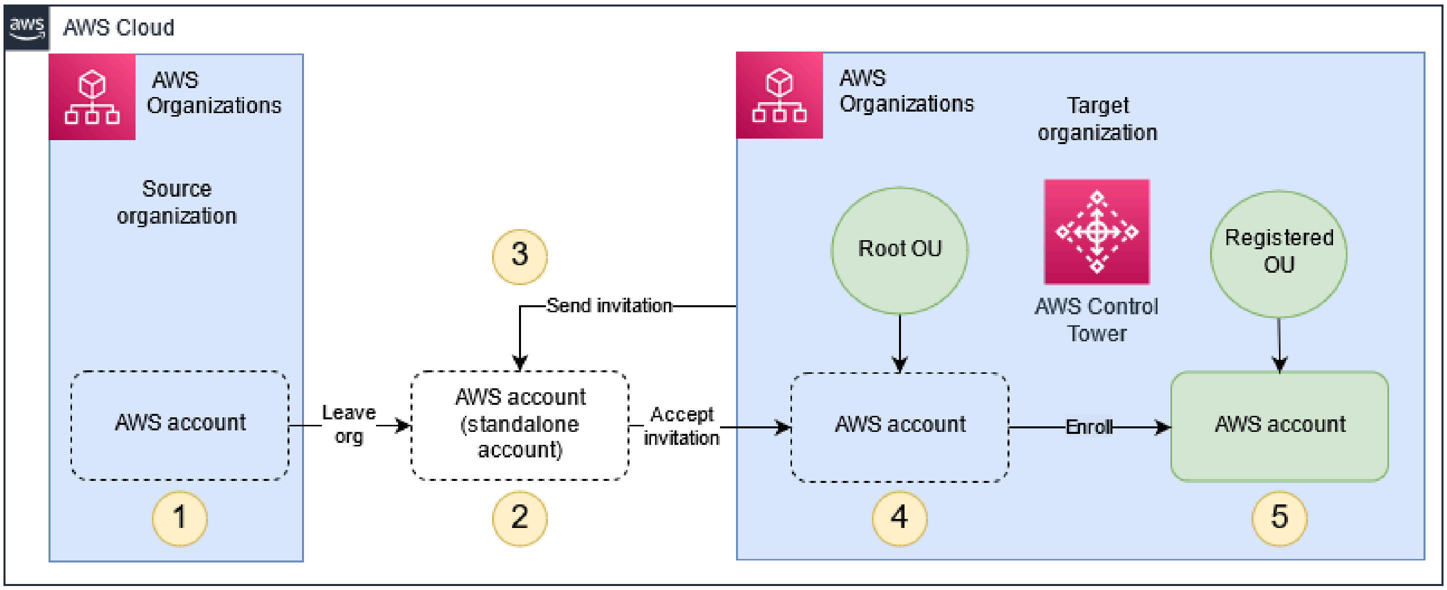
- The source management account in AWS Organizations must be different from the target management account in AWS Control Tower.

Product versions

- AWS Control Tower version 2.3 (February 2020) or later (see release notes)

Architecture

The following diagram illustrates the migration process and reference architecture. This pattern migrates the AWS account from the source organization to a target organization that is governed by AWS Control Tower.



The enrollment process consists of these steps:

1. The account leaves the source organization in AWS Organizations.
2. The account becomes a standalone account. This means that it doesn't belong to any organization, so governance and billing are managed independently by account administrators.
3. The target organization sends an invitation for the account to join the organization.
4. The standalone account accepts the invitation and becomes a member of the target organization.
5. The account is enrolled in AWS Control Tower and moved to a registered organizational unit (OU). (We recommend that you check the AWS Control Tower dashboard to confirm the enrollment.) At this point, all guardrails that are enabled in the registered OU take effect.

Tools

AWS services

- AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into a single entity (an *organization*) that you create and centrally manage.
- AWS Control Tower integrates the capabilities of other services, including AWS Organizations, AWS IAM Identity Center (successor to AWS Single Sign-On), and AWS Service Catalog, to help you enforce and manage governance rules for security, operations, and compliance at scale across all your organizations and accounts in the AWS Cloud.

Epics

Remove the member account from the source organization

Task	Description	Skills required
Verify that the member account can run as a stand-alone account.	Confirm that the member account that will leave the source organization has the information that is required for it to operate as a standalone account. For example, if the member account doesn't have billing information, it can't operate as a standalone account, because AWS uses the payment information to charge for any billable AWS activity that occurs while the account isn't attached to an organization. Typically, if you created the member account by using the AWS Organizations console,	Account administrator

Task	Description	Skills required
	API, or AWS Command Line Interface (CLI) commands, the information required of standalone accounts isn't automatically collected. To add this information, sign in to the account, and specify a support plan, contact information, and a payment method. For more information about what you need to know before removing an account from an organization, see [Before removing an account from organization](#) in the AWS Organizations documenta tion.	

Task	Description	Skills required
Remove the member account from its source organization.	Follow the instructions in the AWS Organizations documentation to remove a member account from an organization. You can sign in to the organizat ion's management account and remove the member account, or sign in to the member account and leave the organization. If you don't have administr ator-level credentials to remove or leave the account, ask for assistance from your organization's administrator. If the member account is missing a support plan, contact information, or payment information, you will be prompted to provide and verify that information. When you leave the organizat ion, you are redirected to the **Getting Started** page of the AWS Organizations console, where you can view invitatio ns for your account to join other organizations.	Management account administrator or account administrator

Task	Description	Skills required
	Important At this point, your account is a standalone account. If you are running workloads that aren't covered by AWS Free Tier, you will be charged according to the payment and billing information you provided for the account.	
Verify that the member account is no longer part of the source organization.	In the AWS Organizations console, you should no longer see the **Leave organization** button. Instead. you should see pending invitations, if any, from other organizations.	Account administrator

Task	Description	Skills required
Remove the IAM roles that grant access to your account from the organization you left.	When you remove the account from the source organization, AWS Identity and Access Management (IAM) roles created by AWS Organizations or by administrators aren't automatically deleted. To terminate access from the source organization's management account, you must manually delete the IAM roles. For more information, see [Deleting roles or instance profiles](#) in the IAM documentation. When a member account leaves an organization, all tags that were attached to the account are deleted. Standalone accounts do not support tags.	Account administrator

Invite the account to join the new organization with AWS Control Tower

Task	Description	Skills required
Sign in to AWS Control Tower.	Sign in to the AWS Control Tower console as an administrator. Currently, there is no direct way to move an AWS account from a source organization to an organization in an	AWS Control Tower administrator

Task	Description	Skills required
	OU that's governed by AWS Control Tower. However, you can extend AWS Control Tower governance to an existing AWS account when you enroll it into an OU that's already governed by AWS Control Tower. That's why you have to log in to AWS Control Tower for this step.	

Task	Description	Skills required
Invite the member account.	1. Sign in to the AWS Organizations console, and navigate to the **AWS Accounts** page. 2. On the **Add an AWS account** page, choose **Invite an existing AWS account**. 3. Complete the account information, including the 12-digit account number (without dashes) and the optional description and tags, and then choose **Send invitation**. **Important** Verify that no applications or network connectivity will be affected by the account transfer. This action sends an invitation email with a link to the member account. When the account administrator follows the link and accepts the invitation, the member account appears in the **AWS accounts** page. For more information, see [Inviting an	AWS Control Tower administrator

Task	Description	Skills required
	AWS account to join your organization in the AWS Organizations documentation.	
Test applications and connectivity.	When the member account has been registered into the new organization, it appears in the OU within a root. It also appears in the AWS Control Tower console, flagged as not enrolled in accounts, because it hasn't yet been enrolled in the AWS Control Tower registered OU. Verify the following: • Check the AWS Control Tower dashboard to see if there are any guardrail violations. • Check network connectivity (VPN or AWS Direct Connect) to make sure it wasn't affected by the transfer. • (Application owners) Test the applications that are associated with this account to verify that they run as expected, and that dependencies weren't affected by the account transfer.	AWS Control Tower administrator, Member account administrator, Application owners

Prepare the account for enrollment

Task	Description	Skills required
Review guardrails and fix any violations.	Review the guardrails that are defined in the target OU, especially the preventive guardrails, and fix any violations. A number of [mandatory, preventive guardrails]() are enabled by default when you set up your AWS Control Tower landing zone. These can't be disabled. You must review these mandatory guardrails and fix the member account (manually or by using a script) before you enroll the account. **Note** Preventive guardrails keep AWS Control Tower registered accounts compliant and prevent policy violations. Any violation of preventive guardrails might affect enrollment. Detective guardrail violations appear in the AWS Control Tower dashboard	AWS Control Tower administrator, Member account administrator

Task	Description	Skills required
	, if detected, after successful enrollment. They do not affect the enrollment process. For more information, see [Guardrails in AWS Control Tower](#) in the AWS documentation.	
Check for connectivity issues after fixing guardrail violation s.	In some cases, you might have to close specific ports or disable services to fix guardrail violations. Make sure that applications that use those ports and services are remediated before you enroll the account.	Application owner

Enroll the account into AWS Control Tower

Task	Description	Skills required
Sign in to the AWS Control Tower console.	Use sign-in credentials that have administrative permissio ns for AWS Control Tower. Do not use the root user (management account) credentials to enroll an AWS Organizations account. This will display an error message.	AWS Control Tower administr ator
Enroll the account.	1. From the Account Factory page in AWS Control	AWS Control Tower administr ator

Task	Description	Skills required
	Tower, choose **Enroll account**. 2. Fill in the details, including the email address associated with the account you want to enroll, the display name that will appear in AWS Control Tower, the IAM Identity Center email address, the first and last name of the account owner, and the OU in which you would like to enroll the account. The IAM Identity Center email address is your preferred user email address. You can use the same email address as the account email. 3. Choose **Enroll account**. For more information, see [Enroll an existing account](#) in the AWS Control Tower documentation.	

Verify the account after enrollment

Task	Description	Skills required
Verify the account.	From AWS Control Tower, choose **Accounts**. The account that you just enrolled has an initial state of **Enrolling**.	AWS Control Tower administrator, Member account administrator

Task	Description	Skills required
	When enrollment is complete, its state changes to **Enrolled**.	
Check for guardrail violations.	Guardrails defined in the OU will automatically apply to the enrolled member account. Monitor the AWS Control Tower dashboard for violations and fix them accordingly. For more information, see [Guardrails in AWS Control Tower](#) in the AWS documentation.	AWS Control Tower administrator, Member account administrator

Troubleshooting

Issue	Solution
You receive the error message: **An unknown error occurred. Try again later, or contact AWS Support.**	This error occurs when you use root user credentials (management account) in AWS Control Tower to enroll a new account. AWS Service Catalog can't map the Account Factory Portfolio or product to the root user, which results in the error message. To remediate this error, use non-root, full-access user (administrator) credentials to enroll the new account. For more information about how to assign administrative access to an administrative user, see [Getting started](#) in the AWS IAM Identity Center (successor to AWS Single Sign-On) documentation.

Issue	Solution
The AWS Control Tower **Activities** page displays a **Get Catastrophic Drift** action.	This action reflects a drift check of the service and does not indicate any issues with the AWS Control Tower setup. No action is required.

Related resources

Documentation

- AWS Organizations terminology and concepts (AWS Organizations documentation)
- What is AWS Control Tower? (AWS Control Tower documentation)
- Removing a member account from your organization (AWS Organizations documentation)
- Creating an administrator account in AWS Control Tower (AWS Control Tower documentation)

Tutorials and videos

- AWS Control Tower Workshop (self-paced workshop)
- What is AWS Control Tower? (video)
- Provisioning Users in AWS Control Tower (video)
- Enable AWS Control Tower for Existing Organization (video)

Monitor SAP RHEL Pacemaker clusters by using AWS services

Harsh Thoria, Randy Germann, and RAVEENDRA Voore, Amazon Web Services

Summary

This pattern outlines the steps for monitoring and configuring alerts for a Red Hat Enterprise Linux (RHEL) Pacemaker cluster for SAP applications and SAP HANA database services by using Amazon CloudWatch and Amazon Simple Notification Service (Amazon SNS).

The configuration enables you to monitor SAP SCS or ASCS, Enqueue Replication Server (ERS), and SAP HANA cluster resources when they are in a "stopped" state with the help of CloudWatch log streams, metric filters, and alarms. Amazon SNS sends an email to the infrastructure or SAP Basis team about the stopped cluster status.

You can create the AWS resources for this pattern by using AWS CloudFormation scripts or the AWS service consoles. This pattern assumes that you're using the consoles; it doesn't provide CloudFormation scripts or cover infrastructure deployment for CloudWatch and Amazon SNS. Pacemaker commands are used to set the cluster alerting configuration.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- Amazon SNS set up to send email or mobile notifications.
- An SAP ASCS/ERS for ABAP or SCS/ERS for Java, and SAP HANA Database RHEL Pacemaker cluster. For instructions, see the following:
  - SAP HANA cluster setup
  - SAP Netweaver ABAP/Java cluster setup

Limitations

- This solution currently works for RHEL version 7.3 and later Pacemaker-based clusters. It hasn’t been tested on SUSE operating systems.

Product versions

- RHEL 7.3 and later

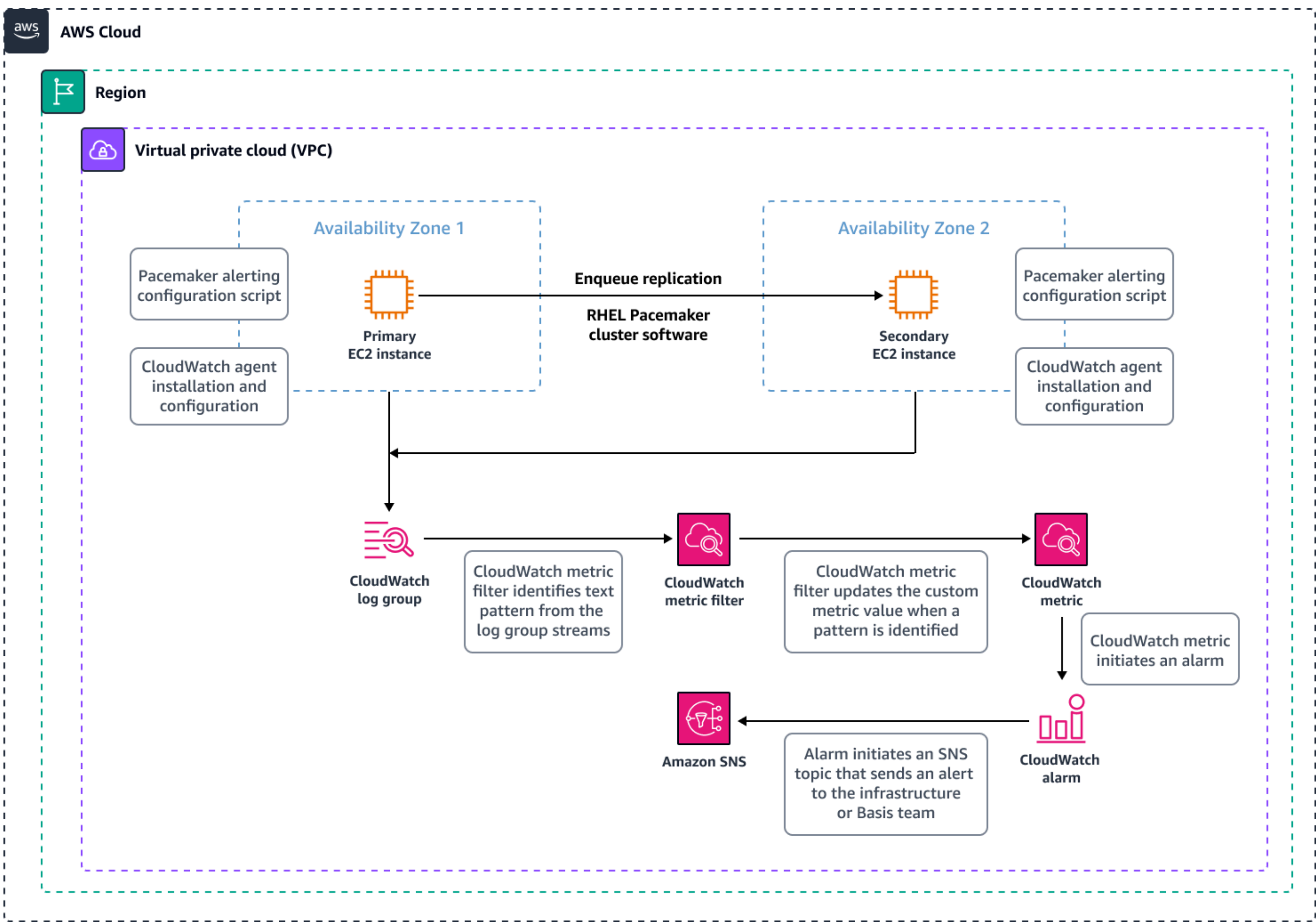
Architecture

Target technology stack

- RHEL Pacemaker alert event-driven agent
- Amazon Elastic Compute Cloud (Amazon EC2)
- CloudWatch alarm
- CloudWatch log group and metric filter
- Amazon SNS

Target architecture

The following diagram illustrates the components and workflows for this solution.



Automation and scale

- You can automate the creation of AWS resources by using CloudFormation scripts. You can also use additional metric filters to scale and cover multiple clusters.

Tools

AWS services

- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

Tools

- CloudWatch agent (unified) is a tool that collects system-level metrics, logs, and traces from EC2 instances, and retrieves custom metrics from your applications.
- Pacemaker alert agent (for RHEL 7.3 and later) is a tool that initiates an action when there's a change, such as when a resource stops or restarts, in a Pacemaker cluster.

Best practices

- For best practices for using SAP workloads on AWS, see the SAP Lens for the AWS Well-Architected Framework.
- Consider the costs involved in setting up CloudWatch monitoring for SAP HANA clusters. For more information, see the CloudWatch documentation.
- Consider using a pager or ticketing mechanism for Amazon SNS alerts.
- Always check for RHEL high availability (HA) versions of the RPM package for **pcs**, Pacemaker, and the AWS fencing agent.

Epics

Set up Amazon SNS

Task	Description	Skills required
Create an SNS topic.	1. Sign in to the AWS Management Console and open the Amazon SNS console at https://console.aws.amazon.com/sns/v3/home. 2. On the Amazon SNS dashboard, under **Common actions**, choose **Create Topic**. 3. In the **Create new topic** dialog box, for **Type**, choose **Standard**. 4. For **Topic name**, enter a name for the topic (for example, `my-topic`). 5. Choose **Create topic**. This creates an SNS topic with a resource policy that lets you publish notificat ions. 6. Copy the **Topic ARN** (for example, `arn:aws:sns:us-east-1:111122223333:my-topic`). You will use this ARN in a later step.	AWS administrator
Modify the access policy for the SNS topic.	1. On the Amazon SNS console, in the navigation	AWS systems administrator

Task	Description	Skills required
	pane, choose **Topics**, and then choose the topic you created. 2. Choose **Edit** and go to the **Access policy** section. 3. Make sure that the access policy includes CloudWatch as one of the service principals that are allowed to publish to this topic. For example: `{ "Sid": "Allow AWS CloudWatch to Publish to this SNS topic", "Effect": "Allow", "Principal": { "Service": [ "cloudwatch.amazonaws.com" ] }, "Action": "SNS:Publish", "Resource": "arn:aws:sns:us-east-1:111122223333:my-topic" }` 4. Choose **Save changes**.	

Task	Description	Skills required
Subscribe to the SNS topic.	1. On the Amazon SNS console, in the navigatio n pane, choose **Subscript ions**, **Create subscription**. 2. For **Topic ARN**, paste the ARN that you created in the first task. 3. For **Protocol**, choose **Email**. 4. For **Endpoint**, enter an email address for the person or team that is responsible for the SAP Pacemaker cluster and should receive notificat ions. For example, this can be the email address for the SAP Basis or infrastru cture team's distribution list. 5. Choose **Create subscript ion**. 6. From your email applicati on, open the message from AWS Notifications and confirm your subscription. Your web browser displays a confirmation response from Amazon SNS.	AWS systems administrator

Confirm the setup of the cluster

Task	Description	Skills required
Check cluster status.	Use the **pcs status** command to confirm that the resources are online.	SAP Basis administrator

Configure Pacemaker alerts

Task	Description	Skills required

Configure the Pacemaker alert agent on the primary cluster instance.	Log in to the EC2 instance in the pimary cluster and run the following commands:
	`install --mode=0755 /usr/share/pacemaker/alerts/alert_file.sh.sample`
	`touch /var/lib/pacemaker/alert_file.sh`
	`touch /var/log/pcmk_alert_file.log`
	`chown hacluster:haclient /var/log/pcmk_alert_file.log`
	`chmod 600 /var/log/pcmk_alert_file.log`
	`pcs alert create id=alert_file description="Log events to a file." path=/var/lib/pacemaker/alert_file.sh`
	`pcs alert recipient add alert_file id=my-alert_logfile value=/va`
		SAP Basis administrator

Task	Description	Skills required
	`r/log/pcmk_alert_f` `ile.log`	

Configure the Pacemaker alert agent on the secondary cluster instance.	Log in to the secondary cluster EC2 instance in the secondary cluster and run the following commands:
	
	`install --mode=0755 /`
	`usr/share/pacemak`
	`er/alerts/alert_fi`
	`le.sh.sample`
	`touch /var/lib/`
	`pacemaker/alert_fi`
	`le.sh`
	`touch /var/log/`
	`pcmk_alert_file.log`
	`chown hacluster`
	`:haclient /var/log/`
	`pcmk_alert_file.log`
	`chmod 600 /var/log/`
	`pcmk_alert_file.log`
		SAP Basis administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Confirm that the RHEL alert resource was created.</td><td>Use the following command to confirm that the alert resource was created:

<pre>pcs alert</pre>

The output of the command will look like this:

<pre>[root@xxxxxxx ~]# pcs
 alert
Alerts:
 Alert: alert_file
 (path=/var/lib/pac
emaker/alert_file.sh)
  Description: Log
 events to a file.
  Recipients:
   Recipient: my-
alert_logfile (value=/
var/log/pcmk_alert_
file.log)</pre></td><td>SAP Basis administrator</td></tr>
</table>

Configure the CloudWatch agent

Task	Description	Skills required
Install the CloudWatch agent.	There are several ways to install the CloudWatch agent on an EC2 instance. To use the command line: 1. Download the CloudWatch agent package:	AWS systems administrator

Task	Description	Skills required

	`wget https://s3.<region>.amazonaws.com/amazoncloudwatch-agent-region/redhat/amd64/latest/amazon-cloudwatch-agent.rpm`
	
	where `<region>` is the AWS Region where the EC2 instance is located (for example, `us-west-2`).
	
	2. Optional) Verify the package signature. For instructions, see [Verifying the signature of the CloudWatch agent package](#) in the CloudWatch documentation.
	
	3. Install the package on the first instance:
	
	`sudo rpm -U ./amazon-cloudwatch-agent.rpm`
	
	4. Repeat for the secondary instance.
	
	For more information, see the [CloudWatch documentation](#).	

Task	Description	Skills required
Attach an IAM role to the EC2 instance.	To enable the CloudWatch agent to send data from the instances, you must attach the IAM **CloudWatchAgentServerRole** role to each instance. Or, you can add a policy for the CloudWatch agent to your existing IAM role. For more information, see the [CloudWatch documentation]().	AWS administrator

Task	Description	Skills required
Configure the CloudWatch agent to monitor the Pacemaker alert agent log file on the primary cluster instance.	1. Configure the primary cluster instance by running the command: `sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-config-wizard` 2. Choose **1** for Linux, and then select the options for your monitoring strategy. 3. For the question **"Do you want to monitor any log files,"** choose **Yes** and provide the path of the Pacemaker log file from the **pcs alert** command. In our case, it is `var/log/pcmk_alert_file.log` . 4. Provide the name of the log group and log stream. If you don't specify a log stream, the AWS instance ID is used as the default. 5. Repeat steps 1-4 for the secondary cluster instance.	AWS administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Start the CloudWatch agent on the primary and secondary cluster instances.</td><td>To start the agent, run the following command on the EC2 instances in the primary and secondary clusters:
<pre><code>sudo /opt/aws/amazon-cloudwatch-agent/bin/amazon-cloudwatch-agent-ctl -a fetch-config -m
ec2 -s -c file:/opt/aws/amazon-cloudwatch-agent/bin/config.json</code></pre></td><td>AWS administrator</td></tr>
</table>

Set up CloudWatch resources

Task	Description	Skills required
Set up CloudWatch log groups.	1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/ 2. In the navigation pane, choose **Log groups, Create log group**. 3. Enter a name for the log group,and then choose **Create log group**. The CloudWatch agent will transfer the Pacemaker alert file to the CloudWatch log group as a log stream.	AWS administrator

Task	Description	Skills required
Set up CloudWatch metric filters.	Metric filters help you search for a pattern such as `stop <cluster-resource-name>` in the CloudWatch log streams. When this pattern is identified, the metric filter updates a custom metric. 1. On the CloudWatch console, in the navigation pane, choose **Log groups**. 2. Choose the name of the log group you created in the previous task. 3. Choose **Actions**, **Create metric filter**. 4. For **Filter pattern**, enter the filter pattern to use, such as `stop ABC_scs`, to match the stop event for an SAP SCS cluster resource named `ABC_scs`. For more information, see [Filter pattern syntax](#) in the CloudWatch documentation. 5. (Optional) To test your filter pattern, under **Test Pattern**, enter one or more log events to use to test the pattern. Each log event must be specified on a	AWS administrator, SAP Basis administrator

Task	Description	Skills required
	separate line, because line breaks are used to separate log events in the **Log event messages** box. 6. Choose **Next**, and then enter a name for the filter. 7. Under **Metric details**, for **Metric namespace**, enter a name for the CloudWatch namespace where the metric will be published (for example, `sapcluster_monitoring`). If this namespace doesn't already exist, select **Create new**. 8. For **Metric name**, enter a name for the new metric (for example, `sapcluster_<sid>` , where `<sid>` is the SAP system identification name). 9. For **Metric value**, enter **1**. Alternatively, you can enter a token such as `$size`. This increments the metric by the value of the number in the `size` field for every log event that contains a `size` field. 10. For **Default value**, enter **0**. 11. Choose **Create metric filter**.	

Task	Description	Skills required
	When the metric filter identifies the pattern in step 4, it updates the value of the CloudWatch custom metric `sapcluster_abc` to **1**. The CloudWatch alarm `SAP-Cluster-QA1-ABC` monitors the metric `sapcluster_abc` and sends out an SNS notification when the value of the metric changes to **1**. This indicates that the cluster resource has stopped and action needs to be taken.	

Task	Description	Skills required

Set up a CloudWatch metric alarm for the SAP ASCS/SCS and ERS metric.	To create an alarm based on a single metric:
	
	1. On the CloudWatch console, in the navigation pane, choose **Alarms**, **All alarms**.
	2. Choose **Create alarm**.
	3. Choose **Select Metric**.
	4. Search for the custom metric `sapcluster_monitoring` that was created in the previous task.
	5. Choose the metric name for SAP SCS (for example, `sapcluster_<abc>`), which was also created in the previous task.
	6. On the **Graphed metrics** tab, set the following:
	  • For **Statistic**, choose **Maximum**.
	  • For **Period**, choose **1 minute**.
	  • For **Threshold type**, choose **Static** and set the threshold for `sapcluster_<sid>` to a value that's greater than or equal to 1.
	7. Choose **Next**.
		AWS administrator

Task	Description	Skills required
	8. For **Notification**, select the SNS topic you created in the first epic. 9. For **Name** and **Description**, provide the alarm name and a brief description, and then choose **Next**. 10.Choose **Create Alarm**.	
Set up a CloudWatch metric alarm for the SAP HANA metric.	Repeat the steps for setting up a CloudWatch metric alarm from the previous task, with these changes: • For step 5, choose the metric name for SAP HANA (for example, `sapcluster_db_<abc>`). • For step 6, set the threshold for `sapcluster_<sid>` to a value that's greater than 0.	AWS administrator

Related resources

- Triggering Scripts for Cluster Events (RHEL documentation)
- Create the CloudWatch agent configuration file with the wizard (CloudWatch documentation)
- Installing and running the CloudWatch agent on your servers (CloudWatch documentation)
- Create a CloudWatch alarm based on a static threshold (CloudWatch documentation)
- Manual deployment of SAP HANA on AWS with high availability clusters (SAP documentation on the AWS website)
- SAP NetWeaver guides (SAP documentation on the AWS website)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Monitor application activity by using CloudWatch Logs Insights

Ram Kandaswamy, Amazon Web Services

Summary

This pattern provides a solution for automatically detecting and alerting on application exceptions by using Amazon CloudWatch Logs Insights. By implementing automated log analysis and alerting, you can quickly identify and respond to application issues in your production environment.

Logs play a crucial role in monitoring system behavior, identifying issues, and ensuring optimal performance. During a migration process, log files are invaluable for validating the system's functioning in the new environment, detecting compatibility problems, and identifying any unexpected behaviors. Issues could be related to operations or security. For security-related issues, enabling the detection of unauthorized access attempts or suspicious activities early is essential for maintaining security and regulatory compliance. This capability is especially important when dealing with sensitive data or critical systems.

This pattern is particularly valuable for teams that need to maintain high application availability and quickly respond to production issues. It’s relevant across various industries and use cases. For example, in telecommunications, it enables quick identification of network misconfigurations or outages and detection of suboptimal routing paths to pinpoint potential congestion. In the Internet of Things (IoT) domain, Greengrass components can publish logs to CloudWatch, allowing this technique to extract relevant log details and present them in a comprehensive dashboard.

Prerequisites and limitations

Prerequisites

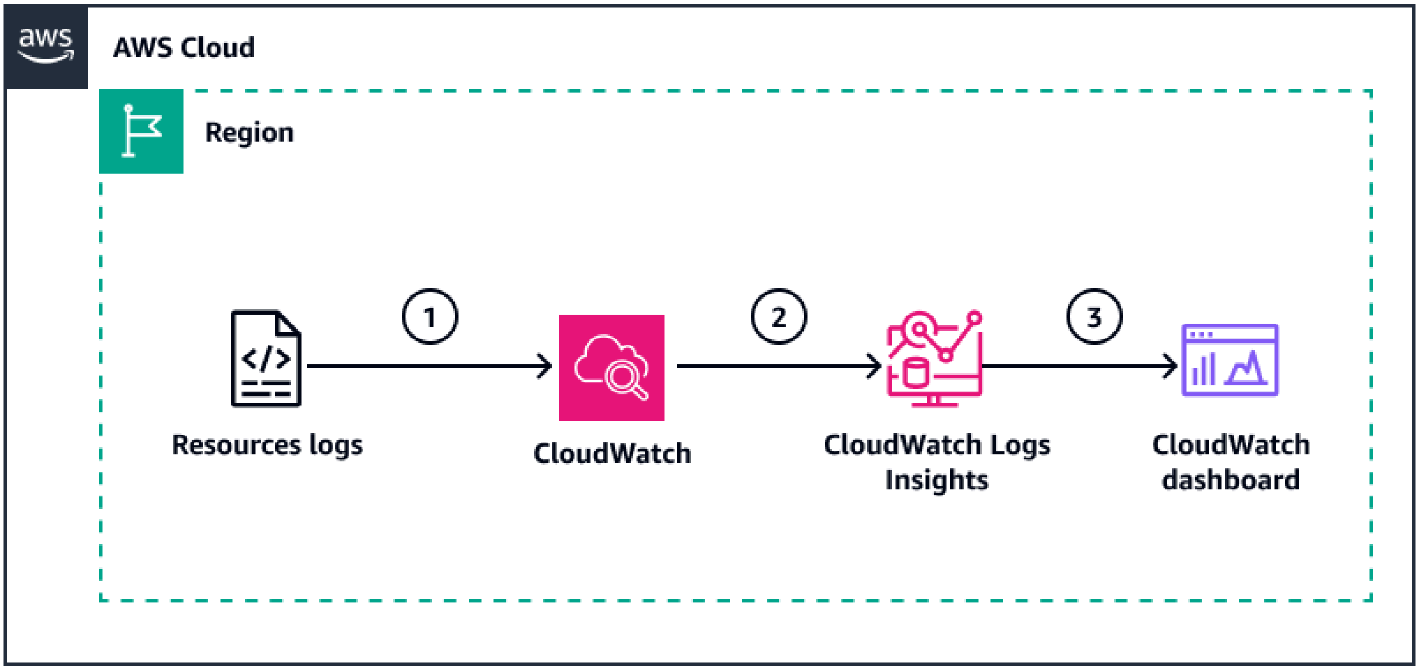
- A production application deployed in active AWS account
- Basic understanding of the production application's logging format and exception patterns
- Application logs configured to stream to Amazon CloudWatch Logs

Limitations

- Some AWS services aren’t available in all AWS Regions. For Region availability, see [AWS Services by Region]. For specific endpoints, see [Service endpoints and quotas], and choose the link for the service.

Architecture

The following diagram shows how CloudWatch Logs Insights evaluates resource logs and sends a relevant data visualization to a CloudWatch dashboard.



The diagram shows the following workflow:

1. The resources publish logs to CloudWatch Logs. Resources can include AWS resources such as Amazon Elastic Compute Cloud (Amazon EC2) instances or Amazon Simple Storage Service (Amazon S3) buckets. Another example includes on-premises systems with CloudWatch Agent installed that can publish logs to CloudWatch.
2. CloudWatch Logs Insights filters for the relevant pattern string. Examples of search pattern strings include "error", "exception", or a specific regular expression.
3. Typically, the production support team or developers add the pattern visualization to the CloudWatch dashboard.

Automation and scale

Developers can automate this pattern’s solution by using the AWS Cloud Development Kit (AWS CDK), AWS CloudFormation, or AWS SDKs to handle multiple string patterns. Teams can incorporate this automation into their continuous integration and deployment (CI/CD) DevOps processes.

Tools

AWS services

- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.

Best practices

- Define and configure log groups to analyze relevant log data.
- Use field explorers to understand the structure and fields available in your log data.
- Write efficient queries by using CloudWatch Logs Insights query syntax.
- Adapt sample queries to your specific requirements for quicker analysis.
- Monitor queries to identify potential performance issues or bottlenecks.
- Set query limits to avoid excessive costs or resource consumption.
- Save queries for future use to save time and ensure consistent analysis.
- Apply appropriate IAM policies to control access to CloudWatch Logs Insights and log groups. Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.
- Enable log data encryption using AWS KMS for sensitive log data.

Epics

Create log group and configure logs to view in dashboard.

Task	Description	Skills required
Configure IAM permissions.	To configure IAM permissions, do the following:	AWS administrator, AWS DevOps, AWS systems administrator, Cloud

Task	Description	Skills required
	1. Identify the IAM roles used by your application for writing logs and by users or services that will create dashboards, queries, and alarms. 2. For the application role, add the following API actions and resource to a policy: `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents" ], "Resource": [ "arn:aws:logs:*:*:*" ] } ] }`	administrator, Cloud architect , DevOps engineer

Task	Description	Skills required
	3. For managing CloudWatch resources, add the following to a policy: ```{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["cloudwatch:PutDashboard", "cloudwatch:GetDashboard", "logs:StartQuery", "logs:StopQuery", "logs:GetQueryResults", "cloudwatch:PutMetricAlarm"], "Resource": "*" }] }``` For information about how to create IAM policies or to add permissions to	

Task	Description	Skills required
	existing policies, see Define custom IAM permissions with customer managed policies and Edit IAM policies in the *IAM User Guide*. For more information, see Identity and access management for Amazon CloudWatch Logs and CloudWatch Logs permissions reference in the *Amazon CloudWatch Logs User Guide*.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Create a log group.</td><td>To create a log group, use any of the following options:
Create or open a CloudFormation template file (YAML or JSON format) with a <code>.yaml</code> or <code>.json</code> extension. (The following code uses YAML format.) Add the following resource definition to the Resources section of your template:<pre><code>MyLogGroup:
  Type: AWS::Logs
::LogGroup
  Properties:
    LogGroupName: my-
log-group</code></pre>For more information, see Quick Start: Use AWS CloudFormation to get started with CloudWatch Logs in the Amazon CloudWatch Logs User Guide.
Use the CloudWatch console, as described in Create a log group in CloudWatch Logs in the Amazon CloudWatch Logs User Guide.</td><td>AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect , DevOps engineer</td></tr>
</table>

Task	Description	Skills required
	• Use the CreateLogGroup operation in the CloudWatch Logs API.	
Generate a CloudWatch Logs Insights query.	To create and save a CloudWatch Logs Insights query: 1. Open the CloudWatch console. 2. Navigate to **Logs Insights**. 3. Create your query by using one of the following methods: • Start with a sample query from the **Queries** section. • Write a custom query. • Modify the following example query: `fields @timestamp, @message | filter @message like /(?i)exception/|stats count() by bin(30s)` This query checks the log file, extracts the timestamp and message content, filters for the word "exception" (case-insensitive), and counts occurrences in 30-second intervals.	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect , DevOps engineer

Task	Description	Skills required
Create visualization in a CloudWatch dashboard.	To use a CloudWatch dashboard to create a visualization, do the following : 1. Open the CloudWatch console, and create a CloudWatch dashboard (if you don't have one already) or open an existing dashboard. 2. Add your exception monitoring visualization. Bar, line, and pie charts are available, depending on your business need. For more information, see Using widgets on CloudWatch dashboards in the *Amazon CloudWatch Logs User Guide*. For real-time data visualizations, you can customize a widget to meet your requirements. For more information about dashboard options and capabilities, see Using Amazon CloudWatch dashboards and Creating flexible CloudWatch dashboards with dashboard variables in the *Amazon CloudWatch Logs User Guide*.	AWS administrator, AWS DevOps, AWS systems administrator, Cloud administrator, Cloud architect , DevOps engineer

Troubleshooting

Issue	Solution
Unable to see query results or query seems broken	Start with a working query that was modified from a sample query. Perform small incremental changes to parts of the query (such as a filter or field), and take advantage of the CloudWatch Logs query generator feature.
Log groups not creating log streams	In the IAM policy, make sure that the resource for the CreateLogStream and the CreateLogGroup operations is provided with a wildcard character (`*`) value. The `create` operation will not succeed without this wildcard permission.

Related resources

- Analyzing log data with CloudWatch Logs Insights
- Amazon CloudWatch FAQs
- Creating flexible CloudWatch dashboards with dashboard variables
- Get started with Logs Insights QL: Query tutorials
- Use natural language to generate and update CloudWatch Logs Insights queries
- Use PutDashboard with an AWS SDK or CLI
- Working with log groups and log streams

Monitor use of a shared Amazon Machine Image across multiple AWS accounts

Naveen Suthar and Sandeep Gawande, Amazon Web Services

Summary

[Amazon Machine Images (AMIs)] are used to create Amazon Elastic Compute Cloud (Amazon EC2) instances in your Amazon Web Services (AWS) environment. You can create AMIs in a separate, centralized AWS account, which is called a *creator account* in this pattern. You can then share the AMI across multiple AWS accounts that are in the same AWS Region, which are called *consumer accounts* in this pattern. Managing AMIs from a single account provides scalability and simplifies governance. In the consumer accounts, you can reference the shared AMI in Amazon EC2 Auto Scaling launch templates and Amazon Elastic Kubernetes Service (Amazon EKS) node groups.

When a shared AMI is deprecated, deregistered, or unshared, AWS services that refer to the AMI in the consumer accounts cannot use this AMI to launch new instances. Any auto scaling event or relaunch of the same instance fails. This can lead to issues in the production environment, such as application downtime or performance degradation. When AMI sharing and usage events occur in multiple AWS accounts, it can be difficult to monitor this activity.

This pattern helps you monitor shared AMI usage and status across accounts in the same Region. It uses serverless AWS services, such as Amazon EventBridge, Amazon DynamoDB, AWS Lambda, and Amazon Simple Email Service (Amazon SES). You provision the infrastructure as code (IaC) by using HashiCorp Terraform. This solution provides alerts when a service in a consumer account references a deregistered or unshared AMI.

Prerequisites and limitations

Prerequisites

- Two or more active AWS accounts: one creator account and one or more consumer accounts
- One or more AMIs that are shared from the creator account to a consumer account
- Terraform CLI, installed (Terraform documentation)
- Terraform AWS Provider, configured (Terraform documentation)
- (Optional, but recommended) Terraform backend, configured (Terraform documentation)
- Git, installed

Limitations

- This pattern monitors AMIs that have been shared to specific accounts by using the account ID. This pattern does not monitor AMIs that have been shared to an organization by using the organization ID.
- AMIs can only be shared to accounts that are within the same AWS Region. This pattern monitors AMIs within a single, target Region. To monitor use of AMIs in multiple Regions, you deploy this solution in each Region.
- This pattern doesn't monitor any AMIs that were shared before this solution was deployed. If you want to monitor previously shared AMIs, you can unshare the AMI and then reshare it with the consumer accounts.

Product versions

- Terraform version 1.2.0 or later
- Terraform AWS Provider version 4.20 or later

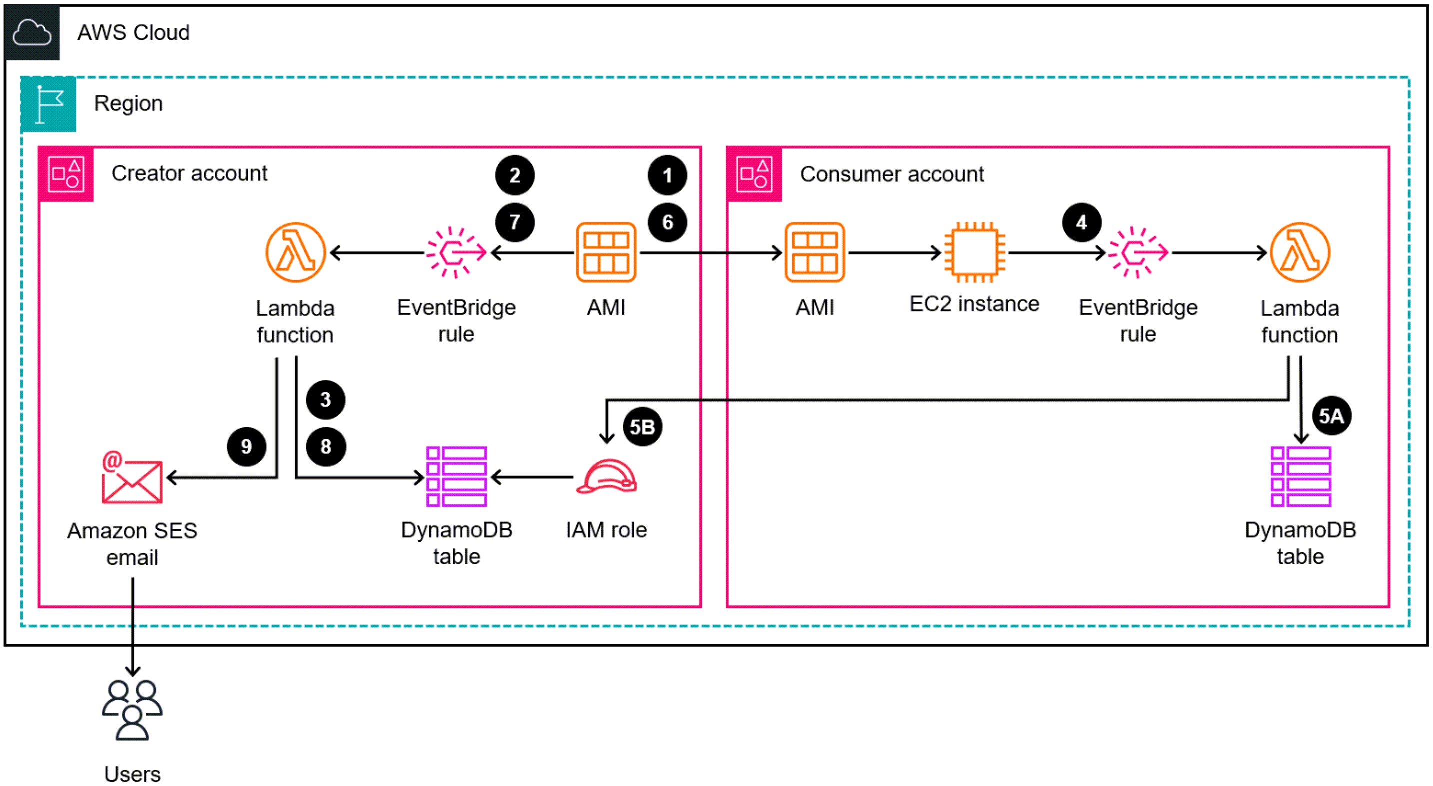
Architecture

Target technology stack

The following resources are provisioned as IaC through Terraform:

- Amazon DynamoDB tables
- Amazon EventBridge rules
- AWS Identity and Access Management (IAM) role
- AWS Lambda functions
- Amazon SES

Target architecture



The diagram shows the following workflow:

1. An AMI in the creator account is shared with a consumer account in the same AWS Region.
2. When the AMI is shared, an EventBridge rule in the creator account captures the `ModifyImageAttribute` event and initiates a Lambda function in the creator account.
3. The Lambda function stores data related to the AMI in a DynamoDB table in the creator account.
4. When an AWS service in the consumer account uses the shared AMI to launch an Amazon EC2 instance or when the shared AMI is associated with a launch template, an EventBridge rule in the consumer account captures use of the shared AMI.
5. The EventBridge rule initiates a Lambda function in the consumer account. The Lambda function does the following:
    a. The Lambda function updates the AMI-related data in a DynamoDB table in the consumer account.
    b. The Lambda function assumes an IAM role in the creator account and updates the Lambda table in the creator account. In the `Mapping` table, it creates an item that maps the instance ID or launch template ID to its respective AMI ID.
6. The AMI that is centrally managed in the creator account is deprecated, deregistered, or unshared.

7. The EventBridge rule in the creator account captures the `ModifyImageAttribute` or `DeregisterImage` event with the `remove` action and initiates the Lambda function.
8. The Lambda function checks the DynamoDB table to determine whether the AMI is used in any of the consumer accounts. If there are no instance IDs or launch template IDs associated with the AMI in the `Mapping` table, then the process is complete.
9. If any instance IDs or launch template IDs are associated with the AMI in the `Mapping` table, then the Lambda function uses Amazon SES to send an email notification to the configured subscribers.

Tools

AWS services

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Email Service (Amazon SES) helps you send and receive emails by using your own email addresses and domains.

Other tools

- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub cross-account-ami-monitoring-terraform-samples repository.

Best practices

- Follow the Best practices for working with AWS Lambda functions.
- Follow the Best practices for building AMIs.
- When creating the IAM role, follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the IAM documentation.
- Set up monitoring and alerting for the AWS Lambda functions. For more information, see Monitoring and troubleshooting Lambda functions.

Epics

Customize the Terraform configuration files

Task	Description	Skills required
Create the AWS CLI named profiles.	For the creator account and each consumer account, create an AWS Command Line Interface (AWS CLI) named profile. For instructions, see Set up the AWS CLI in the AWS Getting Started Resources Center.	DevOps engineer
Clone the repository.	Enter the following command. This clones the cross-account-ami-monitoring-terraform-samples repository from GitHub by using SSH. `git clone git@github.com:aws-samples/`	DevOps engineer

Task	Description	Skills required
	`cross-account-ami-monitoring-terraform-samples.git`	

Task	Description	Skills required
Update the provider.tf file.	1. Enter the following command to navigate into the `terraform` folder in the cloned repository. `cd cross-account-ami-monitoring/terraform` 2. Open the `provider.tf` file. 3. Update the Terraform AWS Provider configurations for the creator account and consumer account as follows: • For `alias`, enter a name for the provider configuration. • For `region`, enter the target AWS Region where you want to deploy this solution. • For `profile`, enter the AWS CLI named profile for accessing the account. 4. If you are configuring more than one consumer account, create a profile for each additional consumer account. 5. Save and close the `provider.tf` file.	DevOps engineer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>For more information about configuring the providers, see Multiple provider configurations in the Terraform documentation.</td><td></td></tr>
<tr><td>Update the terraform.tfvars file.</td><td>Open the <code>terraform.tfvars</code> file.
In the <code>account_email_mapping</code> parameter, configure alerts for the creator account and consumer account as follows:
For <code>account</code>, enter the account ID.
For <code>email</code>, enter the email address where you want to send alerts. You can enter only one email address for each account.
If you are configuring more than one consumer account, enter an account and email address for each additional consumer account.
Save and close the <code>terraform.tfvars</code> file.</td><td>DevOps engineer</td></tr>
</table>

Task	Description	Skills required
Update the main.tf file.	Complete these steps only if you are deploying this solution to more than one consumer account. If you are deploying this solution to only one consumer account, no modification of this file is necessary. 1. Open the `main.tf` file. 2. For each additional consumer account, create a new module that is based off the `consumer_account_A` module in the template. For each consumer account, for `provider`, the value should match the alias you entered in the `provider.tf` file. 3. Save and close the `main.tf` file.	DevOps engineer

Deploy the solution by using Terraform

Task	Description	Skills required
Deploy the solution.	In the Terraform CLI, enter the following commands to deploy the AWS resources in the creator and consumer accounts:	DevOps engineer

Task	Description	Skills required

	1. Enter the following command to initialize Terraform.
	`terraform init`
	2. Enter the following command to validate the Terraform configurations.
	`terraform validate`
	3. Enter the following command to create a Terraform execution plan.
	`terraform plan`
	4. Review the configuration changes in the Terraform plan and confirm that you want to implement these changes.
	5. Enter the following command to deploy the resources.
	`terraform apply`	

Task	Description	Skills required
Verify the email address identity.	When you deployed the Terraform plan, Terraform created an email address identity for each consumer account in Amazon SES. Before notifications can be sent to that email address, you must verify the email address. For instructions, see [Verifying an email address identity](#) in the Amazon SES documentation.	General AWS

Validate resource deployment

Task	Description	Skills required
Validate deployment in the creator account.	1. Sign in to the creator account. 2. In the navigation bar, confirm that are viewing the target AWS Region. If you're in a different Region, choose the name of the currently displayed Region, and then choose the target Region. 3. Open the [DynamoDB console](#). 4. In the navigation pane, choose **Tables**.	DevOps engineer

Task	Description	Skills required
	5. In the list of tables, validate that the `AmiShare` table is present. 6. Open the [Lambda console](https://console.aws.amazon.com/lambda/). 7. In the navigation pane, choose **Functions**. 8. In the list of functions , validate that the `ami-share` function is present. 9. Open the [IAM console](https://console.aws.amazon.com/iam/). 10. In the navigation pane, choose **Roles**. 11. In the list of roles, validate that the `external-ddb-role` role is present. 12. Open the [EventBridge console](https://console.aws.amazon.com/events/). 13. In the navigation pane, choose **Rules**. 14. In the list of rules, validate that the `modify_image_attribute_event` rule is present. 15. Open the [Amazon SES console](https://console.aws.amazon.com/ses/). 16. In the navigation pane, choose **Verified Identities**. 17. In the list of identities, validate that an email address identity has been	

Task	Description	Skills required
	registered and verified for each consumer account.	

Task	Description	Skills required
Validate deployment in the consumer account.	1. Sign in to the consumer account. 2. In the navigation bar, confirm that are viewing the target AWS Region. If you're in a different Region, choose the name of the currently displayed Region, and then choose the target Region. 3. Open the [DynamoDB console](https://console.aws.amazon.com/dynamodb/). 4. In the navigation pane, choose **Tables**. 5. In the list of tables, validate that the `Mapping` table is present. 6. Open the [Lambda console](https://console.aws.amazon.com/lambda/). 7. In the navigation pane, choose **Functions**. 8. In the list of functions , validate that the `ami-usage-function` and `ami-deregister-function` functions are present. 9. Open the [EventBridge console](https://console.aws.amazon.com/events/). 10. In the navigation pane, choose **Rules**. 11. In the list of rules, validate that the `ami_usage`	DevOps engineer

Task	Description	Skills required
	`_events` and `ami_dereg ister_events` rules are present.	

Validate monitoring

Task	Description	Skills required
Create an AMI in the creator account.	1. In the creator account, create a private AMI. For instructions, see Create an AMI from an Amazon EC2 instance. 2. Share the new AMI with one of the consumer accounts. For instructions, see Share an AMI with specific AWS accounts.	DevOps engineer
Use the AMI in the consumer account.	In the consumer account, use the shared AMI to create an Amazon EC2 instance or launch template. For instructi ons, see How do I launch an Amazon EC2 instance from a custom AMI (AWS re:Post Knowledge Center) or Create a launch template for an Auto Scaling group (Amazon EC2 Auto Scaling documentation).	DevOps engineer
Validate monitoring and alerting.	1. Sign in to the creator account.	DevOps engineer

Task	Description	Skills required
	2. Open the [Amazon EC2 console](). 3. In the navigation pane, choose **AMIs**. 4. Select the AMI in the list, and then choose **Actions**, **Edit AMI permissions**. 5. In the **Shared accounts** section, select the consumer account, and then choose **Remove selected**. 6. Choose **Save changes**. 7. Validate that the target email address you defined for the consumer account receives a notification that the sharing was canceled for the AMI.	

(Optional) Stop monitoring shared AMIs

Task	Description	Skills required
Delete the resources.	1. Enter the following command to remove the resources deployed by this pattern and stop monitorin g shared AMIs. `terraform destroy`	DevOps engineer

Task	Description	Skills required
	2. Confirm the `destroy` command by entering `yes`.	

Troubleshooting

Issue	Solution
I did not receive an email alert.	There could be multiple reasons why the Amazon SES email was not sent. Check the following: 1. In the Epics section, use the *Validate resource deployment* epic to confirm that the infrastructure was provisioned properly in all AWS accounts. 2. Validate the Lambda function events in Amazon CloudWatch Logs. For instructions, see Using the CloudWatch console in the Lambda documentation. Confirm that there are no permissions issues, such an explicit deny in any identity-based or resource-based policies. For more information, see Policy evaluation logic in the IAM documentation. 3. In Amazon SES, validate that the status of the email address identity is **Verified**. For more information, see Verifying an email address identity.

Related resources

AWS documentation

- Building Lambda functions with Python (Lambda documentation)

- Create an AMI (Amazon EC2 documentation)
- Share an AMI with specific AWS accounts (Amazon EC2 documentation)
- Deregister your AMI (Amazon EC2 documentation)

Terraform documentation

- Install Terraform
- Terraform Backend Configuration
- Terraform AWS Provider
- Terraform binary download

Set up alerts for programmatic account closures in AWS Organizations

Richard Milner-Watts, Debojit Bhadra, and Manav Yadav, Amazon Web Services

Summary

The CloseAccount API for AWS Organizations enables you to close member accounts within an organization programmatically, without having to log in to the account with root credentials. The RemoveAccountFromOrganization API pulls an account out from an organization in AWS Organizations, so it becomes a standalone account.

These APIs potentially increase the number of operators who can close or remove an AWS account. All users who have access to the organization through AWS Identity and Access Management (IAM) in the AWS Organizations management account can call these APIs, so access isn't limited to the owner of the account's root email with any associated multi-factor authentication (MFA) device.

This pattern implements alerts when the `CloseAccount` and `RemoveAccountFromOrganization` APIs are called, so you can monitor these activities. For alerts, it uses an Amazon Simple Notification Service (Amazon SNS) topic. You can also set up Slack notifications through a webhook.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An organization in AWS Organizations
- Access to the organization management account, under the organization's root, to create the required resources

Limitations

- As described in the AWS Organizations API reference, the `CloseAccount` API allows only 10 percent of active member accounts to be closed within a rolling 30-day period.
- When an AWS account is closed, its status is changed to SUSPENDED. For 90 days after this status transition, AWS Support can reopen the account. After 90 days the account is permanently deleted.

- Users who have access to the AWS Organizations management account and APIs might also have permissions to disable these alerts. If the primary concern is malicious behavior instead of accidental deletion, consider protecting the resources created by this pattern with an IAM permissions boundary.
- The API calls for `CloseAccount` and `RemoveAccountFromOrganization` are processed in the US East (N. Virginia) Region (`us-east-1`). Therefore, you must deploy this solution in `us-east-1` in order to observe the events.

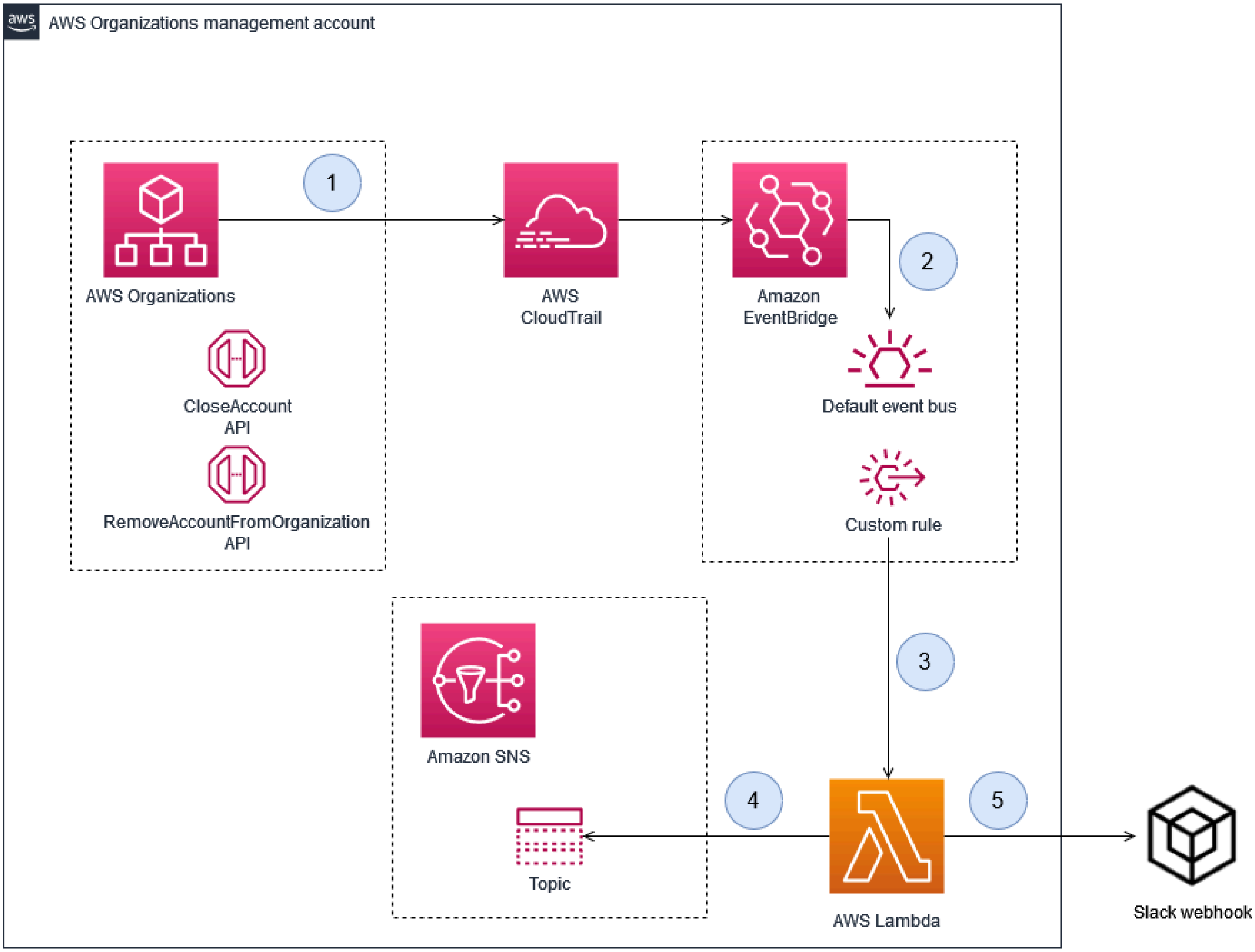
Architecture

Target technology stack

- AWS Organizations
- AWS CloudTrail
- Amazon EventBridge
- AWS Lambda
- Amazon SNS

Target architecture

The following diagram shows the solution architecture for this pattern.



1. AWS Organizations processes a `CloseAccount` or `RemoveAccountFromOrganization` request.
2. Amazon EventBridge is integrated with AWS CloudTrail to deliver these events to the default event bus.
3. A custom Amazon EventBridge rule matches the AWS Organizations requests and calls an AWS Lambda function.
4. The Lambda function delivers a message to an SNS topic, which users can subscribe to for email alerts or further processing.
5. If Slack notifications are enabled, the Lambda function delivers a message to a Slack webhook.

Tools

AWS services

- AWS CloudFormation provides a way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code.
- Amazon EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. EventBridge receives an event, an indicator of a change in environment, and applies a rule to route the event to a target. Rules match events to targets based on either the structure of the event, called an *event pattern*, or on a schedule.
- AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests each day to thousands each second. You pay only for the compute time that you consume. There is no charge when your code is not running.
- AWS Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources. Using AWS Organizations, you can programmatically create new AWS accounts and allocate resources, group accounts to organize your workflows, apply policies to accounts or groups for governance, and simplify billing by using a single payment method for all your accounts.
- AWS CloudTrail monitors and records account activity across your AWS infrastructure, and gives you control over storage, analysis, and remediation actions.
- Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication.

Other tools

- AWS Lambda Powertools for Python library is a set of utilities that provide tracing, logging, metrics, and event handling features for Lambda functions.

Code

The code for this pattern is located in the GitHub AWS Account Closer Notifier repository.

The solution includes a CloudFormation template that deploys the architecture for this pattern. It uses the AWS Lambda Powertools for Python library to provide logging and tracing.

Epics

Deploy the architecture

Task	Description	Skills required
Launch the CloudFormation template for the solution stack.	The CloudFormation template for this pattern is in the main branch of the [GitHub repository](). It deploys the IAM roles, EventBridge rules, Lambda functions, and the SNS topic. To launch the template: 1. Clone the [GitHub repository]() to obtain a copy of the solution code. 2. Open the AWS Management Console for the AWS Organizations management account. 3. Choose the US East (N. Virginia) Region (`us-east-1`), and then open the [CloudFormation console](). 4. Create the stack by using the `account-closure-notifier.yml` template and specifying the following values: • Stack name: `aws-account-closure-notifier-stack`	AWS administrator

Task	Description	Skills required
	• `ResourcePrefix` parameter: `aws-account-closure-notifier` • `SlackNotification` parameter: If Slack notifications are required, change this setting to `true`. • `SlackWebhookEndpoint` parameter: If Slack notifications are required, specify the webhook URL. For more information about launching a CloudFormation stack, see the [AWS documentation].	
Verify that the solution has launched successfully.	1. Wait for the CloudFormation stack to reach a status of CREATE_COMPLETE. 2. Open the [EventBridge console] in `us-east-1` . 3. Verify that a new rule has been created with the name `aws-account-closure-notifier-event-rule` .	AWS administrator

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Subscribe to the SNS topic.</td><td>(Optional) If you want to subscribe to the SNS topic:

1. Open the Amazon SNS console in <code>us-east-1</code> , and find the topic named <code>aws-account-closure-notifier-sns-topic</code> .
2. Choose the topic name, and then choose Create subscription.
3. For Protocol, choose Email.
4. For Endpoint, specify the email address that should receive the notification, and then choose Create subscription.
5. Check your email inbox for a message from AWS Notifications. Use the link in this email to confirm the subscription.

For more information about setting up SNS notifications, see the Amazon SNS documentation.</td><td>AWS administrator</td></tr>
</table>

Verify the solution

Task	Description	Skills required
Send a test event to the default event bus.	The [GitHub repository](#) provides a sample event that you can send to the EventBridge default event bus for testing. The EventBridge rule also reacts to events that use the custom event source `account.closure.notifier` . **Note** You can't use the CloudTrail event source to send this event, because it's not possible to send an event as an AWS service. To send a test event: 1. Open the [EventBridge console](#) in `us-east-1` . 2. In the navigation pane, under **Buses**, choose **Event buses**, and then select the default event bus. 3. Choose **Send events**. 4. For **Event source**, enter `account.closure.notifier` .	AWS administrator

Task	Description	Skills required
	5. For **Detail type**, enter `AWS API Call via CloudTrail` . 6. For **Event detail**, copy and paste the contents of `tests/dummy-event.json` from the GitHub repository into the text box. 7. Choose **Send** to initiate the notification workflow.	
Verify that the email notificat ion was received.	Check the mailbox that subscribed to the SNS topic for notifications. You should receive an email with details of the account that was closed and the principal that performed the API call.	AWS administrator
Verify that the Slack notificat ion was received.	(Optional) If you specified a webhook URL for the `SlackWebhookEndpoint` parameter when you deployed the CloudFormation template, check the Slack channel that is mapped to the webhook. It should display a message with details of the account that was closed and the principal that performed the API call.	AWS administrator

Related resources

- CloseAccount action (AWS Organizations API reference)
- RemoveAccountFromOrganization action (AWS Organizations API reference)
- AWS Lambda Powertools for Python

View EBS snapshot details for your AWS account or organization

Arun Chandapillai and Parag Nagwekar, Amazon Web Services

Summary

This pattern describes how you can automatically generate an on-demand report of all Amazon Elastic Block Store (Amazon EBS) snapshots in your Amazon Web Services (AWS) account or organizational unit (OU) in AWS Organizations.

Amazon EBS is an easy-to-use, scalable, high-performance block- storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). An EBS volume provides durable and persistent storage that you can attach to your EC2 instances. You can use EBS volumes as primary storage for your data and take a point-in-time backup of your EBS volumes by creating a snapshot. You can use the AWS Management Console or the AWS Command Line Interface (AWS CLI) to view the details of specific EBS snapshots. This pattern provides a programmatic way to retrieve information about all EBS snapshots in your AWS account or OU.

You can use the script provided by this pattern to generate a comma-separated values (CSV) file that has the following information about each snapshot: account ID, snapshot ID, volume ID and size, the date the snapshot was taken, instance ID, and description. If your EBS snapshots are tagged, the report also includes the owner and team attributes.

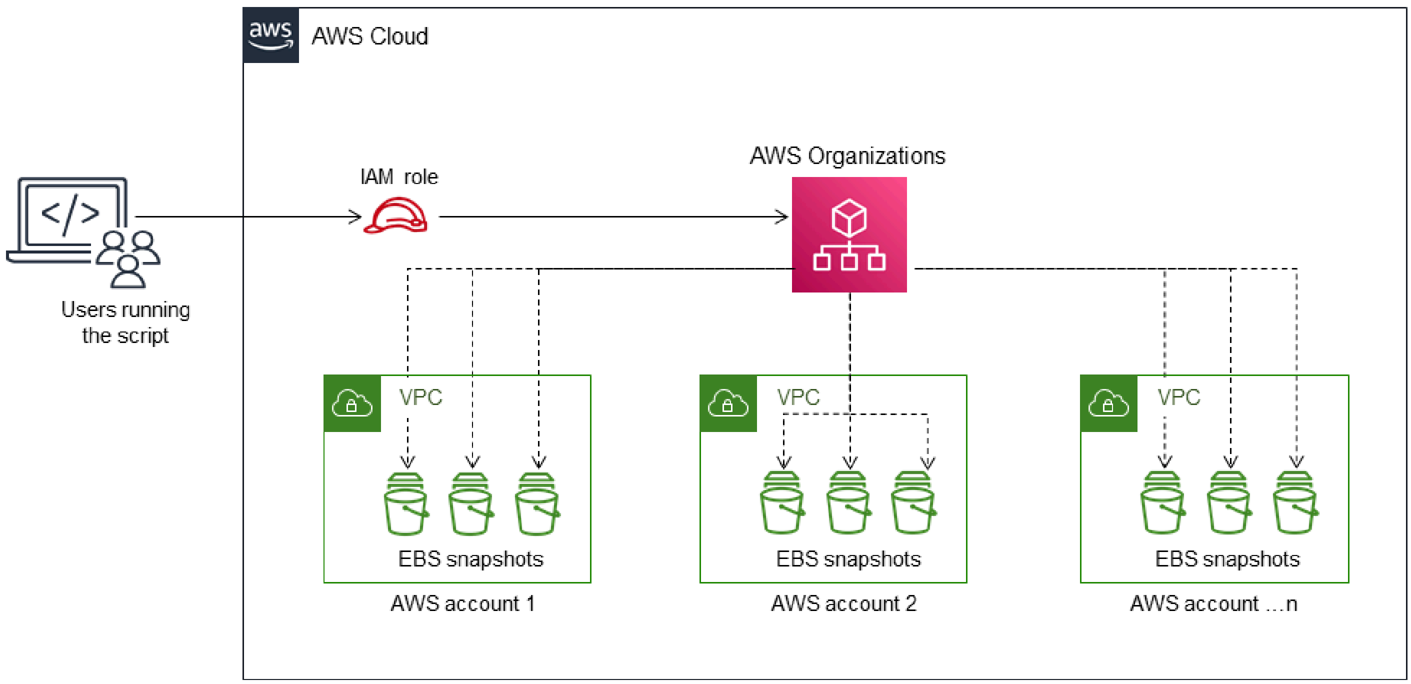
Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS CLI version 2 installed and configured
- AWS Identity and Access Management (IAM) role with the appropriate permissions (access permissions for a specific account or for all accounts in an OU if you're planning to run the script from AWS Organizations)

Architecture

The following diagram shows the script workflow that generates an on-demand report of EBS snapshots that are spread across multiple AWS accounts in an OU.



Tools

AWS services

- [AWS Command Line Interface (AWS CLI)](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with EC2 instances.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.

Code

The code for the sample application used in this pattern is available on GitHub, in the aws-ebs-snapshots-awsorganizations repository. Follow the instructions in the next section to use the sample files.

Epics

Download the script

Task	Description	Skills required
Download the Python script.	Download the script [GetSnapshotDetailsAllAccountsOU.py]() from the [GitHub repository]().	General AWS

Get EBS snapshot details for an AWS account

Task	Description	Skills required

Run the Python script.	Run the command:
	`python3 getsnapshotinfo.py --file <output-file>.csv --region <region-name>`
	where `<output-file>` refers to the CSV output file where you want information about the EBS snapshots placed, and `<region-name>` is the AWS Region where the snapshots are stored. For example:
	`python3 getsnapshotinfo.py --file snapshots.csv --region us-east-1`
		General AWS

Get EBS snapshot details for an organization

Task	Description	Skills required
Run the Python script.	Run the command: `python3 getsnapshotinfo.py --file <output-file>.csv --role <IAM-role> --region <region-name>` where `<output-file>` refers to the CSV output file where you want information about the EBS snapshots placed, `<IAM-role>` is a role that provides permissions to access AWS Organizations, and `<region-name>` is the AWS Region where the snapshots are stored. For example: `python3 getsnapshotinfo.py --file snapshots.csv --role <IAM role> --region us-west-2`	General AWS

Related resources

- Amazon EBS documentation
- Amazon EBS actions
- Amazon EBS API reference
- Improving Amazon EBS performance

- Amazon EBS resources
- EBS snapshot pricing

Additional information

EBS snapshot types

Amazon EBS provides three types of snapshots, based on ownership and access:

- **Owned by you –** By default, only you can create volumes from snapshots that you own.
- **Public snapshots –** You can share snapshots publicly with all other AWS accounts. To create a public snapshot, you modify the permissions for a snapshot to share it with the AWS accounts that you specify. Users that you will authorize can then use the snapshots you share by creating their own EBS volumes, while your original snapshot remains unaffected. You can also make your unencrypted snapshots available publicly to all AWS users. However, you can't make your encrypted snapshots available publicly for security reasons. Public snapshots pose a significant security risk because of the possibility of exposing personal and sensitive data. We strongly recommend against sharing your EBS snapshots with all AWS accounts. For more information about sharing snapshots, see the AWS documentation.
- **Private snapshots –** You can share snapshots privately with individual AWS accounts that you specify. To share the snapshot privately with specific AWS accounts, follow the instructions in the AWS documentation, and choose **Private** for the permissions setting. Users that you have authorized can use the snapshots that you share to create their own EBS volumes, while your original snapshot remains unaffected.

Overviews and procedures

The following table provides links to more information about EBS snapshots, including how you can lower EBS volume costs by finding and deleting unused snapshots, and archive rarely accessed snapshots that do not require frequent or fast retrieval.

For information about	See
Snapshots, their features, and limitations	Create Amazon EBS snapshots
How to create a snapshot	Console: Create a snapshot
	AWS CLI: create-snapshot command

	For example: `aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description " volume snapshot"`
Deleting snapshots (general information)	Delete an Amazon EBS snapshot
How to delete a snapshot	Console: Delete a snapshot AWS CLI: delete-snapshot command For example: `aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0`
Archiving snapshots (general information)	Archive Amazon EBS snapshots Amazon EBS Snapshots Archive (blog post)
How to archive a snapshot	Console: Archive a snapshot AWS CLI: modify-snapshot-tier command
How to retrieve an archived snapshot	Console: Restore an archived snapshot AWS CLI: restore-snapshot-tier command
Snapshot pricing	Amazon EBS pricing

FAQ

What is the minimum archive period?

The minimum archive period is 90 days.

How long would it take to restore an archived snapshot?

It can take up to 72 hours to restore an archived snapshot from the archive tier to the standard tier, depending on the size of the snapshot.

Are archived snapshots full snapshots?

Archived snapshots are always full snapshots.

Which snapshots can a user archive?

You can archive only snapshots that you own in your account.

Can you archive a snapshot of the root device volume of a registered Amazon Machine Image (AMI)?

No, you can’t archive a snapshot of the root device volume of a registered AMI.

What are security considerations for sharing a snapshot?

When you share a snapshot, you are giving others access to all the data on the snapshot. Share snapshots only with people that you trust with your data.

How do you share a snapshot with another AWS Region?

Snapshots are constrained to the Region in which they were created. To share a snapshot with another Region, copy the snapshot to that Region and then share the copy.

Can you share snapshots that are encrypted?

You can't share snapshots that are encrypted with the default AWS managed key. You can share snapshots that are encrypted with a customer managed key only. When you share an encrypted snapshot, you must also share the customer managed key that was used to encrypt the snapshot.

What about unencrypted snapshots?

You can share unencrypted snapshots publicly.

More patterns

- Automate account creation by using the Landing Zone Accelerator on AWS
- Automate AWS infrastructure operations by using Amazon Bedrock
- Automate AWS resource assessment
- Automatically inventory AWS resources across multiple accounts and Regions
- Automate AWS Service Catalog portfolio and product deployment by using AWS CDK
- Automate dynamic pipeline management for deploying hotfix solutions in Gitflow environments by using AWS Service Catalog and AWS CodePipeline
- Automate ingestion and visualization of Amazon MWAA custom metrics on Amazon Managed Grafana by using Terraform
- Automate security scans for cross-account workloads using Amazon Inspector and AWS Security Hub
- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK
- Automatically encrypt existing and new Amazon EBS volumes
- Centralize monitoring by using Amazon CloudWatch Observability Access Manager
- Check EC2 instances for mandatory tags at launch
- Clean up AWS Account Factory for Terraform (AFT) resources safely after state file loss
- Configure logging and monitoring for security events in an AWS IoT environment
- Create an Amazon ECS task definition and mount a file system on EC2 instances using Amazon EFS
- Create AWS Config custom rules by using AWS CloudFormation Guard policies
- Customize default role names by using AWS CDK aspects and escape hatches
- Delete unused Amazon EBS volumes by using AWS Config and AWS Systems Manager
- Deploy and manage AWS Control Tower controls by using AWS CDK and CloudFormation
- Deploy and manage AWS Control Tower controls by using Terraform
- Deploy code in multiple AWS Regions using AWS CodePipeline, AWS CodeCommit, and AWS CodeBuild
- Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates
- Export a report of AWS IAM Identity Center identities and their assignments by using PowerShell

- Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere
- Give SageMaker notebook instances temporary access to a CodeCommit repository in another AWS account
- Integrate Stonebranch Universal Controller with AWS Mainframe Modernization
- Launch a CodeBuild project across AWS accounts using Step Functions and a Lambda proxy function
- Manage AWS permission sets dynamically by using Terraform
- Migrate Windows SSL certificates to an Application Load Balancer using ACM
- Monitor IAM root user activity
- Optimize multi-account serverless deployments by using the AWS CDK and GitHub Actions workflows
- Preserve routable IP space in multi-account VPC designs for non-workload subnets
- Provision least-privilege IAM roles by deploying a role vending machine solution
- Register multiple AWS accounts with a single email address by using Amazon SES
- Remove Amazon EC2 entries across AWS accounts from AWS Managed Microsoft AD by using AWS Lambda automation
- Remove Amazon EC2 entries in the same AWS account from AWS Managed Microsoft AD by using AWS Lambda automation
- Rotate database credentials without restarting containers
- Run event-driven and scheduled workloads at scale with AWS Fargate
- Send notifications for an Amazon RDS for SQL Server database instance by using an on-premises SMTP server and Database Mail
- Set up a Grafana monitoring dashboard for AWS ParallelCluster
- Set up centralized logging at enterprise scale by using Terraform
- Streamline Amazon EC2 compliance management with Amazon Bedrock agents and AWS Config
- Tag Transit Gateway attachments automatically using AWS Organizations
- Use BMC Discovery queries to extract migration data for migration planning
- Verify operational best practices for PCI DSS 4.0 by using AWS Config
- Visualize IAM credential reports for all AWS accounts using Amazon QuickSight

Messaging & communications

Topics

- Automate RabbitMQ configuration in Amazon MQ
- Improve call quality on agent workstations in Amazon Connect contact centers
- More patterns

Automate RabbitMQ configuration in Amazon MQ

Yogesh Bhatia and Afroz Khan, Amazon Web Services

Summary

[Amazon MQ](https://aws.amazon.com/amazon-mq/) is a managed message broker service that provides compatibility with many popular message brokers. Using Amazon MQ with RabbitMQ provides a robust RabbitMQ cluster managed in the Amazon Web Services (AWS) Cloud with multiple brokers and configuration options. Amazon MQ provides a highly available, secure, and scalable infrastructure, and can process a large number of messages per second with ease. Multiple applications can use the infrastructure with different virtual hosts, queues, and exchanges. However, managing these configuration options or creating the infrastructure manually can require time and effort. This pattern describes a way to manage configurations for RabbitMQ in one step, through a single file. You can embed the code provided with this pattern within any continuous integration (CI) tool such as Jenkins or Bamboo.

You can use this pattern to configure any RabbitMQ cluster. All it requires is connectivity to the cluster. Although there are many other ways to manage RabbitMQ configurations, this solution creates entire application configurations in one step, so you can manage queues and other details easily.

Prerequisites and limitations

Prerequisites

- AWS Command Line Interface (AWS CLI) installed and configured to point to your AWS account (for instructions, see [AWS CLI documentation](https://docs.aws.amazon.com/cli/))
- Ansible installed, so you can run playbooks to create the configuration
- **rabbitmqadmin** installed (for instructions, see the [RabbitMQ documentation](https://www.rabbitmq.com/management-cli.html))
- A RabbitMQ cluster in Amazon MQ, created with healthy Amazon CloudWatch metrics

Additional requirements

- Make sure to create the configurations for virtual hosts and users separately and not as part of JSON.
- Make sure that the configuration JSON is part of the repository and is version-controlled.
- The version of the **rabbitmqadmin** CLI must be the same as the version of the RabbitMQ server, so the best option is to download the CLI from the RabbitMQ console.

- As part of the pipeline, make sure that JSON syntax is validated before each run.

Product versions

- AWS CLI version 2.0
- Ansible version 2.9.13
- **rabbitmqadmin** version 3.9.13 (must be the same as the RabbitMQ server version)

Architecture

Source technology stack

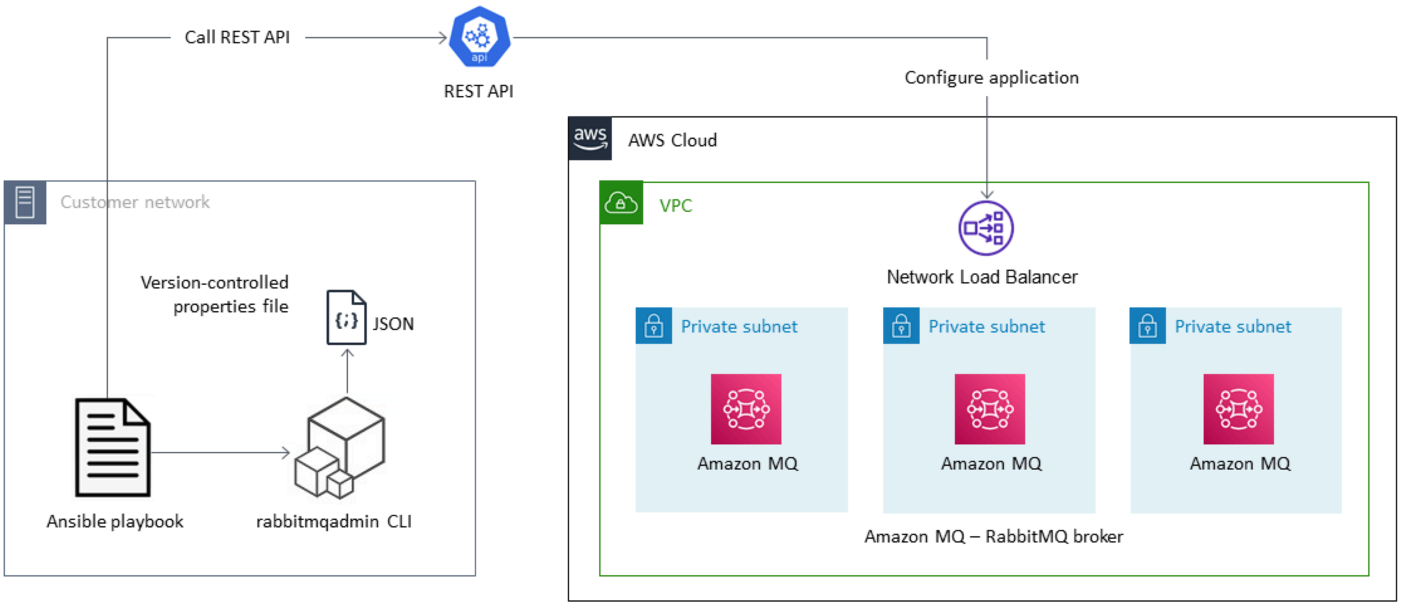
- An RabbitMQ cluster running on an existing on-premises virtual machine (VM) or a Kubernetes cluster (on premises or in the cloud)

Target technology stack

- Automated RabbitMQ configurations on Amazon MQ for RabbitMQ

Target architecture

There are many ways to configure RabbitMQ. This pattern uses the import configuration functionality, where a single JSON file contains all the configurations. This file applies all settings and can be managed by a version-control system such as Bitbucket or Git. This pattern uses Ansible to implement the configuration through the **rabbitmqadmin** CLI.



Tools

Tools

- rabbitmqadmin is a command-line tool for the RabbitMQ HTTP-based API. It is used to manage and monitor RabbitMQ nodes and clusters.
- Ansible is an open-source tool for automating applications and IT infrastructure.
- AWS CLI enables you to interact with AWS services by using commands in a command-line shell.

AWS services

- Amazon MQ is a managed message broker service that makes it easy to set up and operate message brokers in the cloud.
- AWS CloudFormation helps you set up your AWS infrastructure and speed up cloud provisioning with infrastructure as code.

Code

The JSON configuration file used in this pattern and a sample Ansible playbook are provided in the attachment.

Epics

Create your AWS infrastructure

Task	Description	Skills required
Create a RabbitMQ cluster on AWS.	If you don't already have a RabbitMQ cluster, you can use [AWS CloudFormation](https://aws.amazon.com/cloudformation/) to create the stack on AWS. Or, you can use the [Cloudformation module in Ansible](https://docs.ansible.com/ansible/latest/collections/amazon/aws/cloudformation_module.html) to create the stack. With the latter approach, you can use Ansible for both tasks: to create the RabbitMQ infrastructure and to manage configurations.	AWS CloudFormation, Ansible

Create the Amazon MQ for RabbitMQ configuration

Task	Description	Skills required
Create a properties file.	Download the JSON configuration file (`rabbitmqconfig.json`) in the attachment, or export it from the RabbitMQ console. Modify it to configure queues, exchanges, and bindings. This configuration file demonstrates the following: - Creates two queues: `sample-queue1` and `sample-queue2`	JSON

Task	Description	Skills required
	- Creates two exchanges: `sample-exchange1` and `sample-exchange2` - Implements the binding between the queues and exchanges These configurations are performed under the root (/) virtual host, as required by **rabbitmqadmin**.	

Task	Description	Skills required
Retrieve the details of the Amazon MQ for RabbitMQ infrastructure.	Retrieve the following details for the RabbitMQ infrastru cture on AWS: • Broker name • RabbitMQ host • RabbitMQ user name (the administrator user created during cluster creation) • RabbitMQ password You can use the AWS Management Console or the AWS CLI to retrieve this information. These details enable the Ansible playbook to connect to your AWS account and use the RabbitMQ cluster to run commands. **Important** The computer that runs the Ansible playbook must be able to access your AWS account, and AWS CLI must already be configured, as described in the *Prerequisites* section.	AWS CLI, Amazon MQ

Task	Description	Skills required

Create the hosts_var file.	Create the `hosts_var` file for Ansible and make sure that all the variables are defined in the file. Consider using Ansible Vault to store the password. You can configure the `hosts_var` file as follows (replace the asterisks with your informati on):
	`RABBITMQ_HOST: "***********.mq.us-east-2.amazonaws.com"`
	`RABBITMQ_VHOST: "/"`
	`RABBITMQ_USERNAME: "admin"`
	`RABBITMQ_PASSWORD: "*******"`
		Ansible

Task	Description	Skills required
Create an Ansible playbook.	For a sample playbook, see `ansible-rabbit-config.yaml` in the attachment. Download and save this file. The Ansible playbook imports and manages all RabbitMQ configurations, such as queues, exchanges, and bindings, that applications require. Follow best practices for Ansible playbooks, such as securing passwords. Use Ansible Vault for password encryption, and retrieve the RabbitMQ password from the encrypted file.	Ansible

Deploy the configuration

Task	Description	Skills required
Run the playbook.	Run the Ansible playbook that you created in the previous epic. `ansible-playbook ansible-rabbit-config.yaml` You can verify the new configurations on the RabbitMQ console.	RabbitMQ, Amazon MQ, Ansible

Related resources

- Migrating from RabbitMQ to Amazon MQ (AWS blog post)
- Management Command Line Tool (RabbitMQ documentation)
- Create or delete an AWS CloudFormation stack (Ansible documentation)
- Migrating message driven applications to Amazon MQ for RabbitMQ (AWS blog post)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Improve call quality on agent workstations in Amazon Connect contact centers

Ernest Ozdoba, Amazon Web Services

Summary

Call quality issues are some of the most difficult problems to troubleshoot in contact centers. To avoid voice quality issues and complex troubleshooting procedures, you must optimize your agents' work environment and workstation settings. This pattern describes voice quality optimization techniques for agent workstations in Amazon Connect contact centers. It provides recommendations in the following areas:

- Work environment adjustments. Agents' surroundings don't affect how voice is transmitted over the network, but they do have an effect on call quality.
- Agent workstation settings. Hardware and network configurations for contact center workstations have significant effects on call quality.
- Browser settings. Agents use a web browser to access the Amazon Connect Contact Control Panel (CCP) website and communicate with customers, so browser settings can affect call quality.

The following components can also affect call quality, but they fall outside the scope of the workstation and aren't covered in this pattern:

- Traffic flows to the Amazon Web Services (AWS) Cloud over AWS Direct Connect, a full-tunnel VPN, or a split-tunnel VPN
- Network conditions when working in or outside the corporate office
- Public switched telephone network (PSTN) connectivity
- The customer's device and telephony carrier
- Virtual desktop infrastructure (VDI) setup

For more information relating to these areas, see Common Contact Control Panel (CCP) Issues and Use the Endpoint Test Utility in the Amazon Connect documentation.

Prerequisites and limitations

Prerequisites

- Headsets and workstations must comply with the requirements specified in the Amazon Connect Administrator Guide.

Limitations

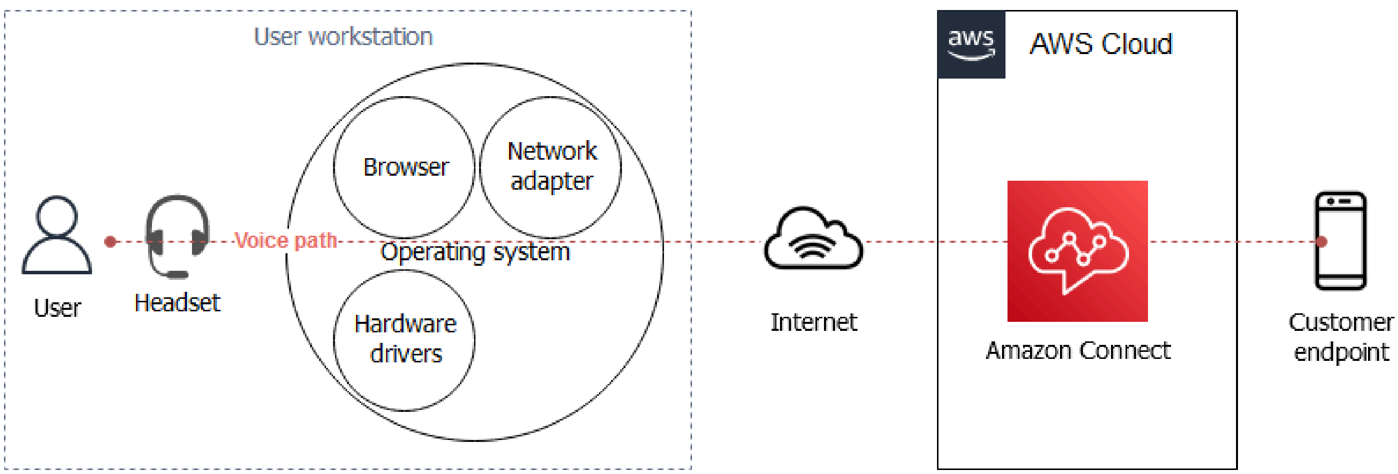
- The optimization techniques in this pattern apply to soft phone voice quality. They do not apply when you configure the Amazon Connect CCP in desk phone mode. However, you can use desk phone mode if your soft phone setup doesn't provide acceptable voice quality for the call.

Product versions

- For supported browsers and versions, see the Amazon Connect Administrator Guide.

Architecture

This pattern is architecture-agnostic because it targets agent workstation settings. As the following diagram shows, the voice path from the agent to the customer is affected by the agent's headset, browser, operating system, workstation hardware, and network.



In Amazon Connect contact centers, the user's audio connectivity is established with WebRTC. Voice is encoded with the Opus interactive audio codec and encrypted with the Secure Real-time Transport Protocol (SRTP) in transit. Other network architectures are possible, including VPN, private WAN/LAN, and ISP networks.

Tools

- [Amazon Connect Endpoint Test Utility](#) – This utility checks network connectivity and browser settings.
- Browser configuration editors for WebRTC settings:
  - For Firefox: **about:config**
  - For Chrome: **chrome://flags**
- [CCP Log Parser](#) – This tool helps you analyze CCP logs for troubleshooting purposes.

Epics

Adjust the work environment

Task	Description	Skills required
Reduce background noise.	Avoid noisy environments. If this is not possible, optimize the environment with these soundproofing tips: • Absorb noise by using sound-dissipating surfaces such as curtains, carpets, and soft furnishings. • Block noise by putting barriers between desks. • Consider an active noise cancellation (ANC) solution such as a white noise generator to help concentra tion and ensure privacy, or use noise-canceling headsets. • Prevent echo on your calls. Big, empty spaces might create echo effects or	Agent, Manager

Task	Description	Skills required
	amplify noises. Covering surfaces that can bounce sounds will help reduce echoes.	

Optimize agent workstation settings

Task	Description	Skills required
Choose the right headset.	• If the environment is noisy, choose a stereo headset. Directing sound to both ears helps agents focus and hear the customer better, and reduces the overall noise by making it less likely for agents to raise their voices. • Avoid using loud speakers or built-in computer audio. For best quality, use a wired headset that’s dedicated to contact center use. Wireless headsets are convenient, but they might be a source of additional audio delay and reduced audio quality because of radio interfere nce and transcoding.	Agent, Manager
Use the headset as intended.	• Enable the active noise canceling and speech enhancement features of your headset if they are	Agent

Task	Description	Skills required
	available. Look for settings such as ANC or ANR. For instructions on activating these settings, see the user manual for your headset. • Adjust your microphone so you can speak directly into it. The best position for your microphone is just below your chin. Correct placement can make a difference of 10 decibels (dB) in the sound level. Most headsets allow you to rotate or bend the microphone arm (boom), so it is important to keep it in the right place when you are talking. • Some headsets are equipped with multiple microphones and advanced features such as voice beamforming, which helps capture speech without a boom. To make sure that you're using the main microphone as intended by the manufacturer, see the user manual for your device.	

Task	Description	Skills required
Check workstation resources.	Make sure that your agents' computers are performan t. If they use third-party applications that consume resources, their computers might not meet the minimum [hardware requirements](#) to run CCP. If agents experienc e call quality issues, make sure that they have enough processing power (CPU), disk space, network bandwidth , and memory available for CCP. Agents should close any unnecessary applicati ons and tabs to improve CCP performance and call quality.	Administrator

Task	Description	Skills required
Configure the operating system's sound settings.	The default settings for microphone level and boost usually work fine. If you find that outbound voice is quiet or the microphone is picking up too much, it might help to adjust these settings. Microphone settings can be found in your computer's system sound configuration (**Sound**, **Input** on MacOS, **Microphone Properties** in Windows). You can access advanced settings that might affect voice quality through system tools or third-party applications. Here are some of the settings you can check: • Sample rate – This value determines how many times the sound is probed per second. The default setting is usually 44 or 48 kilohertz (kHz). The optimal value for Amazon Connect is 48 kHz. You can use your browser settings to override the default value. For more information, see the troubleshooting section of the *Amazon Connect Administrator Guide*. • Gain – This value determine s how much the microphon	Agent, Administrator

Task	Description	Skills required
	e amplifies the sound. If you turn the gain up, your microphone might pick up more background noise. • Bit depth – This digital resolution value describes how many levels of sound amplitude are being recognized. The higher the bit depth, the smoother the voice sounds. However, many traditional telephony networks use the pulse-code modulation (PCM) standard, which supports only 8-bit resolution. • Open threshold – This is the minimal sound amplitude that a microphone picks up. If you’re experiencing voice quality issues, try restoring these values to their default settings before investigating further. For more information about these and other adjustabl e settings, see your device manual.	

Task	Description	Skills required
Use a wired network.	Typically, wired ethernet has lower latency, so it is easier to provide the consistent transmission quality required for voice data transmission. We recommend 100 KB bandwidth per call at the minimum. • If agents are working from home, we recommend wired over wireless connections. It shouldn't take more than 150 milliseconds to hear the customer. You can access Amazon Connect's latency test from the [Amazon Connect Endpoint Test Utility](). However, this utility measures the delay from the browser to Amazon Connect Regions, not to customers. The 150-millisecond one-way delay recommendation prevents the agent and customer from talking over each other. The value is measured from end to end, and each element adds a delay, including the part of the call between the Amazon Connect Region and the customer.	Network administrator, Agent

Task	Description	Skills required
	• If agents are working from the office, corporate Wi-Fi is acceptable as long as parameters are in the recommended range, and Real-time Transport Protocol (RTP) traffic is prioritized.	
Update hardware drivers.	When you use a USB or other type of headset that has its own firmware, we recommend that you keep it updated with the latest version. Simple headsets that use an auxiliary port use the computer’s built-in audio device, so make sure that the operating system hardware driver is up to date. In rare cases, an audio driver update can cause audio issues, and you might need to roll it back. For more information about changing firmware and driver versions, see your device manual.	Administrator

Task	Description	Skills required
Avoid USB hubs and dongles.	When you connect your headset, avoid additional devices such as dongles, port type converters, hubs, and extension cables. These devices might affect call quality. Connect your device directly to the port in your computer instead.	Agent

Task	Description	Skills required
Check CCP logs.	The CCP Log Parser provides an easy way to check applicati on logs. 1. Download the CCP logs after a call. 2. Open the CCP Log Parser. 3. Drag and drop the log file to upload the log for analysis. 4. When the log has been analyzed, the **Snapshots & Logs** tab will be selected by default. Choose the **Metrics** tab next to it to check insights. 5. In the **WebRTC Metrics - audio_input** section, check the following: • The **Audio Level** graph, to see if your received audio level is above 0. This indicates that audio was received from your caller. • The **Packets** graph for any lost packets. If this chart shows significant increases, contact your IT support team. 6. In the **WebRTC Metrics - audio_output** section, check the following:	Agent (advanced skills)

Task	Description	Skills required
	- The **Audio Level** graph, to confirm that audio was sent out from your device. - The **Packets** graph. If you see a packet loss spike, report it to your IT support team. - The **Jitter Buffer & RTT** graph. Round trip time (RTT) values above 300 will affect the call experience. Report these to your IT support team.	

Optimize browser settings

Task	Description	Skills required
Restore default WebRTC settings.	WebRTC has to be enabled to make soft phone calls with CCP. We recommend that you keep the default settings for WebRTC-related features. - In Chrome, you can set flags by navigating to the URL **chrome://flags**. Type **WebRTC** in the search box to find settings that can interfere with CCP, and set these to **Default**. - In Firefox, type **about:config** in the address bar, and	Administrator

Task	Description	Skills required
	then type **WebRTC** in the search box on the configura tion page. Non-default settings appear in bold text and can be changed to **Default**.	
Disable browser extensions when troubleshooting.	Some browser extension s might affect call quality or even prevent calls from connecting properly. Use the incognito window or private mode in your browser, and disable all extensions. If that solves the problem, review your browser extensions and look for suspicious add-ons, or disable them individually.	Agent, Administrator
Check the browser sample rate.	Confirm that your microphon e input is set to the optimal 48 kHz sample rate. For instructions, see the Amazon Connect Administrator Guide.	Agent, Administrator

Related resources

If you've followed the steps in this pattern but you're still encountering problems with call quality, see the following resources for troubleshooting tips.

- Review common Contact Control Panel (CCP) issues.
- Check the connection with the Endpoint Test Utility.
- Follow the troubleshooting guide for any other issues.

If your troubleshooting and adjustments don't solve the call quality issue, the root cause might be external to your workstation. For further troubleshooting, contact your IT support team.

More patterns

- Decompose monoliths into microservices by using CQRS and event sourcing
- Deploy a ChatOps solution to manage SAST scan results by using Amazon Q Developer in chat applications custom actions and AWS CloudFormation
- Integrate Amazon API Gateway with Amazon SQS to handle asynchronous REST APIs
- Run message-driven workloads at scale by using AWS Fargate
- Streamline Amazon Lex bot development and deployment by using an automated workflow

Security, identity & compliance

Topics

- Access AWS services from an ASP.NET Core app using Amazon Cognito identity pools
- Authenticate Microsoft SQL Server on Amazon EC2 using AWS Directory Service
- Automate incident response and forensics
- Automate remediation for AWS Security Hub standard findings
- Automate security scans for cross-account workloads using Amazon Inspector and AWS Security Hub
- Automatically audit AWS security groups that allow access from public IP addresses
- Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config
- Automatically remediate unencrypted Amazon RDS DB instances and clusters
- Automatically rotate IAM user access keys at scale with AWS Organizations and AWS Secrets Manager
- Automatically validate and deploy IAM policies and roles by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros
- Bidirectionally integrate AWS Security Hub with Jira software
- Build a pipeline for hardened container images using EC2 Image Builder and Terraform
- Centralize IAM access key management in AWS Organizations by using Terraform
- Check an Amazon CloudFront distribution for access logging, HTTPS, and TLS version
- Check for single-host network entries in security group ingress rules for IPv4 and IPv6
- Choose an Amazon Cognito authentication flow for enterprise applications
- Create AWS Config custom rules by using AWS CloudFormation Guard policies
- Create a consolidated report of Prowler security findings from multiple AWS accounts
- Delete unused Amazon EBS volumes by using AWS Config and AWS Systems Manager
- Deploy and manage AWS Control Tower controls by using AWS CDK and CloudFormation
- Deploy and manage AWS Control Tower controls by using Terraform
- Deploy the Security Automations for AWS WAF solution by using Terraform
- Deploy a pipeline that simultaneously detects security issues in multiple code deliverables
- Deploy detective attribute-based access controls for public subnets by using AWS Config
- Deploy preventative attribute-based access controls for public subnets

- Detect Amazon RDS and Aurora database instances that have expiring CA certificates
- Dynamically generate an IAM policy with IAM Access Analyzer by using Step Functions
- Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates
- Enable transparent data encryption in Amazon RDS for SQL Server
- Ensure AWS load balancers use secure listener protocols (HTTPS, SSL/TLS)
- Ensure encryption for Amazon EMR data at rest is enabled at launch
- Ensure that an IAM profile is associated with an EC2 instance
- Ensure an Amazon Redshift cluster is encrypted upon creation
- Monitor and remediate scheduled deletion of AWS KMS keys
- Identify public Amazon S3 buckets in AWS Organizations by using Security Hub
- Ingest and analyze AWS security logs in Microsoft Sentinel
- Manage AWS Organizations policies as code by using AWS CodePipeline and Amazon Bedrock
- Manage AWS IAM Identity Center permission sets as code by using AWS CodePipeline
- Manage credentials using AWS Secrets Manager
- Monitor ElastiCache clusters for security groups
- Monitor Amazon EMR clusters for in-transit encryption at launch
- Monitor Amazon ElastiCache clusters for at-rest encryption
- Monitor EC2 instance key pairs using AWS Config
- Monitor IAM root user activity
- Send a notification when an IAM user is created
- Prevent internet access at the account level by using a service control policy
- Export a report of AWS IAM Identity Center identities and their assignments by using PowerShell
- Restrict access based on IP address or geolocation by using AWS WAF
- Scan Git repositories for sensitive information and security issues by using git-secrets
- Secure file transfers by using Transfer Family, Amazon Cognito, and GuardDuty
- Send alerts from AWS Network Firewall to a Slack channel
- Send custom attributes to Amazon Cognito and inject them into tokens
- Simplify private certificate management by using AWS Private CA and AWS RAM
- Streamline Amazon EC2 compliance management with Amazon Bedrock agents and AWS Config
- Turn off security standard controls across all Security Hub member accounts in a multi-account environment

- Update AWS CLI credentials from AWS IAM Identity Center by using PowerShell
- Use AWS Config to monitor Amazon Redshift security configurations
- Use Network Firewall to capture the DNS domain names from the Server Name Indication for outbound traffic
- Use Terraform to automatically enable Amazon GuardDuty for an organization
- Verify operational best practices for PCI DSS 4.0 by using AWS Config
- Verify that new Amazon Redshift clusters have required SSL endpoints
- Verify that new Amazon Redshift clusters launch in a VPC
- More patterns

Access AWS services from an ASP.NET Core app using Amazon Cognito identity pools

Bibhuti Sahu and Marcelo Barbosa, Amazon Web Services

Summary

This pattern discusses how you can configure Amazon Cognito user pools and identity pools, and then enable an ASP.NET Core app to access AWS resources after successful authentication.

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. The two main components of Amazon Cognito are user pools and identity pools.

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito. Your users can also sign in through social identity providers such as Google, Facebook, Amazon, or Apple, and through SAML identity providers.

Amazon Cognito identity pools (federated identities) enable you to create unique identities for your users and federate them with identity providers. With an identity pool, you can obtain temporary, limited-privilege AWS credentials to access other AWS services. Before you can begin using your new Amazon Cognito identity pool, you must assign one or more AWS Identity and Access Management (IAM) roles to determine the level of access you want your application users to have to your AWS resources. Identity pools define two types of identities: authenticated and unauthenticated. Each identity type can be assigned its own role in IAM. Authenticated identities belong to users who are authenticated by a public login provider (Amazon Cognito user pools, Facebook, Google, SAML, or any OpenID Connect providers) or a developer provider (your own backend authentication process), whereas unauthenticated identities typically belong to guest users. When Amazon Cognito receives a user request, the service determines whether the request is authenticated or unauthenticated, determines which role is associated with that authentication type, and then uses the policy attached to that role to respond to the request.

Prerequisites and limitations

Prerequisites

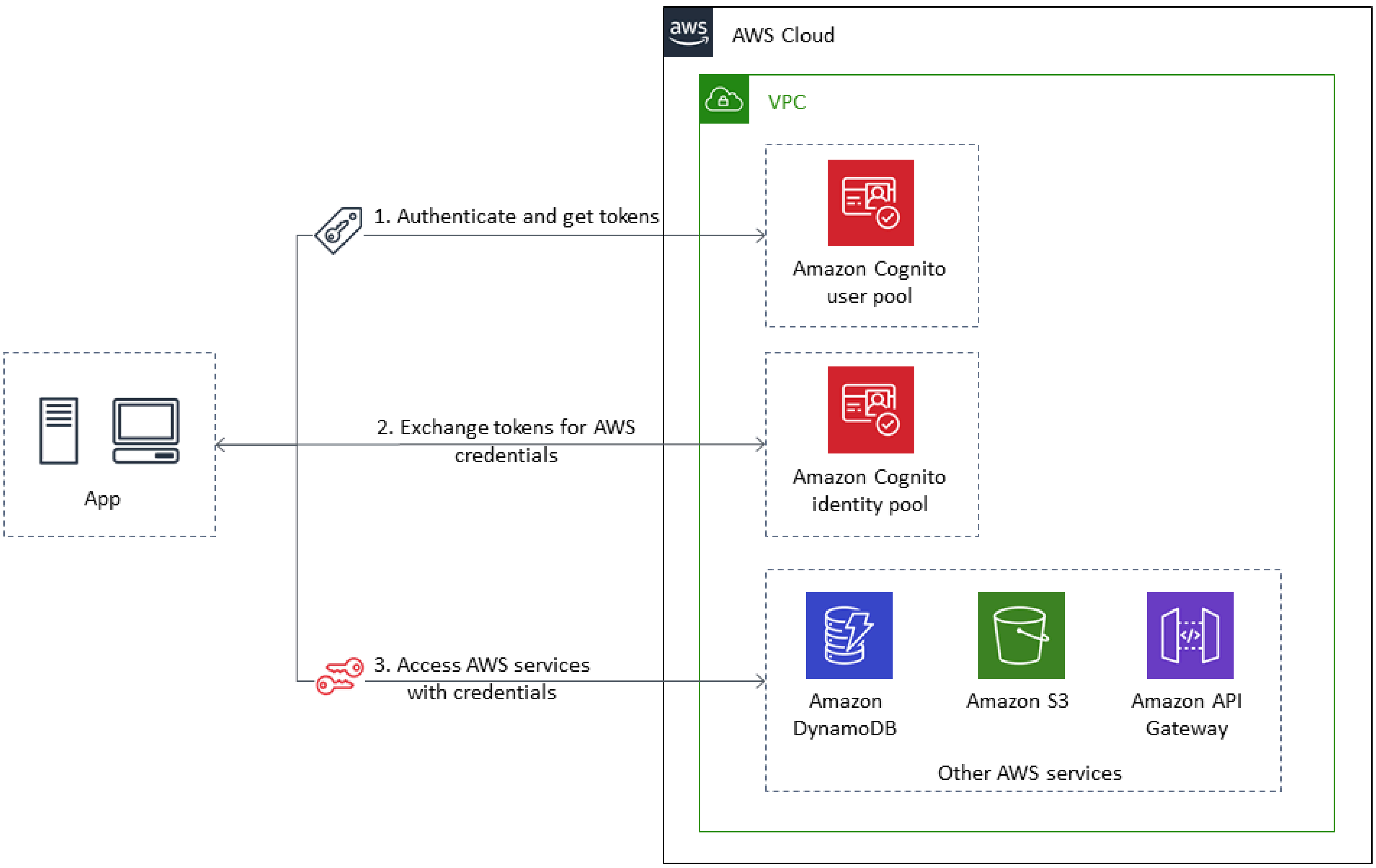
- An AWS account with Amazon Cognito and IAM permissions
- Access to the AWS resources you want to use
- ASP.NET Core 2.0.0 or later

Architecture

Technology stack

- Amazon Cognito
- ASP.NET Core

Target architecture



Tools

Tools, SDKs, and AWS services

- Visual Studio or Visual Studio Code
- Amazon.AspNetCore.Identity.Cognito (1.0.4) – NuGet package
- AWSSDK.S3 (3.3.110.32) – NuGet package
- Amazon Cognito

Code

The attached .zip file includes sample files that illustrate the following:

- How to retrieve an access token for the logged in user
- How to exchange an access token for AWS credentials
- How to access the Amazon Simple Storage Service (Amazon S3) service with AWS credentials

IAM role for authenticated identities

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "mobileanalytics:PutEvents",
        "cognito-sync:*",
        "cognito-identity:*",
        "s3:ListAllMyBuckets*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Epics

Create an Amazon Cognito user pool

Task	Description	Skills required
Create a user pool.	1. Sign in to the AWS Management Console and open the Amazon Cognito console. 2. Choose **Manage User Pools**.	Developer

Task	Description	Skills required
	3. In the top-right corner of the page, choose **Create a user pool**. 4. Provide a name for your user pool, choose **Review defaults**, and then choose **Create pool**. 5. Note the pool ID.	
Add an app client.	You can create an app to use the built-in webpages for signing up and signing in your users. 1. On the navigation bar on the left side of the user pool page, choose **App clients** under **General settings**, and then choose **Add an app client**. 2. Give your app a name, and then choose **Create app client**. 3. Note the app client ID and the client secret (choose **Show Details** to see the client secret).	Developer

Create an Amazon Cognito identity pool

Task	Description	Skills required
Create an identity pool.	1. On the Amazon Cognito console, choose **Manage**	Developer

Task	Description	Skills required
	Identity Pools, and then choose **Create new identity pool**. 2. Type a name for the identity pool. 3. If you want to enable unauthenticated identities, select that option from the **Unauthenticated identities** section. 4. In the **Authentication providers** section, configure the Amazon Cognito identity pool by setting the user pool ID and the app client ID, and then choose **Create Pool**.	
Assign IAM roles for the identity pool.	You can edit the IAM roles for authenticated and unauthenticated users, or keep the defaults, and then choose **Allow**. For this pattern, we will edit the authenticated IAM role and provide access for `s3:ListAllMyBuckets` . For sample code, see the IAM role provided earlier in the [Tools](#) section.	Developer

Task	Description	Skills required
Copy the identity pool ID.	When you choose **Allow** in the previous step, the **Getting started with Amazon Cognito** page is displayed. On this page, you can either copy the identity pool ID from the **Get AWS Credentials** section or choose **Edit identity pool** in the upper right and copy the identity pool ID from the screen that's displayed.	Developer

Configure your sample app

Task	Description	Skills required
Clone the sample ASP.NET Core web app.	1. Clone the sample .NET core web app from https://github.com/aws/aws-aspnet-cognito-identity-provider.git. 2. Navigate to the `samples` folder and open the solution. In this project, you'll configure the `appsettings.json` file and add a new page that will render all S3 buckets after successful sign in.	Developer
Add dependencies.	Add a NuGet dependency for `Amazon.AspNetCore.Identity.Cognito` to	Developer

Task	Description	Skills required
	your ASP.NET Core applicati on.	
Add the configuration keys and values to `appsettin gs.json` .	Include the code from the attached `appsettin gs.json` file in your `appsettings.json` file, and then replace the placeholders with the values from the previous steps.	Developer
Create a new user and sign in.	Create a new user in the Amazon Cognito user pool, and verify that the user exists under **Users and Groups** in the user pool.	Developer
Create a new Razor Page called `MyS3Buckets` .	Add a new ASP.NET Core Razor Page to your sample app, and replace the content for `MyS3Bucket.cshtml` and `MyS3Bucket.cshtml. cs` from the attached sample. Add the new **MyS3Bucket** page under navigation in the `_Layout.c shtml` page.	Developer

Troubleshooting

Issue	Solution
After you open the sample application from the GitHub repository, you get an error when	In the `src` folder, make sure to remove from the reference to the `Amazon.AspNetCore. Identity.Cognito` project from the

Issue	Solution
you try to add the NuGet package to the Samples project.	`Samples.sln` file. You can then add the NuGet package to the Samples project without any issues.

Related resources

- Amazon Cognito
- Amazon Cognito user pools
- Amazon Cognito identity pools
- Access policy samples
- GitHub - AWS ASP.NET Cognito Identity Provider

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Authenticate Microsoft SQL Server on Amazon EC2 using AWS Directory Service

Jagadish Kantubugata and Oludahun Bade Ajidahun, Amazon Web Services

Summary

This pattern describes how to create an AWS Directory Service directory and use it to authenticate Microsoft SQL Server on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

AWS Directory Service provides multiple ways to use Amazon Cloud Directory and Microsoft Active Directory (AD) with other AWS services. Directories store information about users, groups, and devices, and administrators use them to manage access to information and resources. AWS Directory Service provides multiple directory choices for users who want to use their existing Microsoft AD or Lightweight Directory Access Protocol (LDAP)–aware applications in the cloud. It also offers those same choices to developers who need a directory to manage users, groups, devices, and access.

Prerequisites and limitations

Prerequisites

- An active AWS account
- A virtual private cloud (VPC) with a minimum of two private subnets and two public subnets
- An AWS Identity and Access Management (IAM) role to join the server into the domain

Architecture

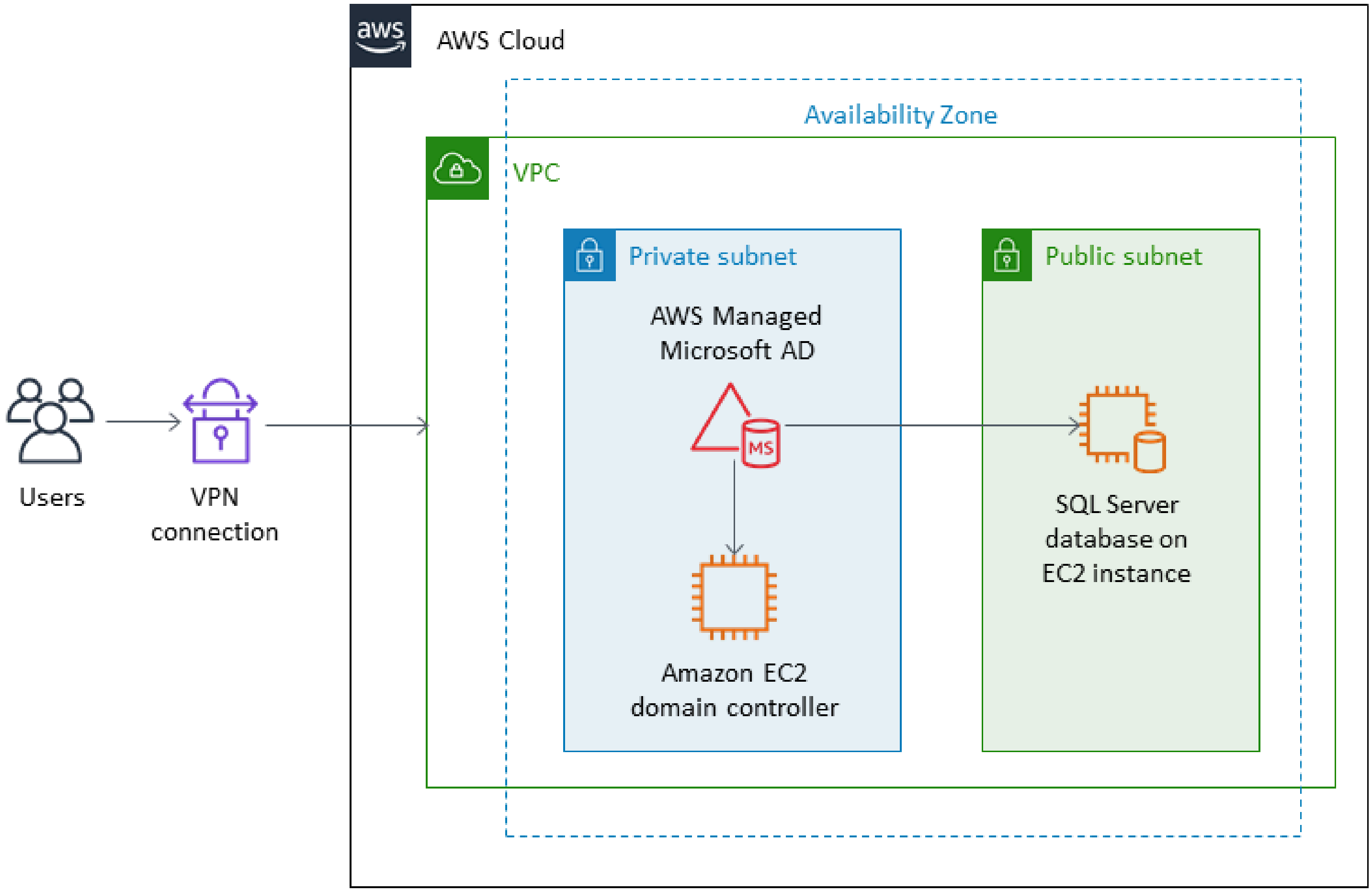
Source technology stack

- The source can be an on-premises Active Directory

Target technology stack

- AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD)

Target architecture



Tools

- SQL Server Management Studio (SSMS) is a tool for managing Microsoft SQL Server, including accessing, configuring, and administering SQL Server components.

Epics

Set up a directory

Task	Description	Skills required
Select AWS Managed Microsoft AD as the directory type.	On the [AWS Directory Service console](), choose **Directori es**, **Set up directory**, **AWS Managed Microsoft AD**, **Next**.	DevOps

Task	Description	Skills required
Select edition.	From the available editions for AWS Managed Microsoft AD, choose **Standard Edition**.	DevOps
Specify the directory DNS name.	Use a fully qualified domain name. This name will resolve inside your VPC only. It does not need to be publicly resolvable.	DevOps
Set the administrator password.	Set the password for the default administrative user, which is named **Admin**.	DevOps
Choose the VPC and subnets.	Choose the VPC that will contain your directory and the subnets for the domain controllers. If you do not have a VPC with at least two subnets, you must create one.	DevOps
Review and launch the directory.	Review the edition and price information for the directory , and then choose **Create directory**.	DevOps

Launch an EC2 instance for SQL Server in the domain

Task	Description	Skills required
Select an AMI for SQL Server.	The steps in this epic seamlessly join a Windows EC2 instance to your AWS Managed Microsoft AD directory.	DevOps, DBA

Task	Description	Skills required
	On the [Amazon EC2 console](https://console.aws.amazon.com/ec2/), choose **Launch instance**, and then select the appropriate Amazon Machine Image (AMI) for SQL Server.	
Configure instance details.	Configure the Windows instance to meet your requirements for SQL Server.	DevOps, DBA
Select the key pair name.	Select a key pair, and then launch the instance.	DevOps, DBA
Add a network.	You can choose the VPC that your directory was created in.	DevOps, DBA
Select an IAM role.	In **Advanced settings**, select an IAM profile that has the AWS managed policies `AmazonSSMManagedInstanceCore` and `AmazonSSMDirectoryServiceAccess` attached to it.	DevOps, DBA
Add a subnet.	Choose one of the public subnets in your VPC. The subnet that you choose must have all external traffic routed to an internet gateway. If this is not the case, you won't be able to connect to the instance remotely.	DevOps, DBA

Task	Description	Skills required
Choose your domain.	Choose the domain that you created from the **Domain join** directory list.	DevOps, DBA
Launch the instance.	Choose **Launch instance**.	DBA

Authenticate SQL Server using Directory Service

Task	Description	Skills required
Log in as a Windows administrator.	Log in to the Windows EC2 instance by using Windows administrator credentials.	DBA
Log in to SQL Server.	Launch SQL Server Management Studio (SSMS) and log in to SQL Server by using the Windows authentic ation method.	DBA
Create a login for the directory user.	In SSMS, choose **Security**, and then choose **New Login**.	DBA
Search for a login name.	Choose the search button next to the login text box.	DBA
Select a location.	In the **Select User or Group** dialog box, choose **Locations**.	DBA
Enter network credentials.	Enter the fully qualified network credentials you used when you created the directory service; for example: `test.com\admin` .	DBA

Task	Description	Skills required
Select the directory.	Choose the AWS directory name, and then choose **OK**.	DBA
Select an object name.	Select the user for which you want to create the login. Select the location, choose the entire directory, search for the user, and add the login.	DBA
Log in to the SQL Server instance.	Log in to the Windows EC2 instance for SQL Server by using your domain credentia ls.	DBA
Log in to SQL Server as a domain user.	Launch SSMS and connect to the database engine by using the Windows authentication method.	DBA

Related resources

- AWS Directory Service documentation (AWS website)
- Create your AWS Managed Microsoft AD directory (AWS Directory Service documentation)
- Seamlessly join a Windows EC2 instance (AWS Directory Service documentation)
- Microsoft SQL Server on AWS (AWS website)
- SSMS documentation (Microsoft website)
- Create a login in SQL Server (SQL Server documentation)

Automate incident response and forensics

Lucas Kauffman and Tomek Jakubowski, Amazon Web Services

Summary

This pattern deploys a set of processes that use AWS Lambda functions to provide the following:

- A way to initiate the incident-response process with minimum knowledge
- Automated, repeatable processes that are aligned with the *AWS Security Incident Response Guide*
- Separation of accounts to operate the automation steps, store artifacts, and create forensic environments

The Automated Incident Response and Forensics framework follows a standard digital forensic process consisting of the following phases:

1. Containment
2. Acquisition
3. Examination
4. Analysis

You can perform investigations on static data (for example, acquired memory or disk images) and on dynamic data that is live but on separated systems.

For more details, see the Additional information section.

Prerequisites and limitations

Prerequisites

- Two AWS accounts:
  - Security account, which can be an existing account, but is preferably new
  - Forensics account, preferably new
- AWS Organizations set up
- In the Organizations member accounts:

  - The Amazon Elastic Compute Cloud (Amazon EC2) role must have Get and List access to Amazon Simple Storage Service (Amazon S3) and be accessible by AWS Systems Manager. We recommend using the `AmazonSSMManagedInstanceCore` AWS managed role. Note that this role will automatically be attached to the Amazon EC2 instance when incident response is initiated. After the response has finished, AWS Identity and Access Management (IAM) will remove all rights to the instance.
  - Virtual private cloud (VPC) endpoints in the AWS member account and in the Incident Response and Analysis VPCs. Those endpoints are: S3 Gateway, EC2 Messages, SSM, and SSM Messages.
- AWS Command Line Interface (AWS CLI) installed on the Amazon EC2 instances. If the Amazon EC2 instances don't have AWS CLI installed, internet access will be required for the disk snapshot and memory acquisition to work. In this case, the scripts will reach out to the internet to download the AWS CLI installation files and will install them on the instances.

Limitations

- This framework does not intend to generate artifacts that can be considered as electronic evidence, submissible in court.
- Currently, this pattern supports only Linux based instances running on x86 architecture.

Architecture

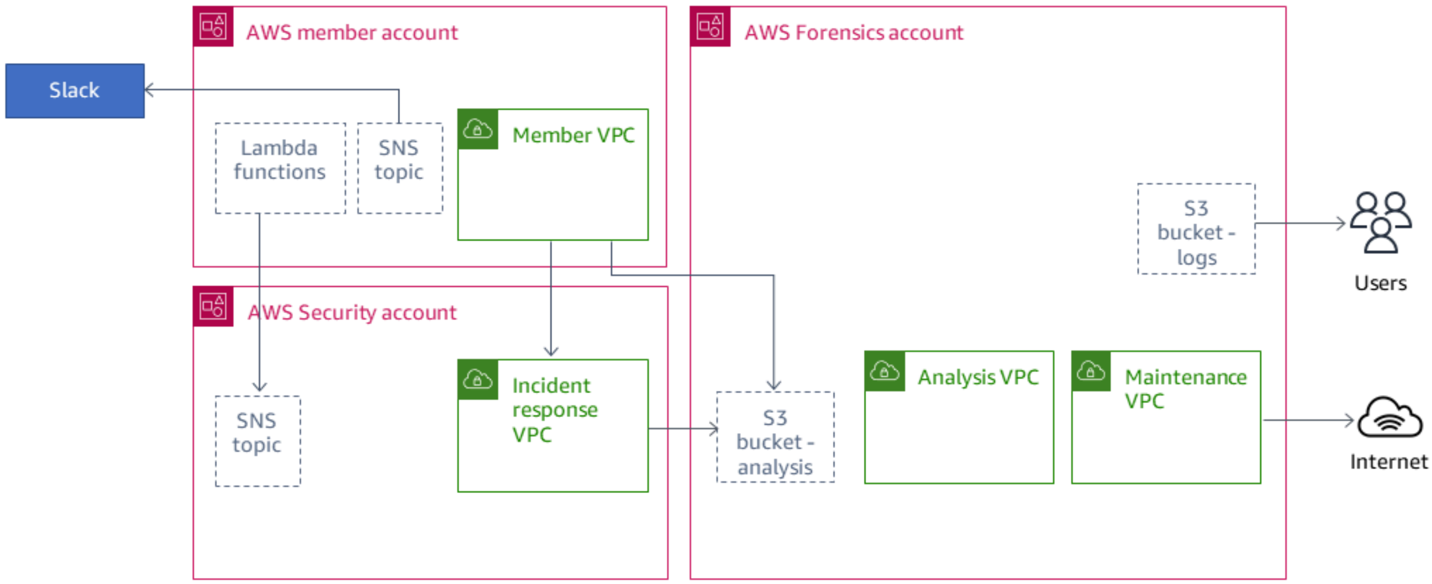
Target architecture

In addition to the member account, the target environment consists of two main accounts: a Security account and a Forensics account. Two accounts are used for the following reasons:

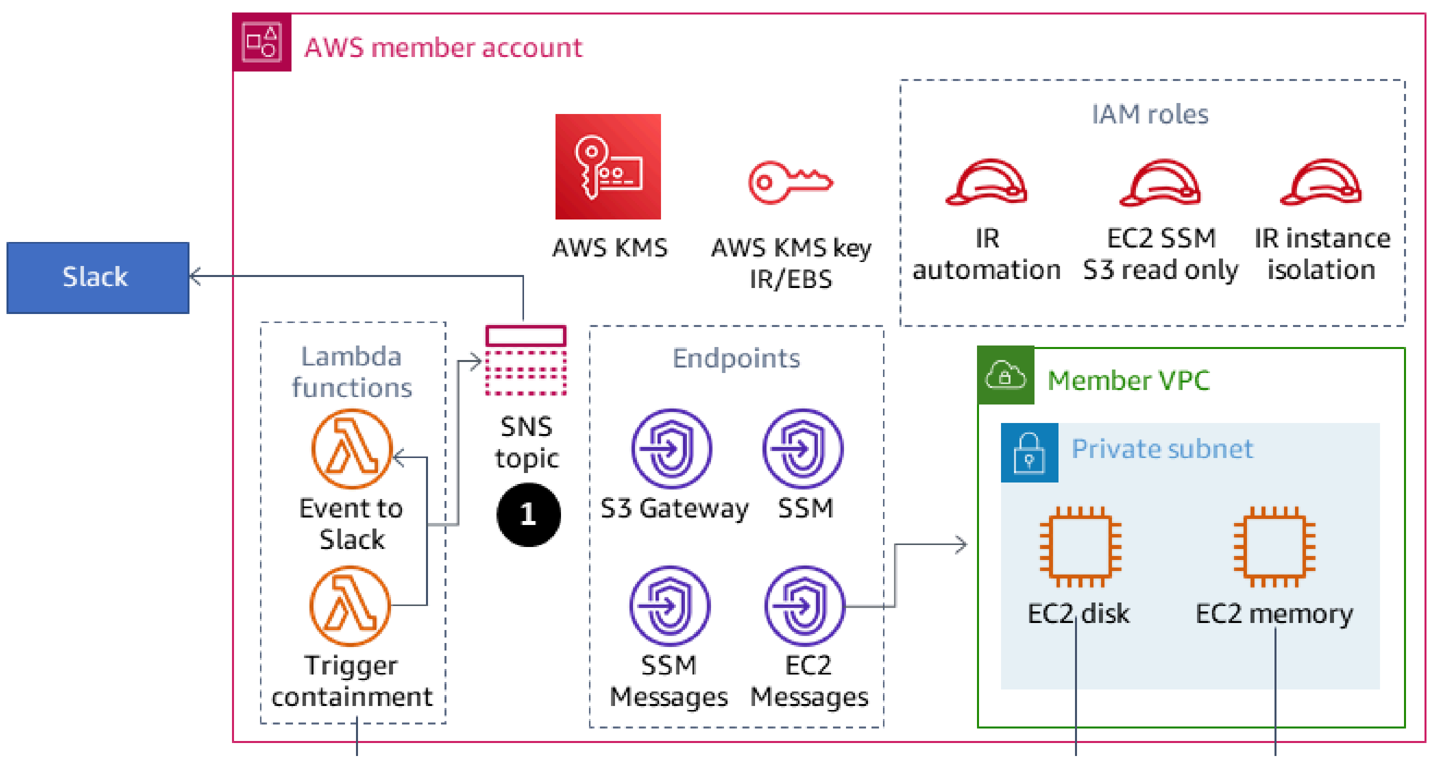
- To separate them from any other customer accounts to reduce blast radius in case of a failed forensic analysis
- To help ensure the isolation and protection of the integrity of the artifacts being analyzed
- To keep the investigation confidential
- To avoid situations where the threat actors might have used all the resources immediately available to your compromised AWS account by hitting service quotas and so preventing you from instantiating an Amazon EC2 instance to perform investigations.

Also, having separate Security and Forensics accounts allows for creating separate roles—a Responder for acquiring evidence and an Investigator for analyzing it. Each role would have access to its separate account.

The following diagram shows only the interaction between the accounts. Details of each account are shown in subsequent diagrams, and a complete diagram is attached.

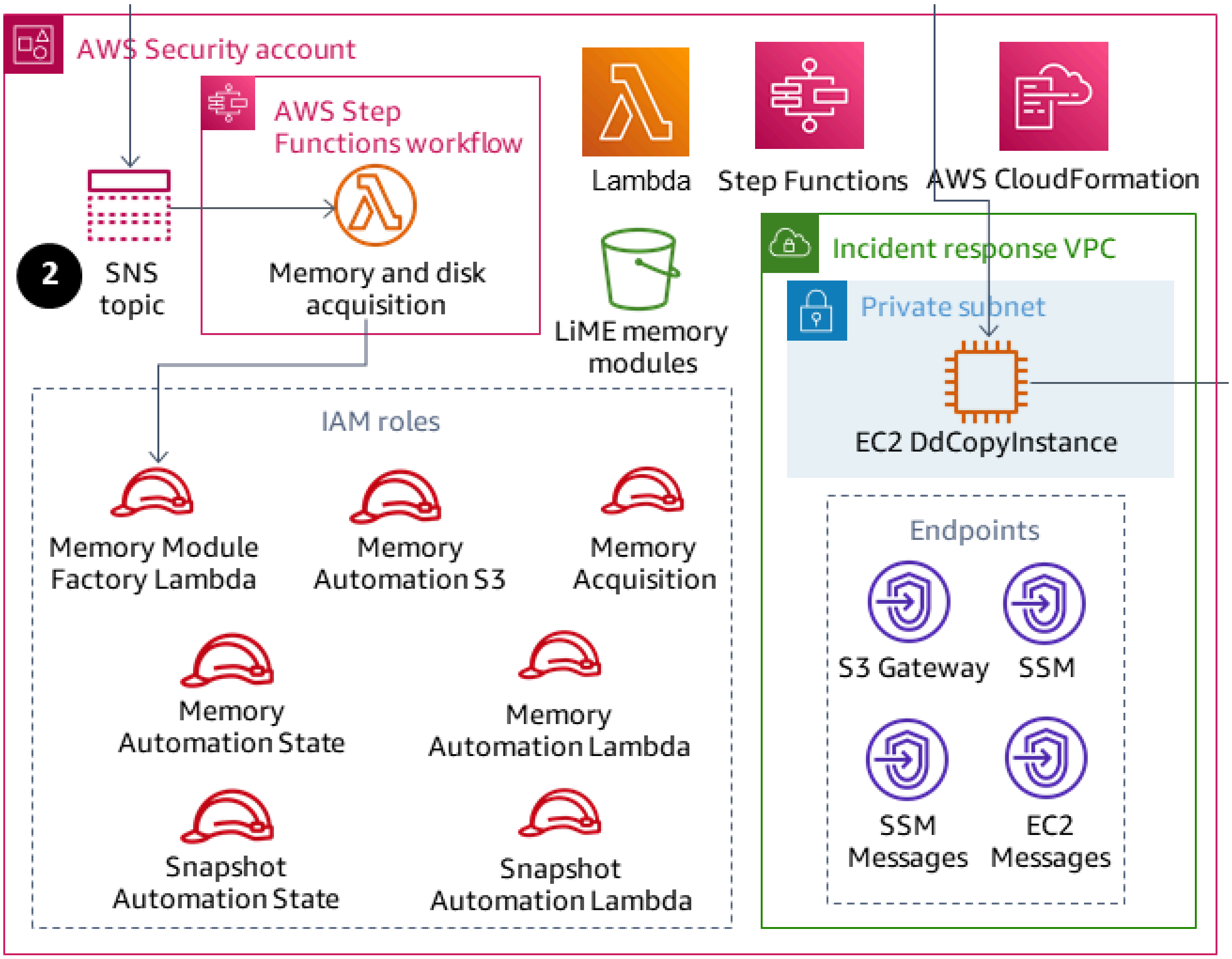


The following diagram shows the member account.



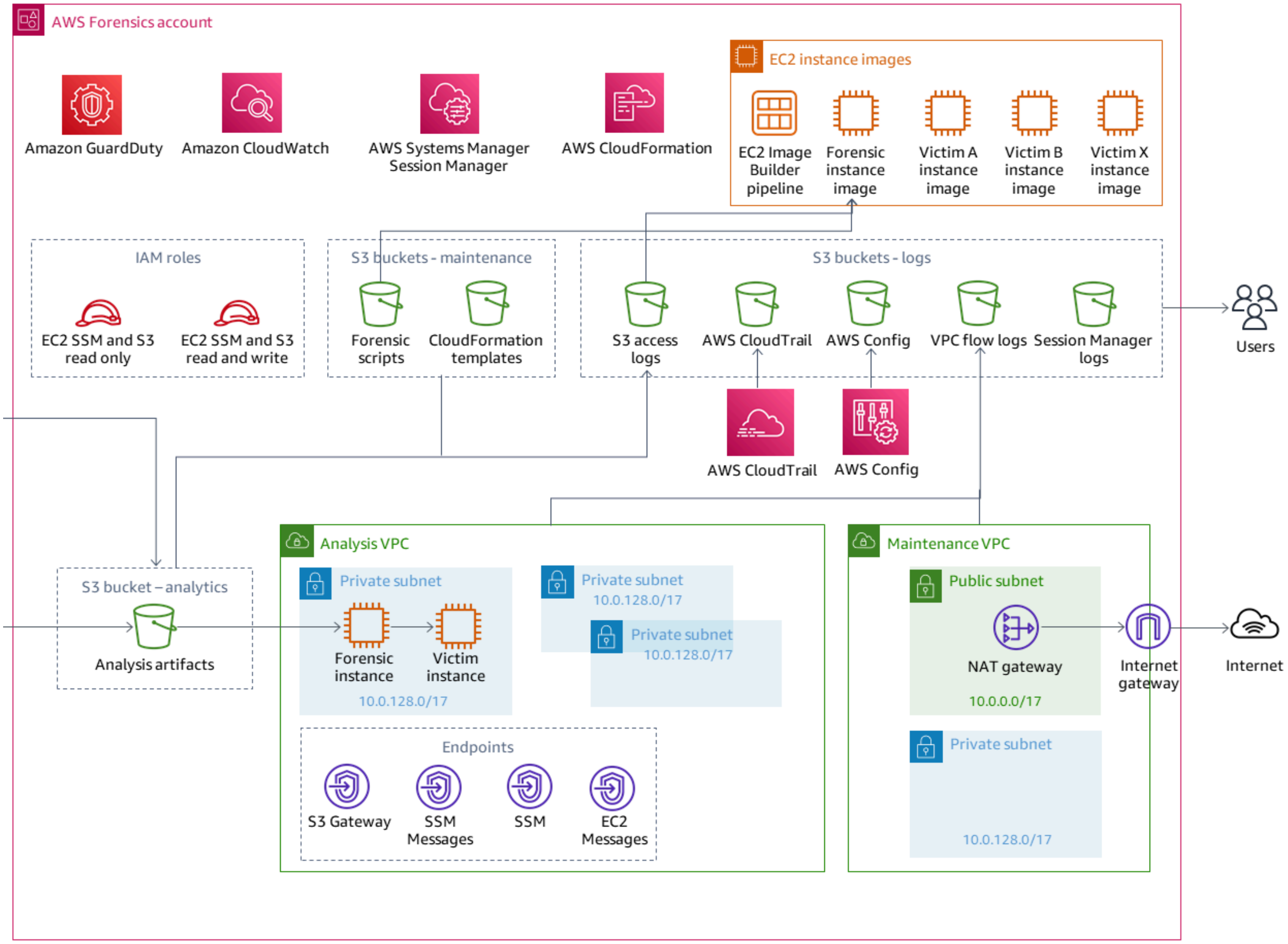
1. An event is sent to the Slack Amazon Simple Notification Service (Amazon SNS) topic.

The following diagram shows the Security account.



2. The Amazon SNS topic in the Security account initiates Forensics events.

The following diagram shows the Forensics account.



The Security account is where the two main AWS Step Functions workflows are created for memory and disk image acquisition. After the workflows are running, they access the member account that has the Amazon EC2 instances involved in an incident, and they initiate a set of Lambda functions that will gather a memory dump or a disk dump. Those artifacts are then stored in the Forensics account.

The Forensics account will hold the artifacts gathered by the Step Functions workflow in the Analysis artifacts Amazon S3 bucket. The Forensics account will also have an Amazon EC2 Image Builder pipeline that builds an Amazon Machine Image (AMI) of a Forensics instance. Currently, the image is based on SANS SIFT Workstation.

The build process uses the Maintenance VPC, which has connectivity to the internet. The image can be later used for spinning up the Amazon EC2 instance for analysis of the gathered artifacts in the Analysis VPC.

The Analysis VPC does not have internet connectivity. By default, the pattern creates three private analysis subnets. You can create up to 200 subnets, which is the quota for the number of subnets in a VPC, but the VPC endpoints need to have those subnets added for AWS Systems Manager Session Manager to automate running commands in them.

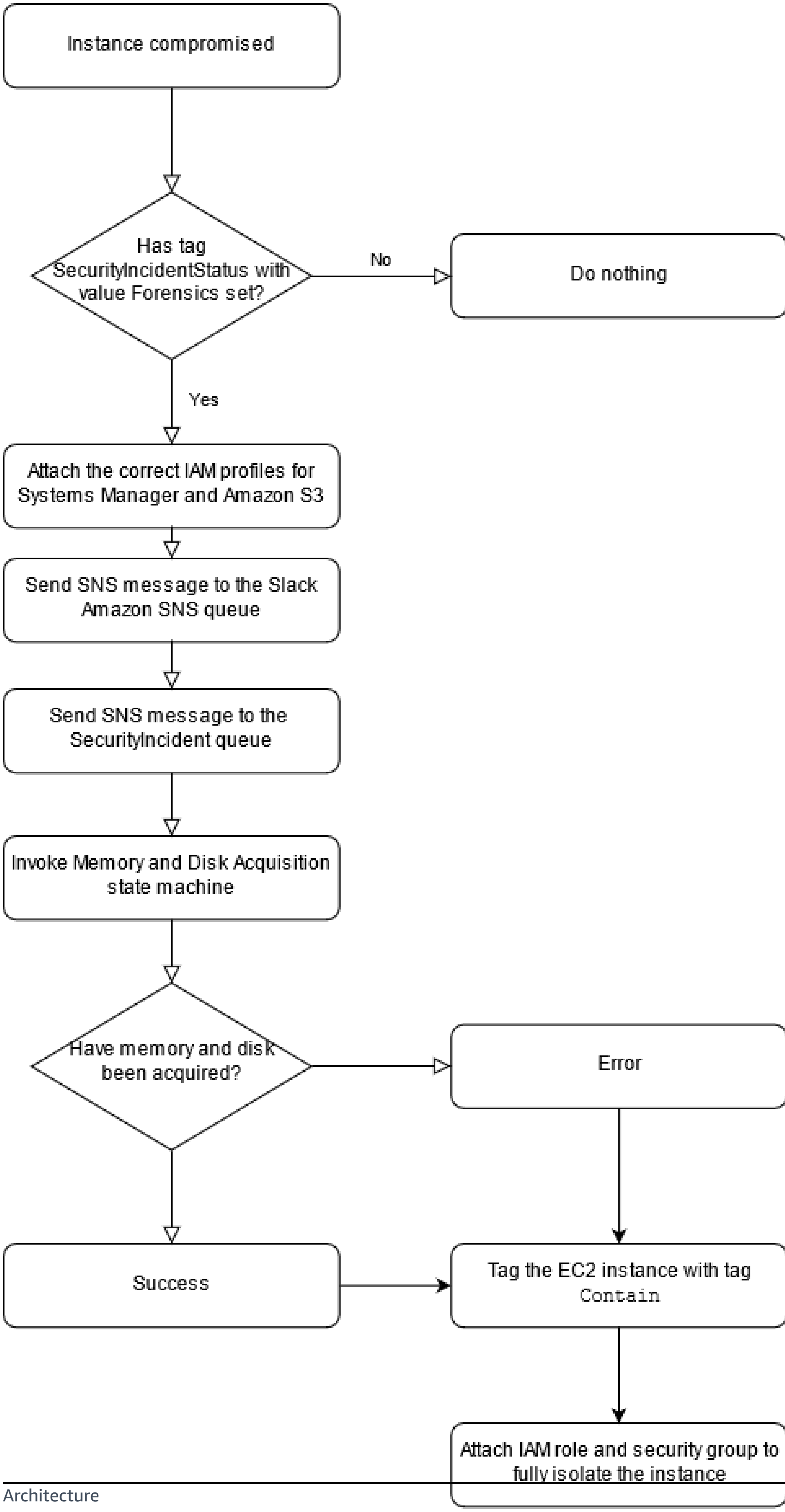
From a best-practices perspective, we recommend using AWS CloudTrail and AWS Config to do the following:

- Track changes made in your Forensics account
- Monitor access and integrity of the artifacts that are stored and analyzed

Workflow

The following diagram shows the key steps of a workflow that includes the process and decision tree from when an instance is compromised until it is analyzed and contained.

1. Has the `SecurityIncidentStatus`tag been set with the value Analyze? If yes, do the following:
   a. Attach the correct IAM profiles for AWS Systems Manager and Amazon S3.
   b. Send an Amazon SNS message to the Amazon SNS queue in Slack.
   c. Send an Amazon SNS message to the `SecurityIncident` queue.
   d. Invoke the Memory and Disk Acquisition state machine.
2. Have memory and disk been acquired? If no, there is an error.
3. Tag the Amazon EC2 instance with the `Contain` tag.
4. Attach the IAM role and security group to fully isolate the instance.


Instance compromised
Has tag SecurityIncidentStatus with value Forensics set?
No
Do nothing
Yes
Attach the correct IAM profiles for Systems Manager and Amazon S3
Send SNS message to the Slack Amazon SNS queue
Send SNS message to the SecurityIncident queue
Invoke Memory and Disk Acquisition state machine
Have memory and disk been acquired?
Error
Success
Tag the EC2 instance with tag Contain
Attach IAM role and security group to fully isolate the instance

Automation and scale

The intent of this pattern is to provide a scalable solution to perform incident response and forensics across several accounts within a single AWS Organizations organization.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Command Line Interface (AWS CLI) is an open source tool for interacting with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Security Hub provides a comprehensive view of your security state in AWS. It also helps you check your AWS environment against security industry standards and best practices.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.

Code

For the code and specific implementation and usage guidance, see the GitHub Automated Incident Response and Forensics Framework repository.

Epics

Deploy the CloudFormation templates

Task	Description	Skills required
Deploy CloudFormation templates.	The CloudFormation templates are marked 1 through 7 with the first word of the script name indicating in which account the template needs to be deployed. Note that the order of launching the CloudFormation templates is important. • `1-forensic-AnalysisVPCnS3Buckets.yaml` : Deployed in the forensics account. It creates the Amazon S3 buckets and the Analysis VPC, and it activates CloudTrail. • `2-forensic-MaintenanceVPCnEC2ImageBuilderPipeline.yaml` : Deploys the maintenance VPC and image builder pipeline based on SANS SIFT. • `3-security_IR-Disk_Mem_automation.yaml` : Deploys the functions in the security account that	AWS administrator

Task	Description	Skills required
	enable disk and memory acquisition. • `4-security_LiME_Volatility_Factory.yaml` : Initiates a build function to start creating the memory modules based on the given AMI IDs. Note that AMI IDs are different across AWS Regions. Whenever you need new memory modules, you can rerun this script with the new AMI IDs. Consider integrating this with your golden image AMI builder pipelines (if used in your environment). • `5-member-IR-automation.yaml` : Creates the member incident-response automation function, which initiates the incident-response process. It allows sharing Amazon Elastic Block Store (Amazon EBS) volumes across accounts, automated posting to Slack channels during the incident-response process, initiating the forensics process, and isolating the instances after the process finishes.	

Task	Description	Skills required
	• `6-forensic-artifact-s3-policies.yaml` : After all the scripts have been deployed this script fixes the permissions required for all the cross-account interactions. • `7-security-IR-vpc.yaml` : Configures a VPC used for incident response volume processing. To initiate the incident response framework for a specific Amazon EC2 instance, create a tag with the key `SecurityIncidentStatus` and the value `Analyze`. This will initiate the member Lambda function that will automatically start isolation and memory as well as disk acquisition.	

Task	Description	Skills required
Operate the framework.	The Lambda function will also retag the asset at the end (or on failure) with `Contain`. This initiates the containment, which fully isolates the instance with a no INBOUND/OUTBOUND security group and with an IAM role that disallows all access. Follow the steps in the [GitHub repository]().	AWS administrator

Deploy custom Security Hub actions

Task	Description	Skills required
Deploy the custom Security Hub actions by using a CloudFormation template.	To create a custom action so that you can use the dropdown list from Security Hub, deploy the `Modules/SecurityHub Custom Actions/SecurityHubCustomActions.yaml` CloudFormation template. Then modify the `IRAutomation` role in each of the member accounts to allow the Lambda function that runs the action to assume the `IRAutomation` role. For more information, see the [GitHub repository]().	AWS administrator

Related resources

- AWS Security Incident Response Guide

Additional information

By using this environment, a Security Operations Center (SOC) team can improve their security incident response process through the following:

- Having the ability to perform forensics in a segregated environment to avoid accidental compromise of production resources
- Having a standardized, repeatable, automated process to do containment and analysis.
- Giving any account owner or administrator the ability to initiate the incident-response process with the minimal knowledge of how to use tags
- Having a standardized, clean environment for performing incident analysis and forensics without the noise of a larger environment
- Having the ability to create multiple analysis environments in parallel
- Focusing SOC resources on incident response instead of on maintenance and documentation of a cloud forensics environment
- Moving away from a manual process toward an automated one to achieve scalability
- Using CloudFormation templates for consistency and to avoid repeatable tasks

Additionally, you avoid using persistent infrastructure, and you pay for resources when you need them.

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automate remediation for AWS Security Hub standard findings

Chandini Penmetsa and Aromal Raj Jayarajan, Amazon Web Services

Summary

With AWS Security Hub, you can enable checks for standard best practices such as the following:

- AWS Foundational Security Best Practices
- CIS AWS Foundations Benchmark
- Payment Card Industry Data Security Standard (PCI DSS)

Each of these standards has predefined controls. Security Hub checks for the control in a given AWS account and reports the findings.

AWS Security Hub sends all findings to Amazon EventBridge by default. This pattern provides a security control that deploys an EventBridge rule to identify AWS Foundational Security Best Practices standard findings. The rule identifies the following findings for automatic scaling, virtual private clouds (VPCs), Amazon Elastic Block Store (Amazon EBS), and Amazon Relational Database Service (Amazon RDS) from the AWS Foundational Security Best Practices standard:

- [AutoScaling.1] Auto Scaling groups associated with a load balancer should use load balancer health checks
- [EC2.2] The VPC default security group should not allow inbound and outbound traffic
- [EC2.6] VPC flow logging should be enabled in all VPCs
- [EC2.7] EBS default encryption should be enabled
- [RDS.1] RDS snapshots should be private
- [RDS.6] Enhanced monitoring should be configured for RDS DB instances and clusters
- [RDS.7] RDS clusters should have deletion protection enabled

The EventBridge rule forwards these findings to an AWS Lambda function, which remediates the finding. The Lambda function then sends a notification with remediation information to an Amazon Simple Notification Service (Amazon SNS) topic.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An email address where you want to receive the remediation notification
- Security Hub and AWS Config enabled in the AWS Region where you intend to deploy the control
- An Amazon Simple Storage Service (Amazon S3) bucket in same Region as the control to upload the AWS Lambda code

Limitations

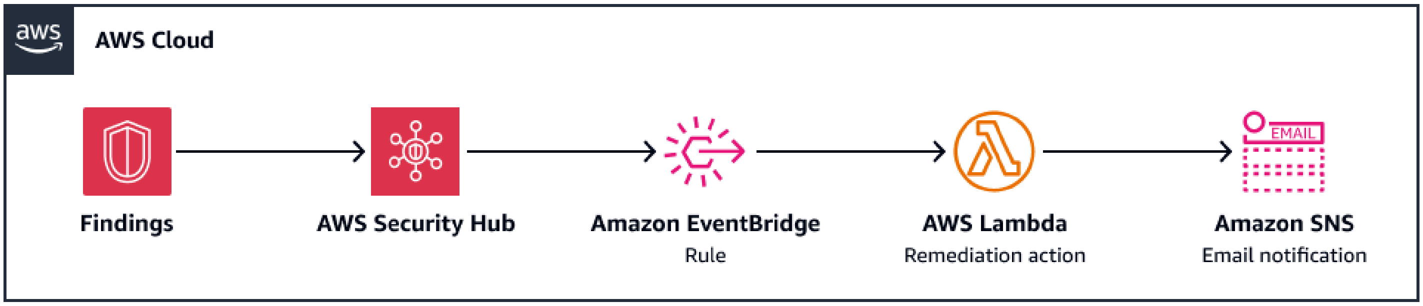
- This security control automatically remediates new findings reported after the security control deployment. To remediate existing findings, select the findings manually on the Security Hub console. Then, under **Actions**, select the **AFSBPRemedy** custom action that was created as part of the deployment by AWS CloudFormation.
- This security control is regional and must be deployed in the AWS Regions that you intend to monitor.
- For the EC2.6 remedy, to enable VPC Flow Logs, an Amazon CloudWatch Logs log group will be created with `/VpcFlowLogs/vpc_id` format. If a log group exists with same name, the existing log group will be used.
- For the EC2.7 remedy, to enable Amazon EBS default encryption, the default AWS Key Management Service (AWS KMS) key is used. This change prevents the use of certain instances that do not support encryption.

Architecture

Target technology stack

- Lambda function
- Amazon SNS topic
- EventBridge rule
- AWS Identity and Access Management (IAM) roles for the Lambda function, VPC Flow Logs, and Amazon RDS Enhanced Monitoring

Target architecture



Automation and scale

If you are using AWS Organizations, you can use AWS CloudFormation StackSets to deploy this template in multiple accounts that you want this to monitor.

Tools

- AWS CloudFormation is a service that helps you model and set up AWS resources by using infrastructure as code.
- Amazon EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, routing that data to targets such as Lambda functions.
- AWS Lambda supports running code without provisioning or managing servers.
- Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Best practices

- Nine AWS Security Hub best practices
- AWS Foundational Security Best Practices standard

Epics

Deploy the security control

Task	Description	Skills required
Define the Amazon S3 bucket.	On the Amazon S3 console, choose or create an Amazon S3 bucket with a unique name that does not contain leading slashes. An Amazon S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your Amazon S3 bucket must be in the same Region as the Security Hub findings that are being evaluated.	Cloud Architect
Upload the Lambda code to the Amazon S3 bucket.	Upload the Lambda code .zip file that's provided in the "Attachments" section to the defined Amazon S3 bucket.	Cloud Architect
Deploy the AWS CloudFormation template.	Deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters.	Cloud Architect

Complete the parameters in the AWS CloudFormation template

Task	Description	Skills required
Provide the Amazon S3 bucket name.	Enter the name of the Amazon S3 bucket that you created in the first epic.	Cloud Architect
Provide the Amazon S3 prefix.	Provide the location of the Lambda code .zip file in your Amazon S3 bucket, without leading slashes (for example, `<directory>/<file-name>.zip`).	Cloud Architect
Provide the ARN of the Amazon SNS topic.	If you want to use an existing Amazon SNS topic for remediation notifications, provide the Amazon Resource Name (ARN) of the Amazon SNS topic. If you want to use a new Amazon SNS topic, keep the value as None (the default value).	Cloud Architect
Provide an email address.	Provide an email address where you want to receive the remediation notifications (needed only when you want AWS CloudFormation to create the Amazon SNS topic).	Cloud Architect
Define the logging level.	Define the logging level and frequency for your Lambda function. `Info` designates detailed informational messages on the application's	Cloud Architect

Task	Description	Skills required
	progress. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	
Provide the ARN of the IAM role for VPC Flow Logs.	Provide the ARN of the IAM role to be used for VPC Flow Logs. If you enter `None`, AWS CloudFormation creates an IAM role and uses it.	Cloud Architect
Provide the ARN of the IAM role for Amazon RDS Enhanced Monitoring.	Provide the ARN of the IAM role to be used for Amazon RDS Enhanced Monitoring. If you enter `None`, AWS CloudFormation creates an IAM role and uses it.	Cloud Architect

Confirm the subscription

Task	Description	Skills required
Confirm the Amazon SNS subscription.	When the template successfully deploys, if a new Amazon SNS topic was created, a subscription message is sent to the email address that you provided. To receive remediation notifications, you must confirm this subscription email message.	Cloud Architect

Related resources

- [Creating a stack on the AWS CloudFormation console]
- [AWS Lambda website]
- [AWS Security Hub documentation]

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automate security scans for cross-account workloads using Amazon Inspector and AWS Security Hub

Ramya Pulipaka and Mikesh Khanal, Amazon Web Services

Summary

This pattern describes how to automatically scan for vulnerabilities in cross-account workloads on the Amazon Web Services (AWS) Cloud.

The pattern helps create a schedule for host-based scans of Amazon Elastic Compute Cloud (Amazon EC2) instances that are grouped by tags or for network-based Amazon Inspector scans. An AWS CloudFormation stack deploys all the required AWS resources and services to your AWS accounts.

The Amazon Inspector findings are exported to AWS Security Hub and provide insights into vulnerabilities across your accounts, AWS Regions, virtual private clouds (VPCs), and Amazon EC2 instances. You can receive these findings by email or you can create an Amazon Simple Notification Service (Amazon SNS) topic that uses an HTTP endpoint to send the findings to ticketing tools, security information and event management (SIEM) software, or other third-party security solutions.

Prerequisites and limitations

Prerequisites

- Active AWS accounts that host cross-account workloads, including a central audit account.
- An existing email address to receive email notifications from Amazon SNS.
- An existing HTTP endpoint used by ticketing tools, SIEM software, or other third-party security solutions.
- Security Hub, enabled and configured. You can use this pattern without Security Hub, but we recommend using Security Hub because of the insights it generates. For more information, see [Setting up Security Hub] in the Security Hub documentation.
- An Amazon Inspector agent must be installed on each EC2 instance that you want to scan. You can install the Amazon Inspector agent on multiple EC2 instances by using [AWS Systems Manager Run Command].

Skills

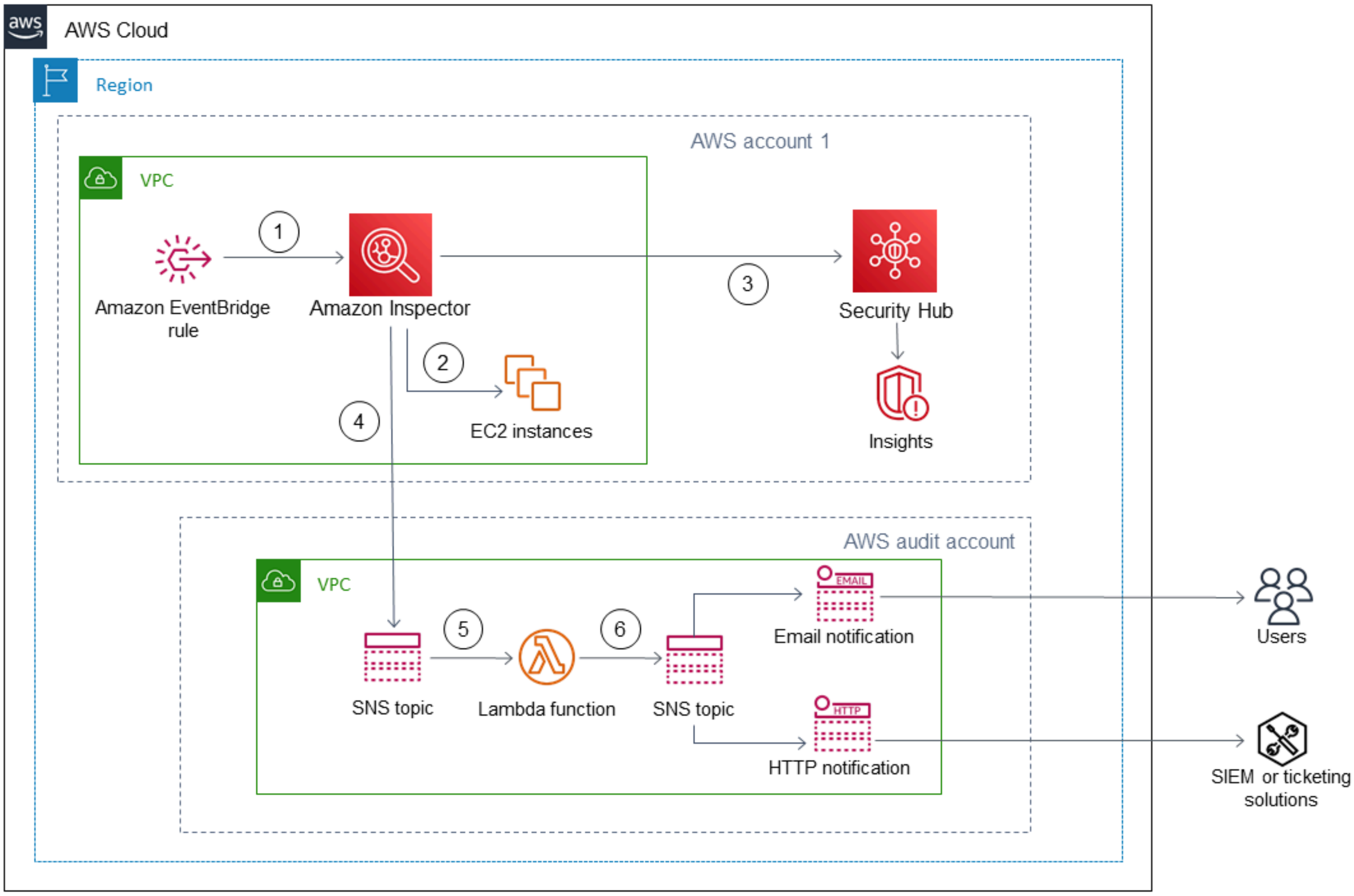
- Experience using `self-managed` and `service-managed` permissions for stack sets in CloudFormation. If you want to use `self-managed` permissions to deploy stack instances to specific accounts in specific Regions, you must create the required AWS Identity and Access Management (IAM) roles. If you want to use `service-managed` permissions to deploy stack instances to accounts managed by AWS Organizations in specific Regions, you don't need to create the required IAM roles. For more information, see [Create a stack set](#) in the CloudFormation documentation.

Limitations

- If no tags are applied to Amazon EC2 instances in an account, then Amazon Inspector scans all the instances in that account.
- The CloudFormation stack sets and the `onboard-audit-account.yaml` file (attached) must be deployed in the same Region.
- This pattern's approach can scale under the publish quota of 30,000 transactions per second (TPS) for an Amazon SNS topic in the US East (N. Virginia) Region (`us-east-1`), although limits vary by Region. To scale more effectively and avoid data loss, we recommend using Amazon Simple Queue Service (Amazon SQS) in front of the Amazon SNS topic.

Architecture

The following diagram illustrates the workflow for automatically scanning Amazon EC2 instances.



The workflow consists of the following steps:

1. An Amazon EventBridge rule uses a cron expression to self-initiate on a specific schedule and initiates Amazon Inspector.
2. Amazon Inspector scans the tagged Amazon EC2 instances in the account.
3. Amazon Inspector sends the findings to Security Hub, which generates insights for workflow, prioritization, and remediation.
4. Amazon Inspector also sends the assessment's status to an Amazon SNS topic in the audit account. An AWS Lambda function is invoked if a `findings reported` event is published to the Amazon SNS topic.
5. The Lambda function fetches, formats, and sends the findings to another Amazon SNS topic in the audit account.
6. Findings are sent to the email addresses that are subscribed to the Amazon SNS topic. The full details and recommendations are sent in JSON format to the subscribed HTTP endpoint.

Tools

- [AWS CloudFormation] helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications.
- AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and Regions with a single operation.
- AWS Control Tower creates an abstraction or orchestration layer that combines and integrates the capabilities of several other AWS services, including AWS Organizations.
- Amazon EventBridge is a serverless event bus service that makes it easy to connect your applications with data from a variety of sources.
- AWS Lambda is a compute service that helps you run code without provisioning or managing servers.
- AWS Security Hub provides you with a comprehensive view of your security state in AWS and helps you check your environment against security industry standards and best practices.
- Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers.

Epics

Deploy the CloudFormation template

Task	Description	Skills required
Deploy the CloudFormation template in the audit account.	Download and save the `onboard-audit-account.yaml` file (attached) to a local path on your computer. Sign in to the AWS Management Console for your audit account, open the CloudFormation console, and then choose **Create stack**.	Developer, Security engineer

Task	Description	Skills required
	Choose **Prepare template** in the **Prerequisites** section, and then choose **Template is ready**. Choose **Template source** in the **Specify template** section, and then choose **Template is ready**. Upload the `onboard-audit-account.yaml` file and then configure the remaining options according to your requirements. Make sure that you configure the following input parameters: • `DestinationEmailAddress` – Enter an email address to receive findings. • `HTTPEndpoint` – Provide an HTTP endpoint for your ticketing or SIEM tools. **Note** You can also deploy the CloudFormation template by using AWS Command Line Interface (AWS CLI). For more information about this, see [Creating a stack](#) in	

Task	Description	Skills required
	the CloudFormation documentation.	
Confirm the Amazon SNS subscription.	Open your email inbox and choose **Confirm subscription** in the email that you receive from Amazon SNS. This opens a web browser window and displays the subscription confirmation.	Developer, Security engineer

Create CloudFormation stack sets to automate the Amazon Inspector scan schedule

Task	Description	Skills required
Create stack sets in the audit account.	Download the `vulnerability-management-program.yaml` file (attached) to a local path on your computer. On the [CloudFormation console](https://console.aws.amazon.com/cloudformation/), choose **View stacksets** and then choose **Create StackSet**. Choose **Template is ready**, choose **Upload a template file**, and then upload the `vulnerability-management-program.yaml` file. If you want to use `self-managed` permissions,	Developer, Security engineer

Task	Description	Skills required
	follow the instructions from Create a stack set with self-managed permissions in the CloudFormation documentation. This creates stack sets in individual accounts. If you want to use `service-managed` permissions, follow the instructions from Create a stack set with service-managed permissions in the CloudFormation documentation. This creates stack sets in your entire organization or specified organizational units (OUs). Make sure that the following input parameters are configured for your stack sets: • `AssessmentSchedule` – The schedule for EventBridge using cron expressions. • `Duration` – The duration of the Amazon Inspector assessment run in seconds. • `CentralSNSTopicArn` – The Amazon Resource Name (ARN) for the central Amazon SNS topic.	

Task	Description	Skills required
	• `Tagkey` – The tag key that is associated with the resource group. • `Tagvalue` – The tag value that is associated with the resource group. If you want to scan Amazon EC2 instances in the audit account, you must run the `vulnerability-management-program.yaml` file as a CloudFormation stack in the audit account.	
Validate the solution.	Check that you receive findings by email or HTTP endpoint on the schedule that you specified for Amazon Inspector.	Developer, Security engineer

Related resources

- Scale your security vulnerability testing with Amazon Inspector (AWS blog post)
- Automatically remediate Amazon Inspector security findings (AWS blog post)
- How to simplify security assessment setup by using Amazon EC2, AWS Systems Manager, and Amazon Inspector (AWS blog post)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automatically audit AWS security groups that allow access from public IP addresses

Eugene Shifer and Stephen DiCato, Amazon Web Services

Summary

As a security best practice, it's crucial to minimize the exposure of AWS resources to only what is absolutely necessary. For example, web servers that serve the general public need to allow inbound access from the internet, but access to other workloads should be restricted to specific networks to reduce unnecessary exposure. Security groups in Amazon Virtual Private Cloud (Amazon VPC) are an effective control to help you limit resource access. However, evaluating security groups can be a cumbersome task, especially in multi-account architectures. AWS Config rules and AWS Security Hub controls can help you identify security groups that permit access from the public internet (0.0.0.0/0) to specific network communication protocols, such as Secure Shell (SSH), HTTP, HTTPS, and Windows remote desktop protocol (RDP). However, these rules and controls are not applicable if services run on non-standard ports or if access is restricted to certain public IP addresses. For instance, this might occur when a web service is associated with TCP port 8443 instead of the standard TCP port 443. This might also occur when developers have access to the server from their home networks, such as for testing purposes.

To address this, you can use the infrastructure as code (IaC) solution provided in this pattern to identify security groups that allow access from any non-private (RFC 1918 noncompliant) IP addresses to any workload in your AWS account or AWS organization. The AWS CloudFormation template provisions a custom AWS Config rule, an AWS Lambda function, and the necessary permissions. You can deploy it as a stack in a single account or as a stack set across the entire organization, managed through AWS Organizations.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Experience using GitHub
- If you're deploying into a single AWS account:
  - Permissions to create CloudFormation stacks

  - AWS Config set up in the target account
  - (Optional) Security Hub set up in the target account
- If you're deploying into an AWS organization:
  - Permissions to create CloudFormation stack sets
  - Security Hub set up with AWS Organizations integration
  - AWS Config set up in the accounts where you are deploying this solution
  - Designate an AWS account to be the delegated administrator for AWS Config and Security Hub

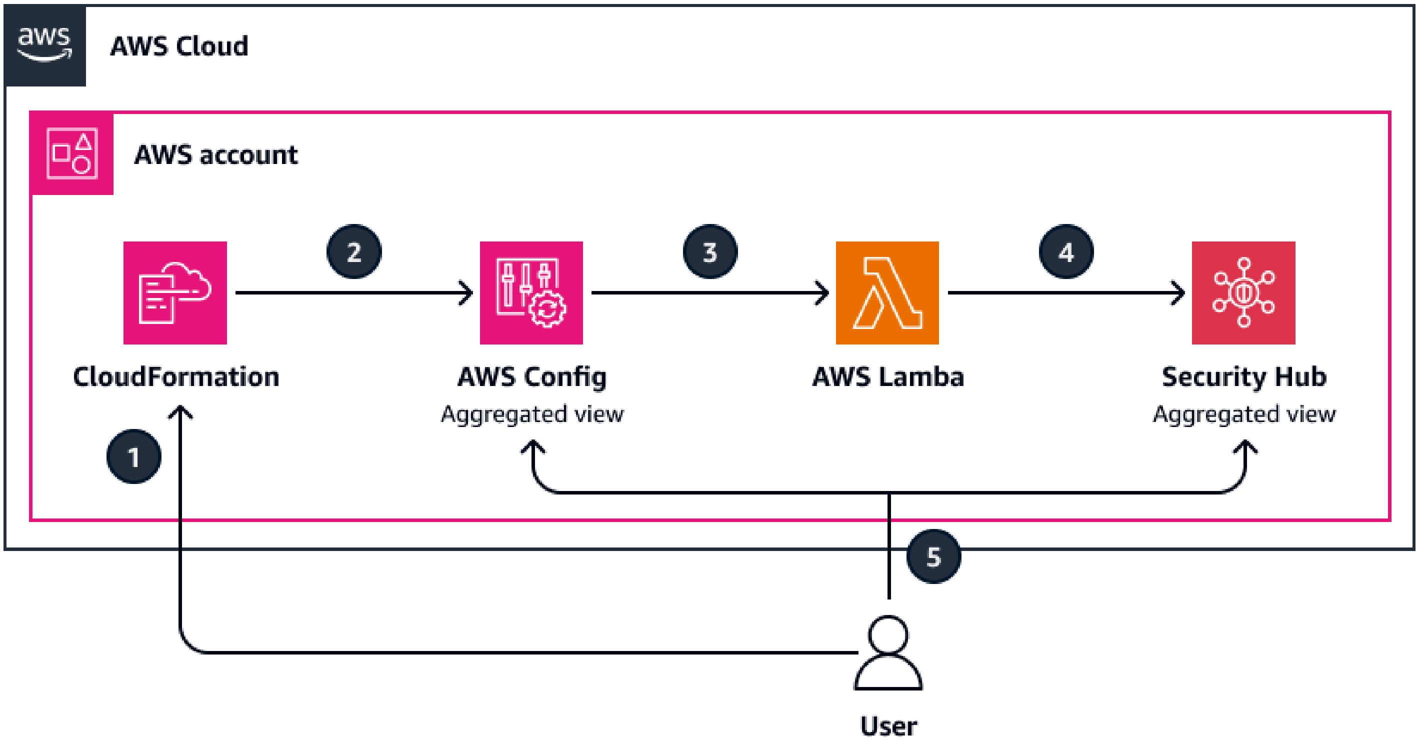
Limitations

- If you're deploying to an individual account that doesn't have Security Hub enabled, you can use AWS Config to evaluate the findings.
- If you're deploying to an organization that doesn't have a delegated administrator for AWS Config and Security Hub, you must log into the individual member accounts to view the findings.
- If you use AWS Control Tower to manage and govern the accounts in your organization, deploy the IaC in this pattern by using Customizations for AWS Control Tower (CfCT). Using the CloudFormation console would create configuration drift from AWS Control Tower guardrails and require that you re-enroll the organizational units (OUs) or managed accounts.
- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

Deploying into an individual AWS account

The following architecture diagram shows the deployment of the AWS resources within a single AWS account. You provision the resources by using a CloudFormation template directly through the CloudFormation console. If Security Hub is enabled, you can view the results in either AWS Config or Security Hub. If Security Hub is not enabled, you can view the results only in AWS Config.

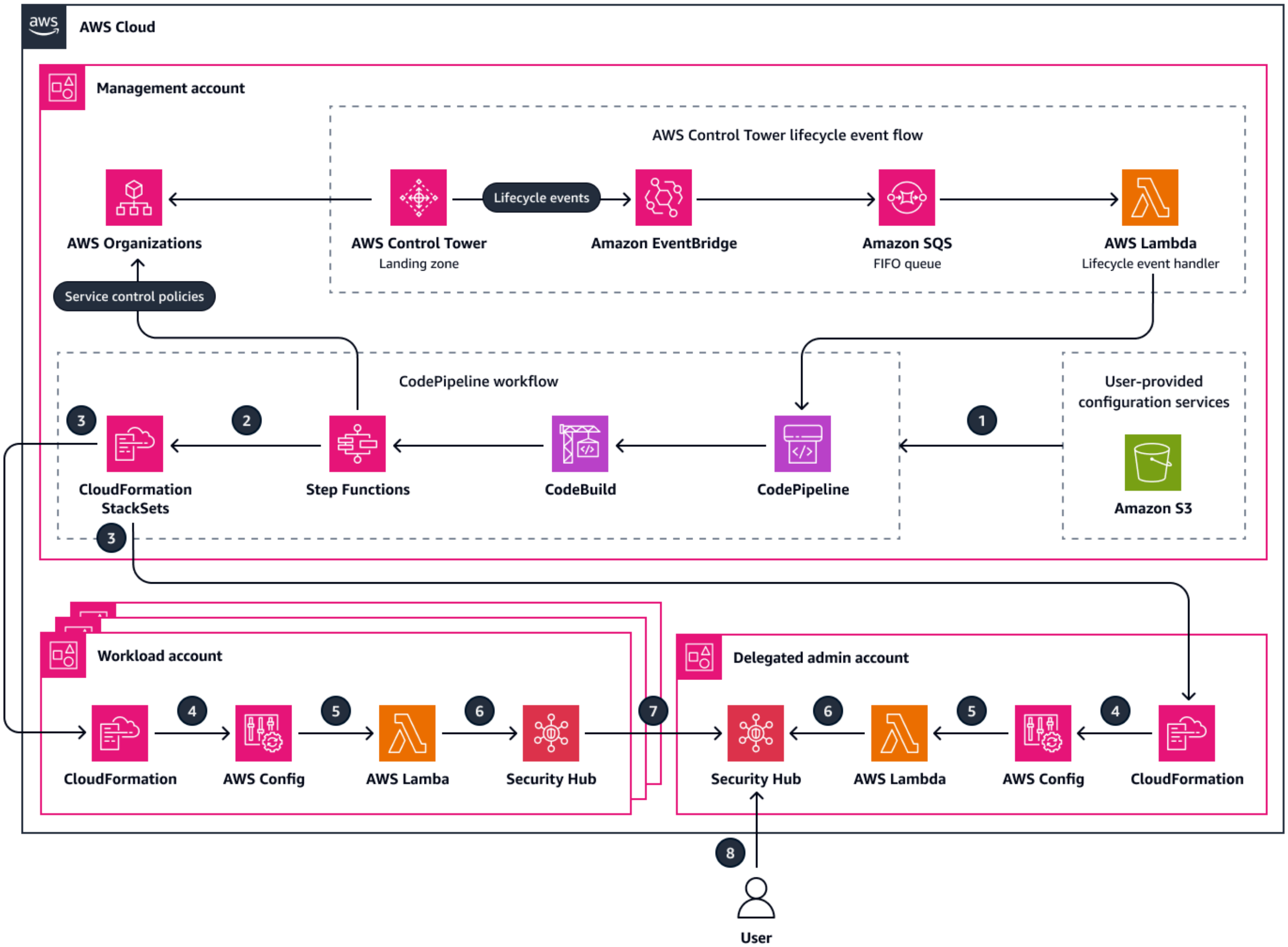


The diagram shows the following workflow:

1. You create a CloudFormation stack. This deploys a Lambda function and an AWS Config rule. Both the rule and function are set up with the AWS Identity and Access Management (IAM) permissions that are required to publish resource evaluations in AWS Config and logs.
2. The AWS Config rule operates in [detective evaluation mode](#) and invokes the Lambda function every 24 hours.
3. The Lambda function assesses the security groups and sends updates to AWS Config.
4. Security Hub receives all of the AWS Config findings.
5. You can view the findings in Security Hub or in AWS Config, depending on the services that you have set up in the account.

Deploying into an AWS organization

The following diagram shows deployment of the pattern across multiple accounts that are managed through AWS Organizations and AWS Control Tower. You deploy the CloudFormation template through CfCT. The assessment outcomes are centralized in Security Hub in the delegated administrator account. The AWS CodePipeline workflow section of the diagram shows the background steps that occur during CfCT deployment.



The diagram shows the following workflow:

1. In the management account, you upload a compressed (ZIP) file of the IaC template to an Amazon Simple Storage Service (Amazon S3) bucket that is deployed by CfCT.
2. The CfCT pipeline unzips the file, runs cfn-nag (GitHub) checks, and deploys the template as a CloudFormation stack set.
3. Depending on the configuration you specify in the CfCT manifest file, CloudFormation StackSets deploys stacks into individual accounts or specified OUs. This deploys a Lambda function and an AWS Config rule in the target accounts. Both the rule and function are set up with the IAM permissions that are required to publish resource evaluations in AWS Config and logs.
4. The AWS Config rule operates in detective evaluation mode and invokes the Lambda function every 24 hours.
5. The Lambda function assesses the security groups and sends updates to AWS Config.

6. AWS Config forwards all of the findings to Security Hub.
7. The Security Hub findings are aggregated in the delegated administrator account.
8. You can view the aggregated findings in Security Hub in the delegated administrator account.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time. An AWS Config rule defines your ideal configuration settings for a resource, and AWS Config can evaluate whether your AWS resources comply with the conditions in the rule.
- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices. Customizations for AWS Control Tower (CfCT) helps you customize your AWS Control Tower landing zone and stay aligned with AWS best practices. Customizations for this solution are implemented through CloudFormation templates and AWS Organizations service control policies (SCPs).
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Security Hub provides a comprehensive view of your security state in AWS. It also helps you check your AWS environment against security industry standards and best practices.

Other tools

- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub Detect vulnerable security groups repository.

Best practices

We recommend that you adhere to the best practices in the following resources:

- Best Practices for Organizational Units with AWS Organizations (AWS Cloud Operations & Migrations Blog)
- Guidance for Establishing an Initial Foundation using AWS Control Tower on AWS (AWS Solutions Library)
- Guidance for creating and modifying AWS Control Tower resources (AWS Control Tower documentation)
- CfCT deployment considerations (AWS Control Tower documentation)
- Apply least-privilege permissions (IAM documentation)

Epics

Review the CloudFormation template

Task	Description	Skills required
Determine your deployment strategy.	Review the solution and code to determine the deploymen t strategy for your AWS environment. Determine whether you are deploying into a single account or an AWS organization.	App owner, General AWS
Clone the repository.	Enter the following command to clone the Detect vulnerabl e security groups repository: `git clone https://github.com/aws-samples/detect-public-security-groups.git`	App developer, App owner

Task	Description	Skills required
Validate the Python version.	1. Navigate into the top-level directory in the cloned repository: `cd detect-public-security-groups` 2. Open **Security-Group-Public-Assessment.yaml**. 3. In the `SgPublicAccessCheckLambdaFunction` resource, confirm that the Python version is compatible with your target AWS Regions. By default, this function uses Python 3.12. For more information, see [AWS Lambda adds support for Python 3.12](). If necessary, update the Python version. 4. Save and close **Security-Group-Public-Assessment.yaml**.	AWS administrator, App developer

Deploy the CloudFormation template

Task	Description	Skills required
Deploy the CloudFormation template.	Deploy the CloudFormation template into your AWS environment. Do one of the following:	App developer, AWS administrator, General AWS

Task	Description	Skills required
	• If you're deploying to a single AWS account, follow the instructions in Creating a stack. • If you're deploying to an organization that isn't managed by AWS Control Tower, follow the instructions in Create a stack set. • If you're deploying to an organization that is managed by AWS Control Tower, see the instructions in the Build your own customizations.	
Verify the deployment.	In the CloudFormation console, verify that the stack or stack set has deployed successfully.	AWS administrator, App owner

Review the findings

Task	Description	Skills required
View the AWS Config rule findings.	In Security Hub, do the following to view a list of individual findings: 1. Open the Security Hub console. 2. In the navigation pane, choose **Findings**.	AWS administrator, AWS systems administrator, Cloud administrator

Task	Description	Skills required
	3. In the **Add filters** box, add the following filters: • **Compliance Status** is `FAILED` • **Title** is `SgPublicAccessCheck` 4. Choose **Apply**. In Security Hub, do the following to view a list of total findings grouped by AWS account: 1. Open the [Security Hub console](https://console.aws.amazon.com/securityhub/). 2. In the navigation pane, choose **Insights**. 3. Choose **Create insight**. 4. To select the grouping attribute for the insight: a. Choose the search box to display the filter options. b. Choose **Group by**. c. Select **AwsAccountId**. d. Choose **Apply**. 5. In the **Add filters** box, add the following filters: • **Title** is `SgPublicAccessCheck` • **Compliance Status** is `FAILED`	

Task	Description	Skills required
	6. Choose **Create insight**. 7. Enter an **Insight name**, and then choose **Create insight**. In AWS Config, to view a list of findings, follow the instructions in [Viewing Compliance Information and Evaluation Results]() in the AWS Config documentation.	

Troubleshooting

Issue	Solution
The CloudFormation stack set creation or deletion fails.	When AWS Control Tower is deployed, it enforces necessary guardrails and assumes control over AWS Config aggregators and rules. This includes preventing any direct alterations through CloudFormation. To properly deploy or remove this CloudFormation template, including all associated resources, you must use CfCT.
CfCT fails to delete the CloudFormation template.	If the CloudFormation template persists even after making necessary changes in the manifest file and removing the template files, confirm that the manifest file contains the `enable_stack_set_deletion` parameter and that the value is set to `false`. For more information, see [Delete a stack set]() in the CfCT documentation.

Related resources

- AWS Config Custom Rules (AWS Config documentation)

Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config

Manigandan Shri, Amazon Web Services

Summary

Visibility over activity in your Amazon Web Services (AWS) account is an important security and operational best practice. AWS CloudTrail helps you with the governance, compliance, and operational and risk auditing of your account.

To ensure that CloudTrail remains enabled in your account, AWS Config provides the `cloudtrail-enabled` managed rule. If CloudTrail is turned off, the `cloudtrail-enabled` rule automatically re-enables it by using automatic remediation.

However, you must make sure that you follow security best practices for CloudTrail if you use automatic remediation. These best practices include enabling CloudTrail in all AWS Regions, logging read and write workloads, enabling insights, and encrypting log files with server-side encryption using AWS Key Management Service (AWS KMS) managed keys (SSE-KMS).

This pattern helps you follow these security best practices by providing a custom remediation action to automatically re-enable CloudTrail in your account.

Important

We recommend using service control policies (SCPs) to prevent any tampering with CloudTrail. For more information about this, see the *Prevent tampering with AWS CloudTrail* section of How to use AWS Organizations to simplify security at enormous scale on the AWS Security Blog.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Permissions to create an AWS Systems Manager Automation runbook

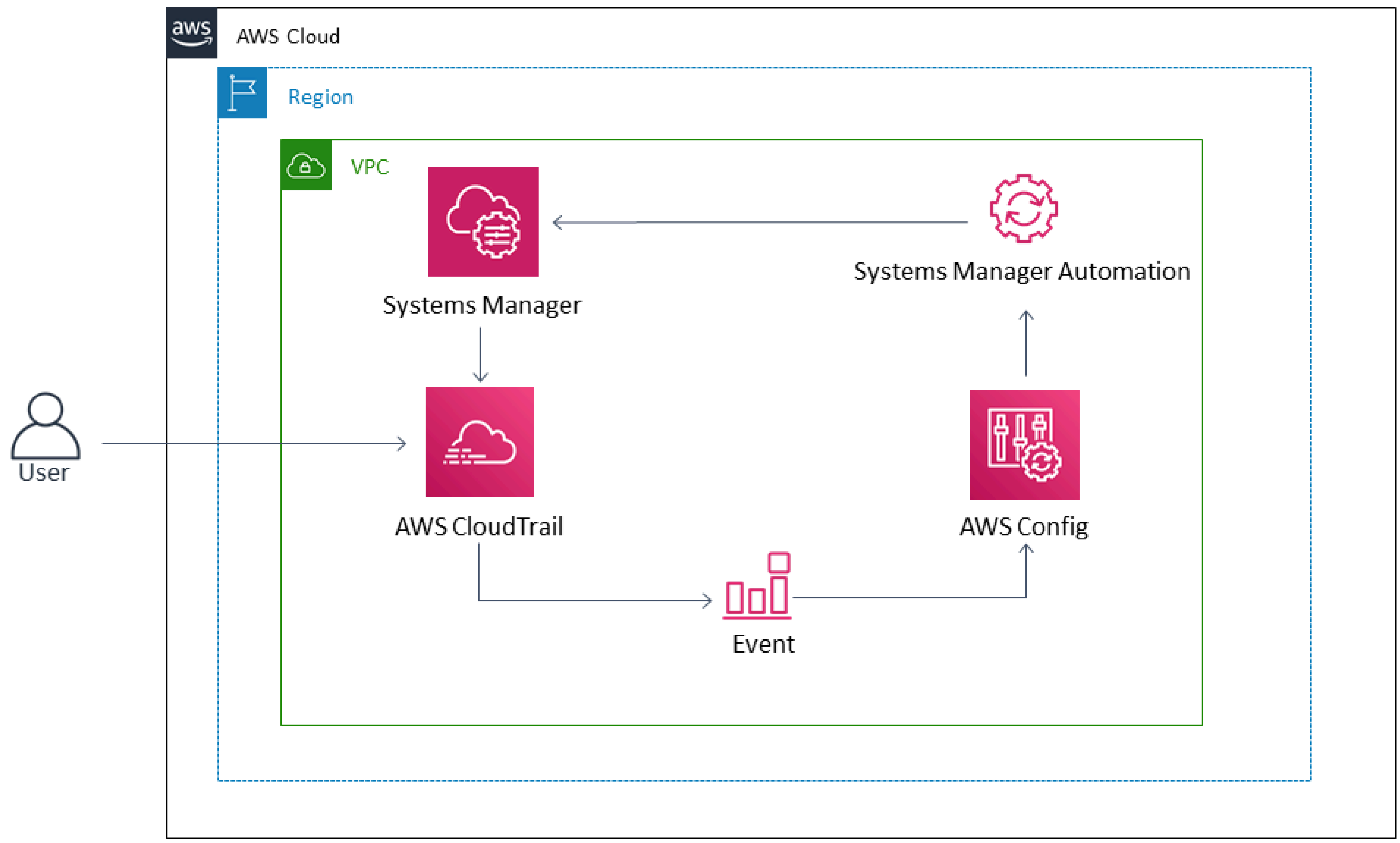
- An existing trail for your account

Limitations

This pattern doesn't support the following actions:

- Setting an Amazon Simple Storage Service (Amazon S3) prefix key for the storage location
- Publishing to an Amazon Simple Notification Service (Amazon SNS) topic
- Configuring Amazon CloudWatch Logs to monitor your CloudTrail logs

Architecture



Technology stack

- AWS Config
- CloudTrail
- Systems Manager

- Systems Manager Automation

Tools

- AWS Config provides a detailed view of the configuration of AWS resources in your account.
- AWS CloudTrail helps you enable governance, compliance, and operational and risk auditing of your account.
- AWS Key Management Service (AWS KMS) is an encryption and key management service.
- AWS Systems Manager helps you view and control your infrastructure on AWS.
- AWS Systems Manager Automation simplifies common maintenance and deployment tasks of Amazon Elastic Compute Cloud (Amazon EC2) instances and other AWS resources.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code

The **cloudtrail-remediation-action.yml** file (attached) helps you create a Systems Manager Automation runbook to set up and re-enable CloudTrail using security best practices.

Epics

Configure CloudTrail

Task	Description	Skills required
Create an S3 bucket.	Sign in to the AWS Management Console, open the Amazon S3 console, and then create an S3 bucket to store the CloudTrail logs. For more information, see Create an S3 bucket in the Amazon S3 documentation.	Systems administrator

Task	Description	Skills required
Add a bucket policy to allow CloudTrail to deliver log files to the S3 bucket.	CloudTrail must have the required permissions to deliver log files to your S3 bucket. On the Amazon S3 console, choose the S3 bucket that you created earlier and then choose **Permissions**. Create an S3 bucket policy by using the Amazon S3 bucket policy for CloudTrail from the CloudTrail documentation. For steps on how to add a policy to an S3 bucket, see Adding a bucket policy using the Amazon S3 console in the Amazon S3 documentation. **Important** If you specified a prefix when you created your trail in CloudTrail, make sure that you include it in the S3 bucket policy. The prefix is an optional addition to the S3 object key that creates a folder-like organization in your S3 bucket. For more information about this, see Creating a	Systems administrator

Task	Description	Skills required
	trail in the CloudTrail documentation.	
Create a KMS key.	Create an AWS KMS key for CloudTrail to encrypt objects before adding them to the S3 bucket. For help with this story, see Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS) in the CloudTrail documentation.	Systems administrator
Add a key policy to the KMS key.	Attach a KMS key policy to allow CloudTrail to use the KMS key. For help with this story, see Encrypting CloudTrail log files with AWS KMS–managed keys (SSE-KMS) in the CloudTrail documentation. **Important** CloudTrail doesn't require `Decrypt` permissions.	Systems administrator

Task	Description	Skills required
Create AssumeRole for Systems Manager runbook	Create an `AssumeRole` for Systems Manager Automation to run the runbook. For instructions and more information about this, see [Setting up automation](#) in the Systems Manager documentation.	Systems administrator

Create and test the Systems Manager Automation runbook

Task	Description	Skills required
Create the Systems Manager Automation runbook.	Use the `cloudtrail-remediation-action.yml` file (attached) to create the Systems Manager Automation runbook. For more information about this, see [Creating Systems Manager documents](#) in the Systems Manager documentation.	Systems administrator
Test the runbook.	On the Systems Manager console, test the Systems Manager Automation runbook that you created earlier. For more information about this, see [Running a simple automation](#) in the Systems Manager documentation.	Systems administrator

Set up the automatic remediation rule in AWS Config

Task	Description	Skills required
Add the CloudTrail-enabled rule.	On the AWS Config console, choose **Rules** and then choose **Add rule**. On the **Add rule** page, choose **Add custom** rule. On the **Configure rule** page, enter a name and description, and add the `cloudtrail-enabled` rule. For more information, see [Managing your AWS Config rules](#) in the AWS Config documentation.	Systems administrator
Add the automatic remediation action.	From the **Actions** dropdown list, choose **Manage remediation**. Choose **Auto remediation** and then choose the Systems Manager runbook that you created earlier. The following are the required input parameters for CloudTrail: • `CloudTrailName` • `CloudTrailS3BucketName` • `CloudTrailKmsKeyId` • `AssumeRole` (optional)	Systems administrator

Task	Description	Skills required
	The following input parameters are set to **true** by default: • `IsMultiRegionTrail` • `IsOrganizationTrail` • `IncludeGlobalServiceEvents` • `EnableLogFileValidation` Retain the default values for the **Rate Limits parameter** and **Resource ID parameter**. Choose **Save**. For more information, see Remediating noncompliant AWS resources with AWS Config rules in the AWS Config documentation.	

Task	Description	Skills required
Test the automatic remediati on rule.	To test the automatic remediation rule, open the CloudTrail console, choose **Trails**, and then choose the trail. Choose **Stop logging** to turn off logging for the trail. When you are prompted to confirm, choose **Stop logging**. CloudTrail stops logging activity for that trail. Follow the instructions from Evaluating your resources in the AWS Config documenta tion to make sure that CloudTrail was automatically re-enabled.	Systems administrator

Related resources

Configure CloudTrail

- Create an S3 bucket
- Amazon S3 bucket policy for CloudTrail
- Adding a bucket policy using the Amazon S3 console
- Creating a trail
- Setting up automation
- Encrypting CloudTrail log files with AWS KMS managed keys (SSE-KMS)

Create and test the Systems Manager Automation runbook

- Creating Systems Manager documents
- Running a simple automation

Set up the automatic remediation rule in AWS Config

- Managing your AWS Config rules
- Remediating noncompliant AWS resources with AWS Config rules

Additional resources

- AWS CloudTrail - Security best practices
- Getting started with AWS Systems Manager
- Getting started with AWS Config
- Getting started with AWS CloudTrail

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Automatically remediate unencrypted Amazon RDS DB instances and clusters

Ajay Rawat and Josh Joy, Amazon Web Services

Summary

This pattern describes how to automatically remediate unencrypted Amazon Relational Database Service (Amazon RDS) DB instances and clusters on Amazon Web Services (AWS) by using AWS Config, AWS Systems Manager runbooks, and AWS Key Management Service (AWS KMS) keys.

Encrypted RDS DB instances provide an additional layer of data protection by securing your data from unauthorized access to the underlying storage. You can use Amazon RDS encryption to increase data protection of your applications deployed in the AWS Cloud, and to fulfill compliance requirements for encryption at rest. You can enable encryption for an RDS DB instance when you create it, but not after it's created. However, you can add encryption to an unencrypted RDS DB instance by creating a snapshot of your DB instance, and then creating an encrypted copy of that snapshot. You can then restore a DB instance from the encrypted snapshot to get an encrypted copy of your original DB instance.

This pattern uses AWS Config Rules to evaluate RDS DB instances and clusters. It applies remediation by using AWS Systems Manager runbooks, which define the actions to be performed on noncompliant Amazon RDS resources, and AWS KMS keys to encrypt the DB snapshots. It then enforces service control policies (SCPs) to prevent the creation of new DB instances and clusters without encryption.

The code for this pattern is provided in GitHub.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Files from the GitHub source code repository for this pattern downloaded to your computer
- An unencrypted RDS DB instance or cluster
- An existing AWS KMS key for encrypting RDS DB instances and clusters
- Access to update the KMS key resource policy

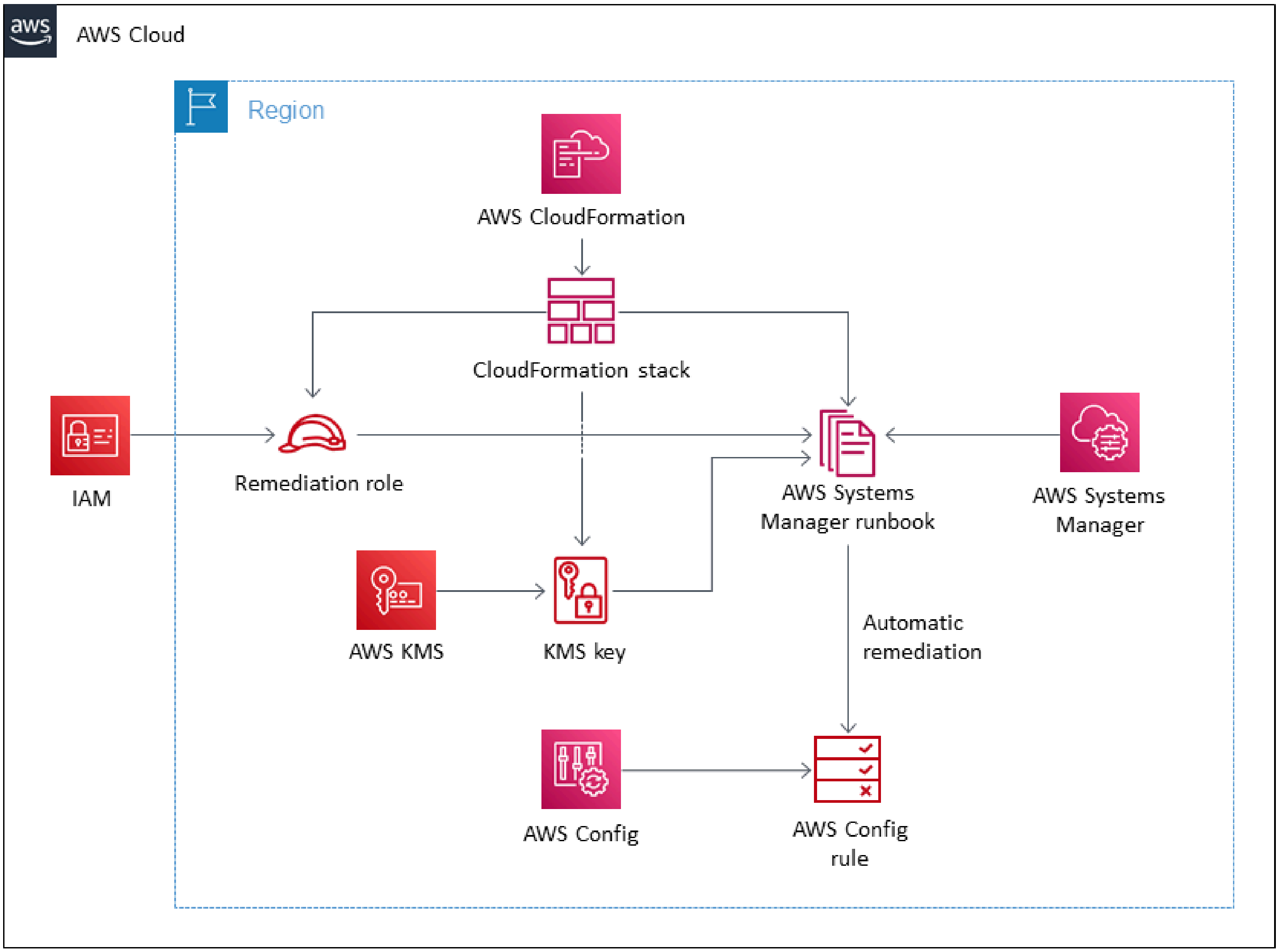
- AWS Config enabled in your AWS account (see Getting Started with AWS Config in the AWS documentation)

Limitations

- You can enable encryption for an RDS DB instance only when you create it, not after it has been created.
- You can't have an encrypted read replica of an unencrypted DB instance or an unencrypted read replica of an encrypted DB instance.
- You can't restore an unencrypted backup or snapshot to an encrypted DB instance.
- Amazon RDS encryption is available for most DB instance classes. For a list of exceptions, see Encrypting Amazon RDS resources in the Amazon RDS documentation.
- To copy an encrypted snapshot from one AWS Region to another, you must specify the KMS key in the destination AWS Region. This is because KMS keys are specific to the AWS Region that they are created in.
- The source snapshot remains encrypted throughout the copy process. Amazon RDS uses envelope encryption to protect data during the copy process. For more information, see Envelope encryption in the AWS KMS documentation.
- You can't unencrypt an encrypted DB instance. However, you can export data from an encrypted DB instance and import the data into an unencrypted DB instance.
- You should delete a KMS key only when you are sure that you don't need to use it any longer. If you aren't sure, consider disabling the KMS key instead of deleting it. You can reenable a disabled KMS key if you need to use it again later, but you cannot recover a deleted KMS key.
- If you don't choose to retain automated backups, your automated backups that are in the same AWS Region as the DB instance are deleted. They can't be recovered after you delete the DB instance.
- Your automated backups are retained for the retention period that is set on the DB instance at the time you delete it. This set retention period occurs whether or not you choose to create a final DB snapshot.
- If automatic remediation is enabled, this solution encrypts all databases that have the same KMS key.

Architecture

The following diagram illustrates the architecture for the AWS CloudFormation implementation. Note that you can also implement this pattern by using the AWS Cloud Development Kit (AWS CDK).



Tools

Tools

- [AWS CloudFormation](#) helps you automatically set up your AWS resources. It enables you to use a template file to create and delete a collection of resources together as a single unit (a stack).
- [AWS Cloud Development Kit (AWS CDK)](#) is a software development framework for defining your cloud infrastructure in code and provisioning it by using familiar programming languages.

AWS services and features

- AWS Config keeps track of the configuration of your AWS resources and their relationships to your other resources. It can also evaluate those AWS resources for compliance. This service uses rules that can be configured to evaluate AWS resources against desired configurations. You can use a set of AWS Config managed rules for common compliance scenarios, or you can create your own rules for custom scenarios. When an AWS resource is found to be noncompliant, you can specify a remediation action through an AWS Systems Manager runbook and optionally send an alert through an Amazon Simple Notification Service (Amazon SNS) topic. In other words, you can associate remediation actions with AWS Config Rules and choose to run them automatically to address noncompliant resources without manual intervention. If a resource is still noncompliant after automatic remediation, you can set the rule to try automatic remediation again.
- Amazon Relational Database Service (Amazon RDS) makes it easier to set up, operate, and scale a relational database in the cloud. The basic building block of Amazon RDS is the DB instance, which is an isolated database environment in the AWS Cloud. Amazon RDS provides a selection of instance types that are optimized to fit different relational database use cases. Instance types comprise various combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your database. Each instance type includes several instance sizes, allowing you to scale your database to the requirements of your target workload.
- AWS Key Management Service (AWS KMS) is a managed service that makes it easy for you to create and control AWS KMS keys, which encrypt your data. A KMS key is a logical representation of a root key. The KMS key includes metadata, such as the key ID, creation date, description, and key state.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- Service control policies (SCPs) offer central control over the maximum available permissions for all accounts in your organization. SCPs help you ensure that your accounts stay within your organization’s access control guidelines. SCPs don't affect users or roles in the management account. They affect only the member accounts in your organization. We strongly recommend that you don't attach SCPs to the root of your organization without thoroughly testing the impact that the policy has on accounts. Instead, create an organizational unit (OU) that you can move your accounts into one at a time, or at least in small numbers, to ensure that you don't inadvertently lock users out of key services.

Code

The source code and templates for this pattern are available in a GitHub repository. The pattern provides two implementation options: You can deploy an AWS CloudFormation template to create the remediation role that encrypts RDS DB instances and clusters, or use the AWS CDK. The repository has separate folders for these two options.

The Epics section provides step-by-step instructions for deploying the CloudFormation template. If you want to use the AWS CDK, follow the instructions in the `README.md` file in the GitHub repository.

Best practices

- Enable data encryption both at rest and in transit.
- Enable AWS Config in all accounts and AWS Regions.
- Record configuration changes to all resource types.
- Rotate your IAM credentials regularly.
- Leverage tagging for AWS Config, which makes is easier to manage, search for, and filter resources.

Epics

Create the IAM remediation role and Systems Manager runbook

Task	Description	Skills required
Download the CloudFormation template.	Download the `unencrypted-to-encrypted-rds.template.json` file from the GitHub repository.	DevOps engineer
Create the CloudFormation stack.	1. Sign in to the AWS Management Console and open the CloudFormation console. 2. Launch the `unencrypted-to-encrypted-`	DevOps engineer

Task	Description	Skills required
	`rds.template.json` template to create a new stack. For more information about deploying templates, see the [AWS CloudFormation documentation]().	
Review CloudFormation parameters and values.	1. Review stack details and update values based on your environment requirements. 2. Choose **Create stack** to deploy the template.	DevOps engineer
Review the resources.	When the stack has been created, its status changes to **CREATE_COMPLETE**. Review the created resources (IAM role, Systems Manager runbook) in the CloudFormation console.	DevOps engineer

Update the AWS KMS key policy

Task	Description	Skills required
Update your KMS key policy.	1. Make sure that the key alias `alias/RDSEncryptionAtRestKMSAlias` exists. 2. The key policy statement should include the IAM	DevOps engineer

Task	Description	Skills required
	remediation role. (Check the resources created by the CloudFormation template you deployed in the previous epic.) 3. In the following key policy, update the portions that are in bold to match your account and the IAM role that was created. `{ "Sid": "Allow access through RDS for all principals in the account that are authorized to use RDS", "Effect": "Allow", "Principal": { "AWS": "arn:aws:iam:: <your-AWS-account-ID>":role/<your-IAM-remediation-role>" }, "Action": ["kms:Encrypt", "kms:Decrypt", "kms:ReEncrypt*", "kms:GenerateDataKey*", "kms:CreateGrant", "kms:ListGrants",`	

Task	Description	Skills required

	``
	        "kms:Desc
	ribeKey"
	    ],
	    "Resource": "*",
	    "Condition": {
	        "StringEquals":
	 {
	            "kms:ViaS
	ervice": "rds.us-e
	ast-1.amazonaws.com",
	            "kms:Call
	erAccount": "<your-AW
	S-account-ID>"
	        }
	    }
	}
	``	

Find and remediate noncompliant resources

Task	Description	Skills required
View noncompliant resources.	1. To view a list of noncompli ant resources, open the [AWS Config console](https://console.aws.amazon.com/config/). 2. In the navigation pane, choose **Rules**, and then choose the `rds-stora ge-encrypted` rule. The noncompliant resources listed in the AWS Config console will be instances, not clusters. The remediati on automation encrypts instances and clusters, and creates either a newly	DevOps engineer

Task	Description	Skills required
	encrypted instance or a newly created cluster. However, be sure not to simultaneously remediate multiple instances that belong to the same cluster. Before you remediate any RDS DB instances or volumes, make sure that the RDS DB instance is not in use. Confirm that there are no write operations occurring while the snapshot is being created, to ensure that the snapshot contains the original data. Consider enforcing a maintenance window during which the remediation will run.	

Task	Description	Skills required
Remediate noncompliant resources.	1. When you are ready and the maintenance window is in effect, choose the resource to remediate, and then choose **Remediate**. The **Action status** column should now display **Action execution queued**. 2. View the progress and status of the remediation in Systems Manager. Open the [Systems Manager console](). In the navigation pane, choose **Automation**, and then select the execution ID of the corresponding automation to view further details.	DevOps engineer
Verify that the RDS DB instance is available.	After the automation completes, the newly encrypted RDS DB instance will become available. The encrypted RDS DB instance will have the prefix `encrypted` followed by the original name. For example, if the unencrypted RDS DB instance name was `database-1` , the newly encrypted RDS DB instance would be `encrypted-database-1` .	DevOps engineer

Task	Description	Skills required
Terminate the unencrypted instance.	After remediation is complete and the newly encrypted resource has been validated, you can terminate the unencrypted instance. Make sure to confirm that the newly encrypted resource matches the unencrypted resource before you terminate any resources.	DevOps engineer

Enforce SCPs

Task	Description	Skills required
Enforce SCPs.	Enforce SCPs to prevent DB instances and clusters from being created without encryption in the future. Use the `rds_encrypted.json` file that's provided in the GitHub repository for this purpose, and follow the instructions in the AWS documentation.	Security engineer

Related resources

References

- Setting up AWS Config
- AWS Config custom rules
- AWS KMS concepts

- AWS Systems Manager documents
- Service control policies

Tools

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)

Guides and patterns

- Automatically re-enable AWS CloudTrail by using a custom remediation rule in AWS Config

Additional information

How does AWS Config work?

When you use AWS Config, it first discovers the supported AWS resources that exist in your account and generates a configuration item for each resource. AWS Config also generates configuration items when the configuration of a resource changes, and it maintains historical records of the configuration items of your resources from the time you start the configuration recorder. By default, AWS Config creates configuration items for every supported resource in the AWS Region. If you don't want AWS Config to create configuration items for all supported resources, you can specify the resource types that you want it to track.

How are AWS Config and AWS Config Rules related to AWS Security Hub?

AWS Security Hub is a security and compliance service that provides security and compliance posture management as a service. It uses AWS Config and AWS Config Rules as its primary mechanism to evaluate the configuration of AWS resources. AWS Config Rules can also be used to evaluate resource configuration directly. Other AWS services, such AWS Control Tower and AWS Firewall Manager, also use AWS Config Rules.

Automatically rotate IAM user access keys at scale with AWS Organizations and AWS Secrets Manager

Tracy Hickey, Michael Davie, Arvind Patel, Laura Seletos, and Gaurav Verma, Amazon Web Services

Summary

Important: As a best practice, AWS recommends that you use AWS Identity and Access Management (IAM) roles instead of IAM users with long-term credentials such as access keys. The approach documented in this pattern is intended only for legacy implementations that require long-lived AWS API credentials. For these implementations, we still recommend that you consider options for using short-term credentials, such as using Amazon Elastic Compute Cloud (Amazon EC2) instance profiles or IAM Roles Anywhere. The approach in this article is only for cases where you are unable to change to using short-term credentials immediately, and you require long-term credentials to be rotated on a schedule. With this approach, you are responsib le for periodically updating your legacy application code or configuration to use the rotated API credentials.

Access keys are long-term credentials for an IAM user. Regularly rotating your IAM credentials helps prevent a compromised set of IAM access keys from accessing components in your AWS account. Rotating IAM credentials is also an important part of security best practices in IAM.

This pattern helps you automatically rotate IAM access keys by using AWS CloudFormation templates, which are provided in the GitHub IAM key rotation repository.

The pattern supports deployment in a single account or multiple accounts. If you're using AWS Organizations, this solution identifies all AWS account IDs within your organization and dynamically scales as accounts are removed or new accounts are created. The centralized AWS Lambda function uses an assumed IAM role to locally run the rotation functions across multiple accounts that you select.

- New IAM access keys are generated when existing access keys are 90 days old.
- The new access keys are stored as a secret in AWS Secrets Manager. A resource-based policy allows only the specified IAM principal to access and retrieve the secret. If you choose to store keys in the management account, the keys for all accounts are stored in the management account.

- The email address assigned to the owner of the AWS account where the new access keys were created receives a notification.
- The previous access keys are deactivated at 100 days old, and then deleted at 110 days old.
- A centralized email notification is sent to the AWS account owner.

Lambda functions and Amazon CloudWatch automatically perform these actions. You can then retrieve the new access key pair and replace them in your code or applications. The rotation, deletion, and deactivation periods can be customized.

Prerequisites and limitations

- At least one active AWS account.
- AWS Organizations, configured and set up (see tutorial).
- Permissions to query AWS Organizations from your management account. For more information, see AWS Organizations and service-linked roles in the AWS Organizations documentation.
- An IAM principal that has permissions to launch the AWS CloudFormation template and associated resources. For more information, see Grant self-managed permissions in the AWS CloudFormation documentation.
- An existing Amazon Simple Storage Service (Amazon S3) bucket to deploy the resources.
- Amazon Simple Email Service (Amazon SES) moved out of the sandbox. For more information, see Moving out of the Amazon SES sandbox in the Amazon SES documentation.
- If you choose to run Lambda in a virtual private cloud (VPC), the following resources, which should be created before you run the main CloudFormation template:
  - A VPC.
  - A subnet.
  - Endpoints for Amazon SES, AWS Systems Manager, AWS Security Token Service (AWS STS), Amazon S3, and AWS Secrets Manager. (You can run the endpoint template that's provided in the GitHub IAM key rotation repository to create these endpoints.)
- The Simple Mail Transfer Protocol (SMTP) user and password stored in AWS Systems Manager parameters (SSM parameters). Parameters must match the main CloudFormation template parameters.

Architecture

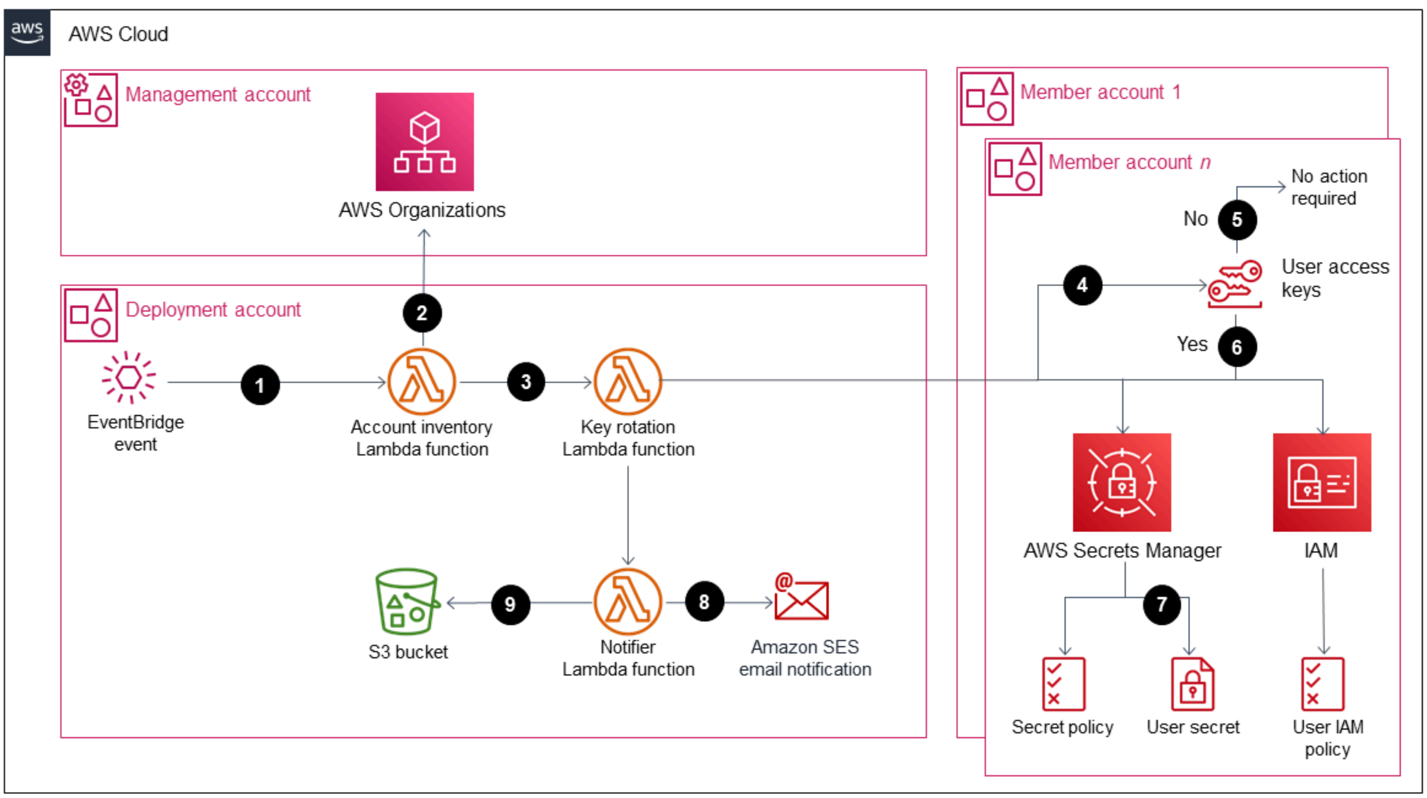
Technology stack

- Amazon CloudWatch
- Amazon EventBridge
- IAM
- AWS Lambda
- AWS Organizations
- Amazon S3

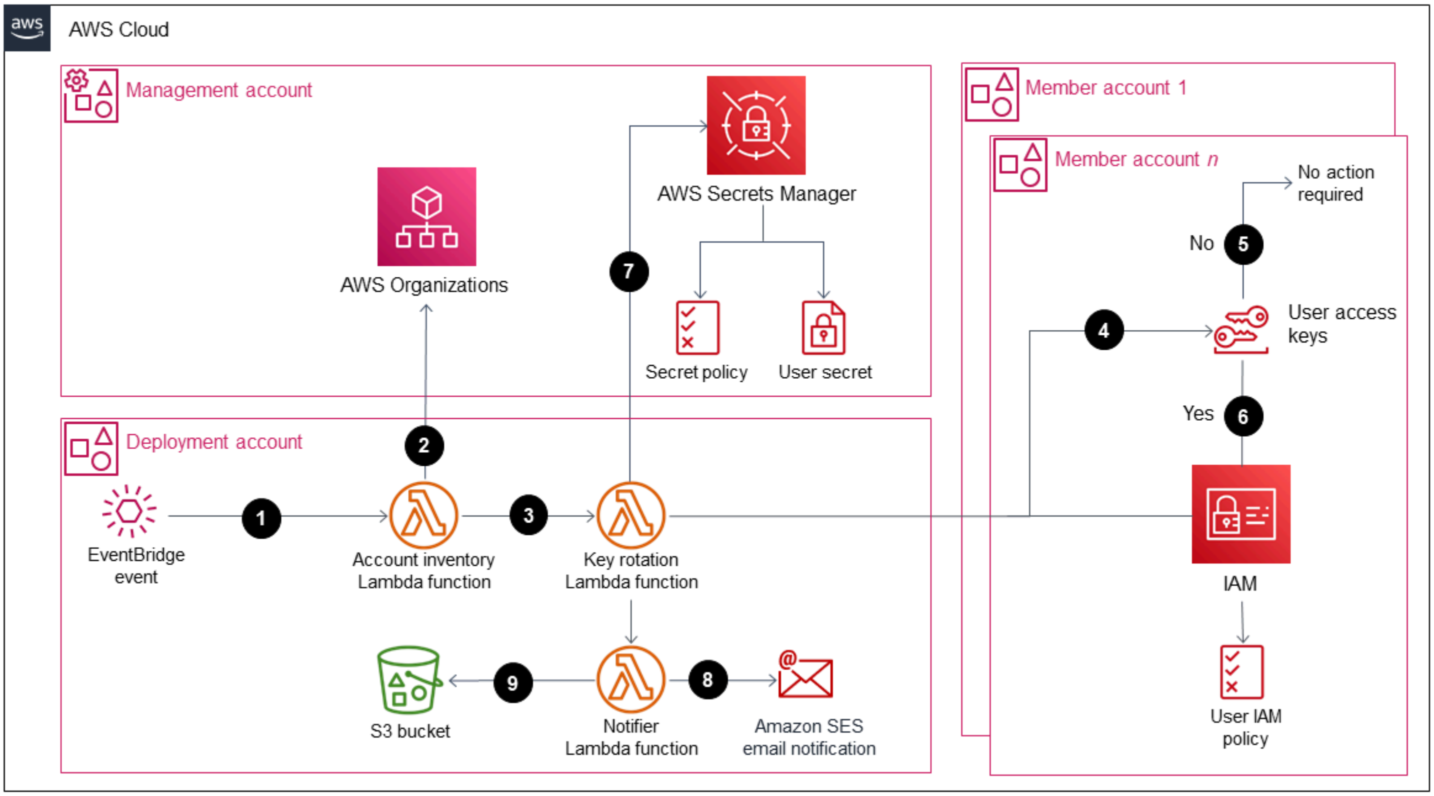
Architecture

The following diagrams show the components and workflows for this pattern. The solution supports two scenarios for storing the credentials: in a member account and in the management account.

Option 1: Store the credentials in a member account



Option 2: Store the credentials in the management account



The diagrams show the following workflow:

1. An EventBridge event initiates an `account_inventory` Lambda function every 24 hours.
2. This Lambda function queries AWS Organizations for a list of all AWS account IDs, account names, and account emails.
3. The `account_inventory` Lambda function initiates an `access_key_auto_rotation` Lambda function for each AWS account ID and passes the metadata to it for additional processing.
4. The `access_key_auto_rotation` Lambda function uses an assumed IAM role to access the AWS account ID. The Lambda script runs an audit against all users and their IAM access keys in the account.
5. If the IAM access key's age hasn't exceeded the best practice threshold, the Lambda function takes no further action.
6. If the IAM access key's age has exceeded the best practice threshold, the `access_key_auto_rotation` Lambda function determines which rotation action to perform.
7. When action is required, the `access_key_auto_rotation` Lambda function creates and updates a secret in AWS Secrets Manager if a new key is generated. A resource-based policy is also created that allows only the specified IAM principal to access and retrieve the secret. In the

case of option 1, the credentials are stored in Secrets Manager in the respective account. In the case of option 2 (if the `StoreSecretsInCentralAccount` flag is set to **True**), the credentials are stored in Secrets Manager in the management account.

8. A `notifier` Lambda function is initiated to notify the account's owner of the rotation activity. This function receives the AWS account ID, account name, account email, and the rotation actions that were performed.
9. The `notifier` Lambda function queries the deployment S3 bucket for an email template and dynamically updates it with the relevant activity metadata. The email is then sent to the account owner's email address.

Notes:

- This solution supports resiliency in multiple Availability Zones. However, it doesn't support resiliency in multiple AWS Regions. For support in multiple Regions, you can deploy the solution in the second Region and keep the key rotation EventBridge rule disabled. You can then enable the rule when you want to run the solution in the second Region.
- You can run this solution in audit mode. In audit mode, IAM access keys aren't modified, but an email is sent to notify users. To run the solution in audit mode, set the `DryRunFlag` flag to **True** when you run the key rotation template or in the environment variable for the `access_key_auto_rotation` Lambda function.

Automation and scale

The CloudFormation templates that automate this solution are provided in the GitHub [IAM key rotation] repository and listed in the *Code* section. In AWS Organizations, you can use [CloudFormation StackSets] to deploy the `ASA-iam-key-auto-rotation-iam-assumed-roles.yaml` CloudFormation template in multiple accounts instead of deploying the solution individually to each member account.

Tools

AWS services

- [Amazon CloudWatch] helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- [AWS Identity and Access Management (IAM)] helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.

- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- Amazon Simple Email Service (Amazon SES) helps you send and receive emails by using your own email addresses and domains.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.
- Amazon VPC endpoints provide an interface to connect to services powered by AWS PrivateLink, including many AWS services. For each subnet that you specify from your VPC, an endpoint network interface is created in the subnet and assigned a private IP address from the subnet address range.

Code

The required AWS CloudFormation templates, Python scripts, and runbook documentation are available in the GitHub IAM key rotation repository. The templates are deployed as follows.

Template	Deploy in	Notes
`ASA-iam-key-auto-rotation-and-notifier-solution.yaml`	Deployment account	This is the main template for the solution.
`ASA-iam-key-auto-rotation-iam-assumed-roles.yaml`	Single or multiple member accounts where you want to rotate the credentials	You can use CloudFormation stack sets to deploy

		this template in multiple accounts.
`ASA-iam-key-auto-rotation-list-accounts-role.yaml`	Central/management account	Use this template to keep an inventory of accounts in AWS Organizations.
`ASA-iam-key-auto-rotation-vpc-endpoints.yaml`	Deployment account	Use this template to automate the creation of endpoints only if you want to run the Lambda functions in a VPC (set the `RunLambdaInVPC` parameter to **True** in the main template).

Epics

Set up the solution

Task	Description	Skills required
Choose your deployment S3 bucket.	Sign in to the AWS Management Console for your account, open the [Amazon S3 console](), and then choose the S3 bucket for your deployment. If you want to implement the solution for multiple accounts in AWS Organizations, sign in to the management account for your organization.	Cloud architect
Clone the repository.	Clone the GitHub [IAM key rotation]() repository to your local desktop.	Cloud architect

Task	Description	Skills required
Upload the files to the S3 bucket.	Upload the cloned files to your S3 bucket. Use the following default folder structure to copy and paste all cloned files and directories: `asa/asa-iam-rotation` **Note** You can customize this folder structure in the CloudFormation templates.	Cloud architect
Modify the email template.	Modify the `iam-auto-key-rotation-enforcement.html` email template (located in the `template` folder) according to your requirements. Replace `[Department Name Here]` at the end of the template with your department's name.	Cloud architect

Deploy the solution

Task	Description	Skills required
Launch the CloudFormation template for key rotation.	1. Launch the `ASA-iam-key-auto-rotation-and-notifier-solution.yaml` template in	Cloud architect

Task	Description	Skills required
	the deployment account. For more information, see [Selecting a stack template](#) in the CloudFormation documentation. 2. Specify values for parameters, including: • **CloudFormation S3 Bucket Name** (`S3BucketName`) – The name of the deploymen t S3 bucket that contains your Lambda code. • **CloudFormation S3 Bucket Prefix** (`S3BucketPrefix`) – The S3 bucket's prefix. • **Assumed IAM Role Name** (`IAMRoleName`) – The role name that the `key-rotation` Lambda function will assume to rotate the keys. • **IAM Execution Role Name** (`Execution RoleName`) – The name of the IAM execution role that's used by the `key-rotation` Lambda function. • **Inventory Execution Role Name** (`Inventory`	

Task	Description	Skills required
	`ExecutionRoleName`) – The name of the IAM execution role that's used by the `account_inventory` Lambda function. • **Dry Run Flag (Audit Mode)** (`DryRunFlag`) – Set to **True** to turn on audit mode (default). Set to **False** to turn on enforcement mode. • **Account to List Organization Accounts** (`OrgListAccount`) – The account ID of the central/management account that will be used to list the accounts in the organization. • **List Accounts Role Name** (`OrgListRole`) – The role name that will be used to list the accounts in the organization. • **Secrets Store flag for central account** (`StoreSecretsInCentralAccount`) – Set to **True** to store secrets in the central account. Set to **False** to store	

Task	Description	Skills required
	secrets in the respective account. • **Regions to replicate the credentials** (`CredentialReplicationRegions`) – The AWS Regions where you want to replicate the credentials (Secrets Manager), separated by commas; for example, `us-east-2,us-west-1,us-west-2` . Skip the Region where you are creating the stack. • **Run Lambda in VPC** (`RunLambdaInVpc`) – Set to **True** to run Lambda functions in a specified VPC. You must have VPC endpoints created and attach a NAT gateway to the subnet that contains the Lambda function. For more information, see the re:Post article that covers this option. • **VPC Id for Lambda functions** (`VpcId`), **VPC CIDR for Security Group Rule** (`VpcCidr`), and **Subnet Id for Lambda functions** (`SubnetId`)	

Task	Description	Skills required
	– Provide information about the VPC, CIDR, and subnet if you set `RunLambdaInVpc` to **True**. • **Admin Email Address** (`AdminEmailAddress`) – A valid email address to send notifications to. • **AWS Organization ID** (`AWSOrgID`) – The unique ID of your organization. This ID begins with `o-` and is followed by 10-32 lowercase letters or digits. • **Email Template File name [Audit Mode]** (`EmailTemplateAudit`) and **[Enforce Mode]** (`EmailTemplateEnforce`) – The filename of the email HTML template to be sent out by the `notifier` module for audit mode and enforce mode. • **SMTP User SSM Parameter Name** (`SMTPUserParamName`) and **SMTP Password**	

Task	Description	Skills required
	SSM Parameter Name (`SMTPPasswordParamName`) – User and password information for Simple Mail Transfer Protocol (SMTP).	

Task	Description	Skills required
Launch the CloudFormation template for assumed roles.	1. In the [AWS CloudFormation console](), launch the `ASA-iam-key-auto-rotation-iam-assumed-roles.yaml` template for each account where you want to rotate keys. If you have more than one account, you can deploy the main CloudFormation template in your management account as a stack and deploy the `ASA-iam-key-auto-rotation-iam-assumed-roles.yaml` template with CloudFormation stack sets to all required accounts. For more information, see [Working with AWS CloudFormation StackSets]() in the CloudFormation documentation. 2. Specify values for the following parameters: • **Assumed IAM Role Name** (`IAMRoleName`) – IAM role name that will be assumed by the Lambda `access_key_auto_rotation` function. You can keep the default value.	Cloud architect

Task	Description	Skills required
	• **IAM Execution Role Name** (`ExecutionRoleName`) – The IAM role that will assume the sub-account role to run the Lambda function. • **Primary AWS Account ID** (`PrimaryAccountID`) – The AWS account ID where the main template will be deployed. • **IAM Exemption Group** (`IAMExemptionGroup`) – The IAM group name being used to facilitate IAM accounts that you want to exclude from automatic key rotation.	

Task	Description	Skills required
Launch the CloudFormation template for account inventory.	1. Launch the `ASA-iam-key-auto-rotation-list-accounts-role.yaml` template in the management/central account 2. Specify values for the following parameters: • **Assumed IAM Role Name** (`IAMRoleName`) – IAM role name that the Lambda `access_key_auto_rotation` function will assume. • **IAM Execution Role Name for Account Lambda** (`AccountExecutionRoleName`) – The name of the IAM role that the Lambda `notifier` function will assume. • **IAM Execution Role Name for rotation Lambda** (`RotationExecutionRoleName`) – The name of the IAM role that the Lambda `access_key_auto_rotation` function will assume. • **Primary AWS Account ID** (`PrimaryAc`	Cloud architect

Task	Description	Skills required
	`countID`) – The AWS account ID where the main template will be deployed.	
Launch the CloudFormation template for VPC endpoints.	This task is optional. 1. Launch the `ASA-iam-key-auto-rotation-vpc-endpoints.yaml` template in the deployment account. 2. Specify values for the following parameters: • **VPC ID** (`pVpcId`), **Subnet Id** (`pSubnetId`), and **CIDR range for VPC** (`pVPCCidr`) – Provide information about the VPC, CIDR, and subnet. • Set the parameter for each VPC endpoint to **True**. If you already have endpoints, you can choose **False**.	Cloud architect

Related resources

- Security best practices in IAM (IAM documentation)
- AWS Organizations and service-linked roles (AWS Organizations documentation)
- Selecting a stack template (CloudFormation documentation)
- Working with AWS CloudFormation StackSets (CloudFormation documentation)

Automatically validate and deploy IAM policies and roles by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros

Helton Ribeiro and Guilherme Simoes, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more.

This pattern describes the steps and provides code to create a deployment pipeline that allows your development teams to create AWS Identity and Access Management (IAM) policies and roles in your Amazon Web Services (AWS) accounts. This approach helps your organization reduce overhead for your operational teams and speed up the deployment process. It also helps your developers to create IAM roles and policies that are compatible with your existing governance and security controls.

This pattern's approach uses AWS Identity and Access Management Access Analyzer to validate the IAM policies that you want to attach to IAM roles and uses AWS CloudFormation to deploy the IAM roles. However, instead of directly editing the AWS CloudFormation template file, your development team creates JSON-formatted IAM policies and roles. An AWS CloudFormation macro transforms these JSON-formatted policy files into AWS CloudFormation IAM resource types before beginning the deployment.

The deployment pipeline (`RolesPipeline`) has source, validation, and deployment stages. During the source stage, your development team pushes the JSON files that contain the definition of the IAM roles and policies to an AWS CodeCommit repository. AWS CodeBuild then runs a script to validate those files and copies them to an Amazon Simple Storage Service (Amazon S3) bucket. Because your development teams don't have direct access to the AWS CloudFormation template file stored in a separate S3 bucket, they must follow the JSON file creation and validation process.

Finally, during the deployment phase, AWS CodeDeploy uses an AWS CloudFormation stack to update or delete the IAM policies and roles in an account.

> **Important**
>
> This pattern's workflow is a proof of concept (POC) and we recommend that you only use it in a test environment. If you want to use this pattern's approach in a production environment, see Security best practices in IAM in the IAM documentation and make the required changes to your IAM roles and AWS services.

Prerequisites and limitations

Prerequisites

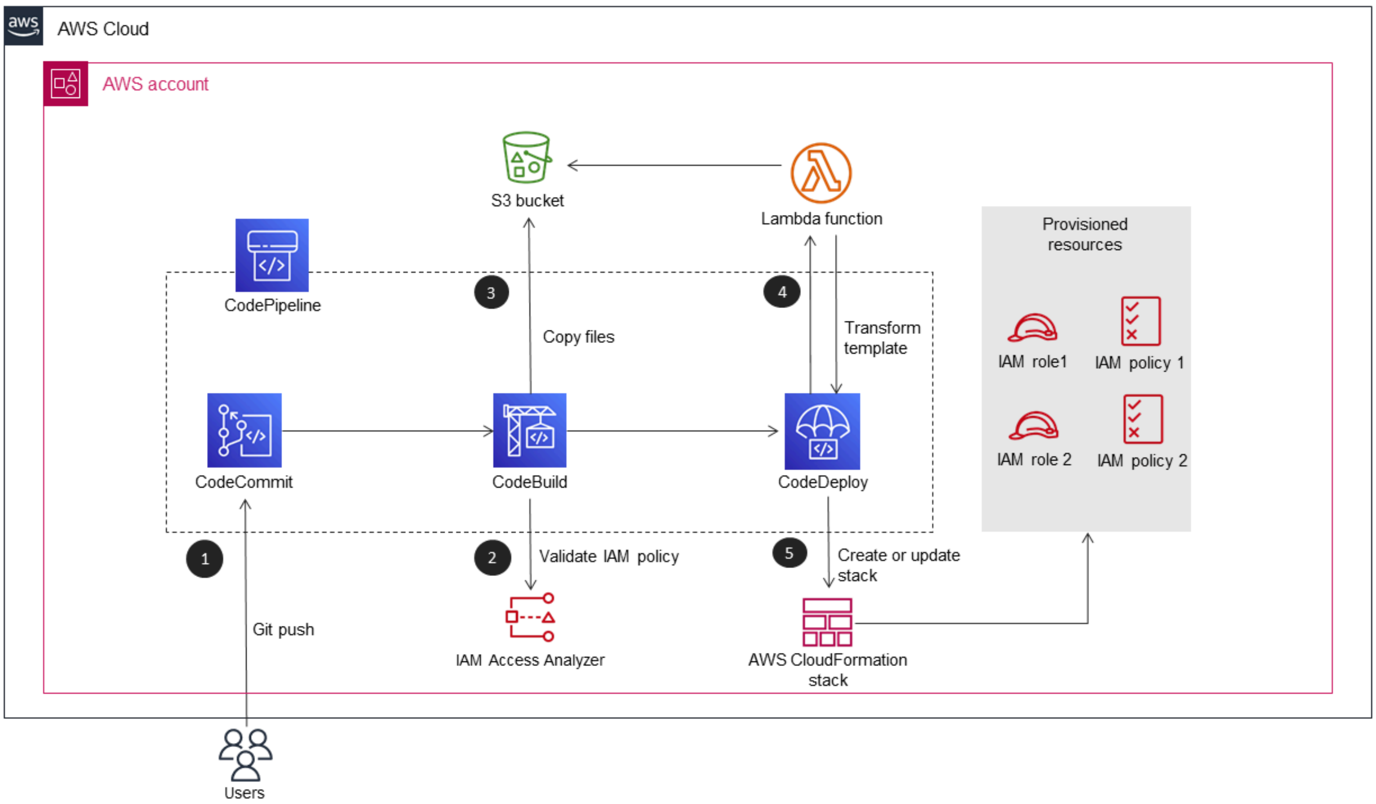
- An active AWS account.
- A new or existing S3 bucket for the `RolesPipeline` pipeline. Make sure that the access credentials you're using have permissions to upload objects to this bucket.
- AWS Command Line Interface (AWS CLI), installed and configured. For more information about this, see Installing, updating, and uninstalling the AWS CLI in the AWS CLI documentation.
- AWS Serverless Application Model (AWS SAM) CLI, installed and configured. For more information about this, see Installing the AWS SAM CLI in the AWS SAM documentation.
- Python 3, installed on your local machine. For more information about this, see the Python documentation.
- A Git client, installed and configured.
- The GitHub `IAM roles pipeline` repository, cloned to your local machine.
- Existing JSON-formatted IAM policies and roles. For more information about this, see the ReadMe file in the Github `IAM roles pipeline` repository.
- Your developer team must not have permissions to edit this solution's AWS CodePipeline, CodeBuild, and CodeDeploy resources.

Limitations

- This pattern's workflow is a proof of concept (POC) and we recommend that you only use it in a test environment. If you want to use this pattern's approach in a production environment, see Security best practices in IAM in the IAM documentation and make the required changes to your IAM roles and AWS services.

Architecture

The following diagram shows you how to automatically validate and deploy IAM roles and policies to an account by using CodePipeline, IAM Access Analyzer, and AWS CloudFormation macros.



The diagram shows the following workflow:

1. A developer writes JSON files that contain the definitions for the IAM policies and roles. The developer pushes the code to a CodeCommit repository and CodePipeline then initiates the `RolesPipeline` pipeline.
2. CodeBuild validates the JSON files by using IAM Access Analyzer. If there are any security or error-related findings, the deployment process is stopped.
3. If there are no security or error-related findings, the JSON files are sent to the `RolesBucket` S3 bucket.
4. An AWS CloudFormation macro implemented as an AWS Lambda function then reads the JSON files from the `RolesBucket` bucket and transforms them into AWS CloudFormation IAM resources types.
5. A predefined AWS CloudFormation stack installs, updates, or deletes the IAM policies and roles in the account.

Automation and scale

AWS CloudFormation templates that automatically deploy this pattern are provided in the GitHub IAM roles pipeline repository.

Tools

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- IAM Access Analyzer helps you identify the resources in your organization and accounts, such as S3 buckets or IAM roles, that are shared with an external entity. This helps you to identify unintended access to your resources and data.
- AWS Serverless Application Model (AWS SAM) is an open-source framework that helps you build serverless applications in the AWS Cloud.

Code

The source code and templates for this pattern are available in the GitHub IAM roles pipeline repository.

Epics

Clone the repository

Task	Description	Skills required
Clone the sample repository.	Clone the GitHub IAM roles pipeline repository to your local machine.	App developer, General AWS

Deploy the RolesPipeline pipeline

Task	Description	Skills required
Deploy the pipeline.	1. Navigate to the directory that contains the cloned repository. 2. **Important** Run the `make deploy bucket=<bucket_name>` command. : you must replace `<bucket_name>` with the bucket name for your existing S3 bucket. 3. Run the `aws codepipeline get-pipeline –name RolesPipeline` command to check if your deployment is successful.	App developer, General AWS
Clone the pipeline's repository.	1. The `RolesPipeline` AWS CloudFormation stack creates the `roles-pipeline-repo` CodeCommit repository. 2. Sign in to the AWS Management Console, open the AWS CodeCommit console, and then copy the CodeCommit repository's URL to clone it to your	App developer, General AWS

Task	Description	Skills required
	local machine. For more information about this, see [Connect to an AWS CodeCommit repository](#) in the AWS CodeCommit documentation.	

Test the RolesPipeline pipeline

Task	Description	Skills required
Test the RolesPipeline pipeline with valid IAM policies and roles.	1. Create JSON files for your IAM policies and roles. You can use the samples in the `role-example` directory from the GitHub `IAM roles pipeline` repository. 2. **Important** Define your IAM policies and roles with the required configurations. : Make sure that you follow the format described in the `ReadMe` file from the GitHub `IAM roles pipeline` repository. 3. Push the modifications into the `roles-pipeline-`	App developer, General AWS

Task	Description	Skills required
	`repo` CodeCommit repository. 4. Verify the implementation of the `RolesPipeline` pipeline. 5. Make sure that the IAM policies and roles are correctly deployed in the account. 6. Validate if there is a permissions boundary associated to the IAM policies or roles. For more information about this, see [Permissions boundaries for IAM entities](#) in the IAM documentation.	
Test the RolesPipeline pipeline with invalid IAM policies and roles.	1. Modify the `roles-pipeline-repo` CodeCommit repository and include invalid IAM roles or policies. For example, you can use an action that doesn't exist or an invalid IAM policy version. 2. Verify the pipeline implementation. IAM Access Analyzer stops the pipeline during the validation stage if it detects invalid IAM policies or roles.	App developer, General AWS

Clean up your resources

Task	Description	Skills required
Prepare for cleanup.	Empty the S3 buckets and then run the `destroy` command.	App developer, General AWS
Delete the RolesStack stack.	1. The `RolesPipeline` pipeline creates a `RolesStack` AWS CloudFormation stack that deploys the IAM policies and roles. You must delete this stack before deleting the `RolesPipeline` pipeline. 2. Sign in to the AWS Management Console, open the AWS CloudFormation console, and then choose the `RolesStack` stack and choose **Delete**.	App developer, General AWS
Delete the RolesPipeline stack.	To delete the `RolesPipeline` AWS CloudFormation stack, follow the instructions from the ReadMe file in the Github `IAM roles pipeline` repository.	App developer, General AWS

Related resources

- IAM Access Analyzer - Policy validation (AWS News Blog)
- Using AWS CloudFormation macros to perform custom processing on templates (AWS CloudFormation documentation)

- Building Lambda functions with Python (AWS Lambda documentation)

Bidirectionally integrate AWS Security Hub with Jira software

Joaquin Rinaudo, Amazon Web Services

Summary

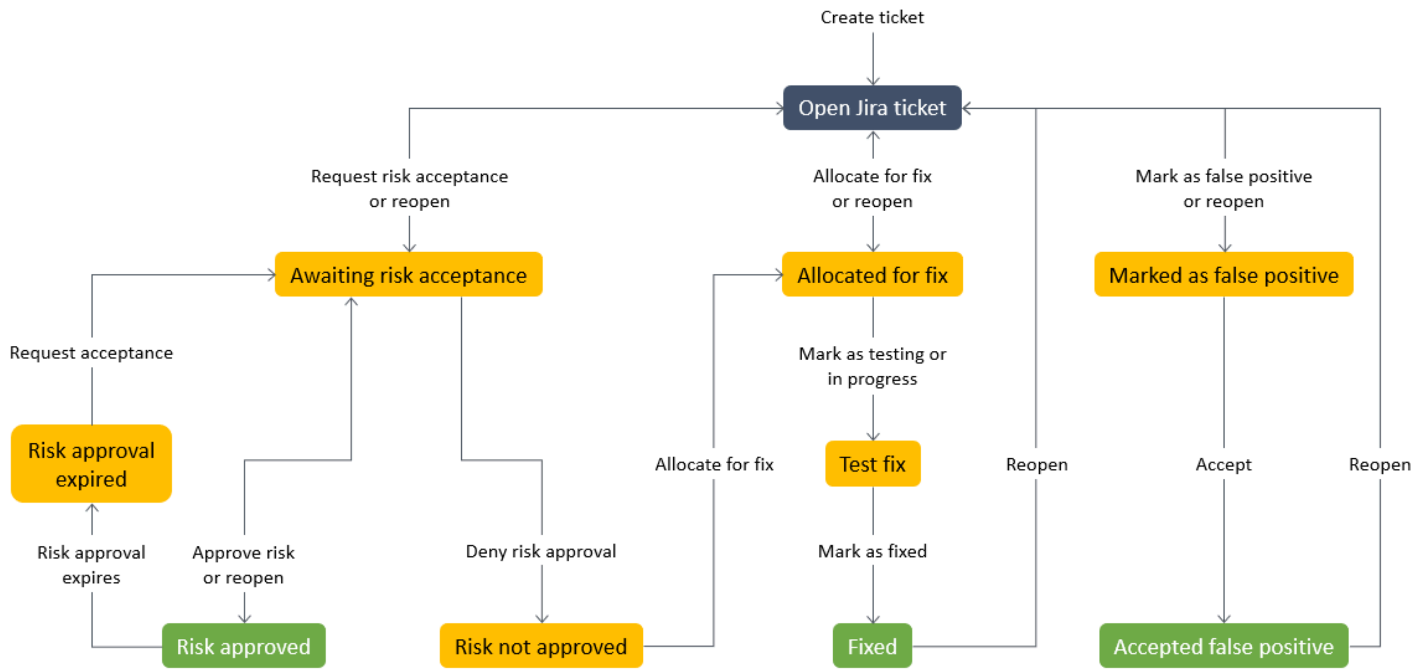
This solution supports a bidirectional integration between AWS Security Hub and Jira. Using this solution, you can automatically and manually create and update Jira tickets from Security Hub findings. Security teams can use this integration to notify developer teams of severe security findings that require action.

The solution allows you to:

- Select which Security Hub controls automatically create or update tickets in Jira.
- In the Security Hub console, use Security Hub custom actions to manually escalate tickets in Jira.
- Automatically assign tickets in Jira based on the AWS account tags defined in AWS Organizations. If this tag is not defined, a default assignee is used.
- Automatically suppress Security Hub findings that are marked as false positive or accepted risk in Jira.
- Automatically close a Jira ticket when its related finding is archived in Security Hub.
- Reopen Jira tickets when Security Hub findings reoccur.

Jira workflow

The solution uses a custom Jira workflow that allows developers to manage and document risks. As the issue moves through the workflow, bidirectional integration ensures that the status of the Jira ticket and Security Hub finding is synchronized across the workflows in both services. This workflow is a derivative of *SecDevOps Risk Workflow* by Dinis Cruz, licensed licensed under [Apache License version 2.0](). We recommend adding a Jira workflow condition so that only members of your security team can change the ticket status.



For an example of a Jira ticket automatically generated by this solution, see the Additional information section of this pattern.

Prerequisites and limitations

Prerequisites

- If you want to deploy this solution across a multi-account AWS environment:
  - Your multi-account environment is active and managed by AWS Organizations.
  - Security Hub is enabled on your AWS accounts.
  - In AWS Organizations, you have designated a Security Hub administrator account.
  - You have a cross-account AWS Identity and Access Management (IAM) role that has `AWSOrganizationsReadOnlyAccess` permissions to the AWS Organizations management account.
  - (Optional) You have tagged your AWS accounts with `SecurityContactID`. This tag is used to assign Jira tickets to the defined security contacts.
- If you want to deploy this solution within a single AWS account:
  - You have an active AWS account.
  - Security Hub is enabled on your AWS account.
- A Jira Data Center instance

> **Important**
>
> This solution supports use of Jira Cloud. However, Jira Cloud does not support importing XML workflows, so you need to manually re-create the workflow in Jira. You can find of the transitions and status in the GitHub repository.

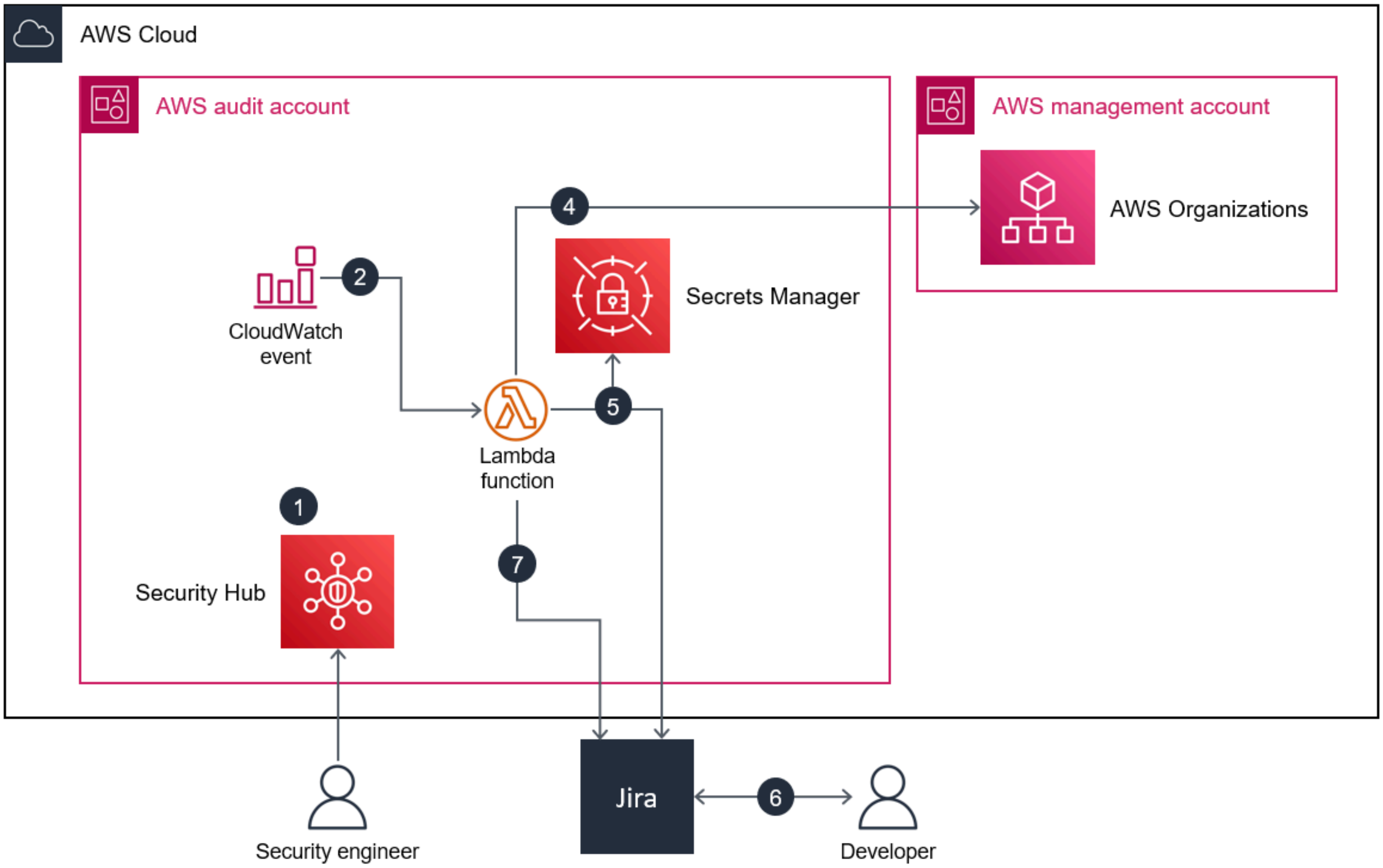
- Administrator permissions in Jira
- One of the following Jira tokens:
  - For Jira Enterprise, a personal access token (PAT). For more information, see Using Personal Access Tokens (Atlassian support).
  - For Jira Cloud, a Jira API token. For more information, see Manage API tokens (Atlassian support).

Architecture

This section illustrates the architecture of the solution in various scenarios, such as when the developer and security engineer decide to accept the risk or decide to fix the issue.

Scenario 1: Developer addresses the issue

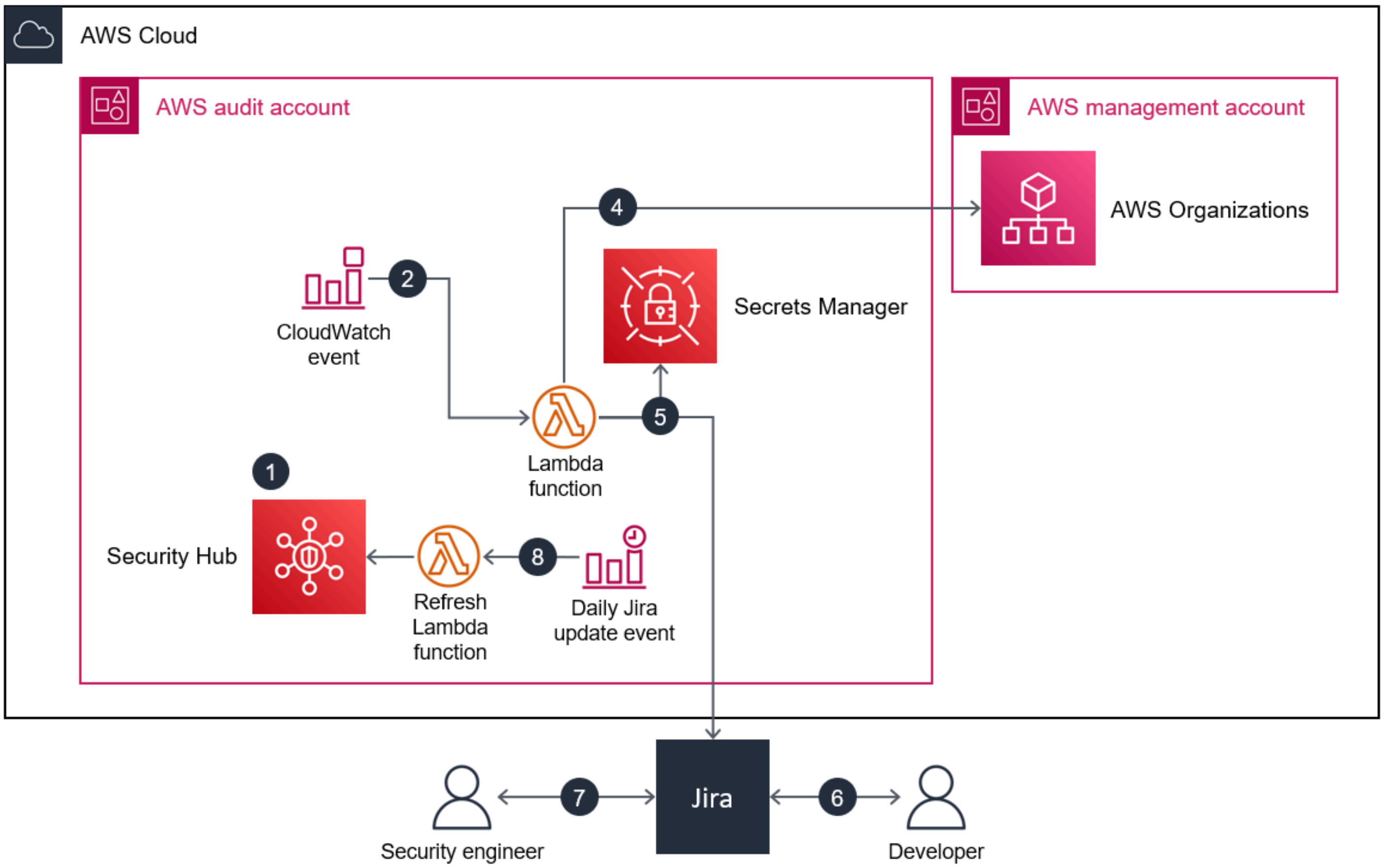
1. Security Hub generates a finding against a specified security control, such as those in the AWS Foundational Security Best Practices standard.
2. An Amazon CloudWatch event associated with the finding and the `CreateJIRA` action initiates an AWS Lambda function.
3. The Lambda function uses its configuration file and the finding's `GeneratorId` field to evaluate whether it should escalate the finding.
4. The Lambda function determines the finding should be escalated, it obtains the `SecurityContactID` account tag from AWS Organizations in the AWS management account. This ID is associated with the developer and is used as the assignee ID for the Jira ticket.
5. The Lambda function uses the credentials stored in AWS Secrets Manager to create a ticket in Jira. Jira notifies the developer.
6. The developer addresses the underlying security finding and, in Jira, changes the status of the ticket to `TEST FIX`.
7. Security Hub updates the finding as `ARCHIVED`, and a new event is generated. This event causes the Lambda function to automatically close the Jira ticket.



Scenario 2: Developer decides to accept the risk

1. Security Hub generates a finding against a specified security control, such as those in the [AWS Foundational Security Best Practices standard](https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-standards-fsbp.html).
2. A CloudWatch event associated with the finding and the `CreateJIRA` action initiates a Lambda function.
3. The Lambda function uses its configuration file and the finding's `GeneratorId` field to evaluate whether it should escalate the finding.
4. The Lambda function determines the finding should be escalated, it obtains the `SecurityContactID` account tag from AWS Organizations in the AWS management account. This ID is associated with the developer and is used as the assignee ID for the Jira ticket.
5. The Lambda function uses the credentials stored in Secrets Manager to create a ticket in Jira. Jira notifies the developer.
6. The developer decides to accept the risk and, in Jira, changes the status of the ticket to `AWAITING RISK ACCEPTANCE`.

7. The security engineer reviews the request and finds the business justification appropriate. The security engineer changes the status of the Jira ticket to `ACCEPTED RISK`. This closes the Jira ticket.
8. A CloudWatch daily event initiates the refresh Lambda function, which identifies closed Jira tickets and updates their related Security Hub findings as `SUPPRESSED`.



Tools

AWS services

- [AWS CloudFormation](#) helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- [Amazon CloudWatch Events](#) helps you monitor system events for your AWS resources by using rules to match events and route them to functions or streams.

- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- AWS Security Hub provides a comprehensive view of your security state in AWS. It also helps you check your AWS environment against security industry standards and best practices.

Code repository

The code for this pattern is available on GitHub, in the aws-securityhub-jira-software-integration repository. It includes the sample code and Jira workflow for this solution.

Epics

Configure Jira

Task	Description	Skills required
Import the workflow.	As an administrator in Jira, import the `issue-workflow.xml` file to your Jira Data Center instance. If you use Jira Cloud, you need to create the workflow according to the `assets/jira-cloud-transitions.png` and `assets/jira-cloud-status.png` files. Files can be found in the aws-securityhub-jira-software-integration repository in GitHub. For instructions,	Jira administrator

Task	Description	Skills required
	see Using XML to create a workflow (Jira documenta tion).	
Activate and assign the workflow.	Workflows are inactive until you assign them to a workflow scheme. You then assign the workflow scheme to a project. 1. For your project, make sure you have identified an issue type scheme for the project. You can create a new issue type or select from an existing one, such as Bug. 2. Assign the imported workflow to a workflow scheme according to the instructions in Activate a workflow (Jira documenta tion). 3. Assign the workflow scheme to a project according to the instructi ons in Associate a workflow scheme with a project (Jira documentation).	Jira administrator

Set up the solution parameters

Task	Description	Skills required
Configure the solution parameters.	1. In the **conf** folder, open `params_prod.shfile`. 2. Provide values for the following parameters: • `ORG_ACCOUNT_ID` – The account ID for your AWS Organizations management account. The solution reads account tags and assigns tickets to the specific security contacts defined in those AWS account tags. • `ORG_ROLE` – The name of the IAM role used to access the AWS Organizations management account. This role must have `OrganizationsReadOnlyAccess` permissions. • `EXTERNAL_ID` – An optional parameter if you are using an external ID to assume the IAM role defined in `ORG_ROLE`. For more information, see [How to use an external ID](#) (IAM documentation).	AWS systems administrator

Task	Description	Skills required
	• JIRA_DEFAULT_ASSIGNEE – This is the Jira default assignee for all security issues. This default assignee is used in case account is not tagged properly or role cannot be assumed. • JIRA_INSTANCE – The HTTPS address for your Jira endpoint in the following format: team-<team-id>.atlassian.net/ • JIRA_PROJECT_KEY – The name of the Jira project key used to create tickets, such as SEC or TEST. This project must already exist in Jira. • ISSUE_TYPE – The name of the issue type scheme assigned to the project in Jira, such as Bug or Security Issue. • REGIONS – List of AWS Region codes where you want to deploy this solution, such as eu-west-1 .	

Task	Description	Skills required
	3. Save and close the solution parameter file.	
Identify the findings you want to automate.	1. Open the [Security Hub console](). 2. In the Security Hub navigation pane, choose **Findings**. 3. Choose the finding title. 4. Choose the finding ID. This displays the complete JSON for the finding. 5. In the JSON, copy the string in the `Generator Id` field. This value is in [AWS Security Finding Format (ASFF)](). For example, `aws-foundational-security-best-practices/v/1.0.0/S3.1` corresponds to findings from the security control *S3.1 S3 Block Public Access setting should be enabled*. 6. Repeat these steps until you have copied all of the `GeneratorID` values for any findings you want to automate.	

Task	Description	Skills required

Add the findings to the configuration file.	1. In **src/code**, open the `config.jsonconfig` file.
	2. Paste the `GeneratorID` values you retrieved in the previous story into the `default` parameter, and use commas to separate each ID.
	3. Save and close the configuration file.
	
	The following code example shows automating the `aws-foundational-security-best-practices/v/1.0.0/SNS.1` and `aws-foundational-security-best-practices/v/1.0.0/S3.1` findings.
	
	`{`
	`    "Controls" : {`
	`        "eu-west-1": [`
	`         "arn:aws:securityhub:::ruleset/cis-aws-foundations-benchmark/v/1.2.0/rule/1.22"`
	`     ],`
	`        "default": [`
	`aws-foundational-security-best-practices/v/1.0.0/SNS.1,`
		AWS systems administrator

Task	Description	Skills required
	``` aws-foundational-security-best-practices/v/1.0.0/S3.1] } } ``` **Note** You can choose to automate different findings for each AWS Region. A good practice to help prevent duplicated findings is to select a single Region to automate creation of controls related to IAM.	

Deploy the integration

Task	Description	Skills required
Deploy the integration.	In a command line terminal, enter the following command: `./deploy.sh prod`	AWS systems administrator
Upload Jira credentials to Secrets Manager.	1. Open the [Secrets Manager console]().	AWS systems administrator

Task	Description	Skills required
	2. Under **Secrets**, choose **Store a new secret**. 3. For **Secret type**, choose **Other type of secret**. 4. If you are using Jira Enterprise, for **Key/value pairs**, do the following: • In the first row, enter `auth` in the key box, and then enter `token_auth` in the value box. • Add a second row, enter `token` in the key box, and then enter your personal access token in the value box. If you are using Jira Cloud, for **Key/value pairs**, do the following: • In the first row, enter `auth` in the key box, and then enter `basic_auth` in the value box. • Add a second row, enter `token` in the key box, and then enter your API token in the value box. • Add a third row, enter `email` in the key box, and then enter your email address in the value box.	

Task	Description	Skills required
	5. Choose **Next**. 6. For **Secret name**, enter `Jira-Token` , and then at the bottom of the page, choose **Next**. 7. On the **Secret rotation** page, keep **Disable automatic rotation**, and then at the bottom of the page, choose **Next**. 8. On the **Review** page, review the secret details, and then choose **Store**.	

Task	Description	Skills required
Create the Security Hub custom action.	1. For each AWS Region, in the AWS Command Line Interface (AWS CLI), use the [create-action-target](#) command to create a Security Hub custom action named `CreateJiraIssue`. `aws securityhub create-action-target --name "CreateJiraIssue" \ --description "Create ticket in JIRA" \ --id "CreateJiraIssue" --region $<aws-region>` 2. Open the [Security Hub console](#). 3. In the Security Hub navigation pane, choose **Findings**. 4. In the list of findings, select the findings you want to escalate. 5. In the **Actions** menu, choose `CreateJiraIssue`.	AWS systems administrator

Related resources

- AWS Service Management Connector for Jira Service Management
- AWS Foundational Security Best Practices standard

Additional information

Example of a Jira ticket

When a specified Security Hub finding occurs, this solution automatically creates a Jira ticket. The ticket includes the following information:

- **Title** – The title identifies the security issue in the following format:

```
AWS Security Issue :: <AWS account ID> :: <Security Hub finding title>
```

- **Description** – The description section of the ticket describes the security control associated with the finding, includes a link to the finding in the Security Hub console, and provides a short description of how to handle the security issue in the Jira workflow.

The following is an example of an automatically generated Jira ticket.

Title	AWS Security Issue :: 012345678912 :: Lambda.1 Lambda function policies should prohibit public access.
Description	**What is the problem?** We detected a security finding within the AWS account 012345678 912 you are responsible for. This control checks whether the AWS Lambda function policy attached to the Lambda resource prohibits public access. If the Lambda function policy allows public access, the control fails. <Link to Security Hub finding>

What do I need to do with the ticket?

- Access the account and verify the configura tion. Acknowledge working on ticket by moving it to "Allocated for Fix". Once fixed, moved to test fix so security validates the issue is addressed.
- If you think risk should be accepted, move it to "Awaiting Risk acceptance". This will require review by a security engineer.
- If you think is a false positive, transitio n it to "Mark as False Positive". This will get reviewed by a security engineer and reopened/closed accordingly.

Build a pipeline for hardened container images using EC2 Image Builder and Terraform

Mike Saintcross and Andrew Ranes, Amazon Web Services

Summary

This pattern builds an EC2 Image Builder pipeline that produces a hardened Amazon Linux 2 base container image. Terraform is used as an infrastructure as code (IaC) tool to configure and provision the infrastructure that is used to create hardened container images. The recipe helps you deploy a Docker-based Amazon Linux 2 container image that has been hardened according to Red Hat Enterprise Linux (RHEL) 7 STIG Version 3 Release 7 – Medium. (See STIG-Build-Linux-Medium version 2022.2.1 in the *Linux STIG components* section of the EC2 Image Builder documentation.) This is referred to as a *golden* container image.

The build includes two Amazon EventBridge rules. One rule starts the container image pipeline when the Amazon Inspector finding is **High** or **Critical** so that non-secure images are replaced. This rule requires both Amazon Inspector and Amazon Elastic Container Registry (Amazon ECR) enhanced scanning to be enabled. The other rule sends notifications to an Amazon Simple Queue Service (Amazon SQS) queue after a successful image push to the Amazon ECR repository, to help you use the latest container images.

> **Note**
>
> Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs.

Prerequisites and limitations

Prerequisites

- An AWS account that you can deploy the infrastructure in.
- AWS Command Line Interface (AWS CLI) installed for setting your AWS credentials for local deployment.
- Terraform downloaded and set up by following the instructions in the Terraform documentation.
- Git (if you're provisioning from a local machine).

- A role within the AWS account that you can use to create AWS resources.
- All variables defined in the .tfvars file. Or, you can define all variables when you apply the Terraform configuration.

Limitations

- This solution creates an Amazon Virtual Private Cloud (Amazon VPC) infrastructure that includes a NAT gateway and an internet gateway for internet connectivity from its private subnet. You cannot use VPC endpoints, because the bootstrap process by AWS Task Orchestrator and Executor (AWSTOE) installs AWS CLI version 2 from the internet.

Product versions

- Amazon Linux 2
- AWS CLI version 1.1 or later

Architecture

Target technology stack

This pattern creates 43 resources, including:

- Two Amazon Simple Storage Service (Amazon S3) buckets: one for the pipeline component files and one for server access and Amazon VPC flow logs
- An Amazon ECR repository
- A virtual private cloud (VPC) that contains a public subnet, a private subnet, route tables, a NAT gateway, and an internet gateway
- An EC2 Image Builder pipeline, recipe, and components
- A container image
- An AWS Key Management Service (AWS KMS) key for image encryption
- An SQS queue
- Three roles: one to run the EC2 Image Builder pipeline, one instance profile for EC2 Image Builder, and one for EventBridge rules
- Two EventBridge rules

Terraform module structure

For the source code, see the GitHub repository [Terraform EC2 Image Builder Container Hardening Pipeline](https://github.com/aws-samples/terraform-ec2-image-builder-container-hardening-pipeline).

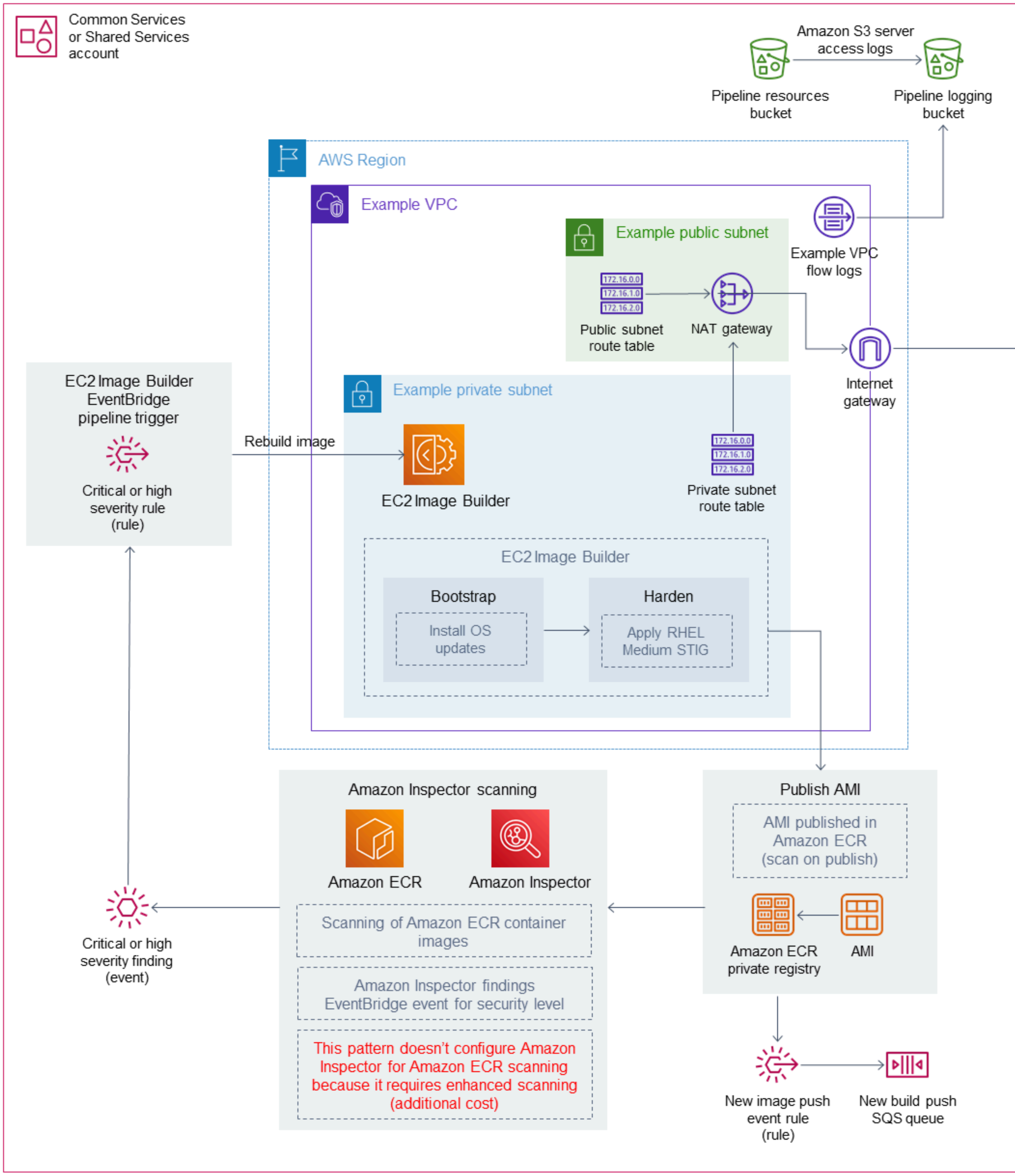
```
### components.tf
### config.tf
### dist-config.tf
### files
#   ###assumption-policy.json
### hardening-pipeline.tfvars
### image.tf
### infr-config.tf
### infra-network-config.tf
### kms-key.tf
### main.tf
### outputs.tf
### pipeline.tf
### recipes.tf
### roles.tf
### sec-groups.tf
### trigger-build.tf
### variables.tf
```

Module details

- `components.tf` contains an Amazon S3 upload resource to upload the contents of the `/files` directory. You can also modularly add custom component YAML files here as well.
- `/files` contains the `.yml` files that define the components used in `components.tf`.
- `image.tf` contains the definitions for the base image operating system. This is where you can modify the definitions for a different base image pipeline.
- `infr-config.tf` and `dist-config.tf` contain the resources for the minimum AWS infrastructure needed to spin up and distribute the image.
- `infra-network-config.tf` contains the minimum VPC infrastructure to deploy the container image into.
- `hardening-pipeline.tfvars` contains the Terraform variables to be used at apply time.
- `pipeline.tf` creates and manages an EC2 Image Builder pipeline in Terraform.
- `recipes.tf` is where you can specify different mixtures of components to create container recipes.

- `roles.tf` contains the AWS Identity and Access Management (IAM) policy definitions for the Amazon Elastic Compute Cloud (Amazon EC2) instance profile and pipeline deployment role.
- `trigger-build.tf` contains the EventBridge rules and SQS queue resources.

Target architecture



The diagram illustrates the following workflow:

1. EC2 Image Builder builds a container image by using the defined recipe, which installs operating system updates and applies the RHEL Medium STIG to the Amazon Linux 2 base image.

2. The hardened image is published to a private Amazon ECR registry, and an EventBridge rule sends a message to an SQS queue when the image has been published successfully.
3. If Amazon Inspector is configured for enhanced scanning, it scans the Amazon ECR registry.
4. If Amazon Inspector generates a **Critical** or **High** severity finding for the image, an EventBridge rule triggers the EC2 Image Builder pipeline to run again and publish a newly hardened image.

Automation and scale

- This pattern describes how to provision the infrastructure and build the pipeline on your computer. However, it is intended to be used at scale. Instead of deploying the Terraform modules locally, you can use them in a multi-account environment, such as an AWS Control Tower with Account Factory for Terraform environment. In that case, you should use a backend state S3 bucket to manage Terraform state files instead of managing the configuration state locally.
- For scaled use, deploy the solution to one central account, such as a Shared Services or Common Services account, from a Control Tower or landing zone account model, and grant consumer accounts permission to access the Amazon ECR repository and AWS KMS key. For more information about the setup, see the re:Post article How can I allow a secondary account to push or pull images in my Amazon ECR image repository? For example, in an account vending machine or Account Factory for Terraform, add permissions to each account baseline or account customization baseline to provide access to that Amazon ECR repository and encryption key.
- After the container image pipeline is deployed, you can modify it by using EC2 Image Builder features such as components, which help you package more components into the Docker build.
- The AWS KMS key that is used to encrypt the container image should be shared across the accounts that the image is intended to be used in.
- You can add support for other images by duplicating the entire Terraform module and modifying the following `recipes.tf` attributes:
  - Modify `parent_image = "amazonlinux:latest"` to another image type.
  - Modify `repository_name` to point to an existing Amazon ECR repository. This creates another pipeline that deploys a different parent image type to your existing Amazon ECR repository.

Tools

Tools

- Terraform (IaC provisioning)
- Git (if provisioning locally)
- AWS CLI version 1 or version 2 (if provisioning locally)

Code

The code for this pattern is in the GitHub repository Terraform EC2 Image Builder Container Hardening Pipeline. To use the sample code, follow the instructions in the next section.

Epics

Provision the infrastructure

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Set up local credentials.</td><td>Set up your AWS temporary credentials.

1. See if the AWS CLI is installed:
<pre>$ aws --version
aws-cli/1.16.249
 Python/3.6.8...</pre>• AWS CLI version should be 1.1 or later.
• If the command isn't found, install the AWS CLI.

2. Run <code>aws configure</code> and provide the following values:
<pre>$ aws configure
AWS Access Key ID
 [************xxxx</pre></td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
	```]: <Your AWS access key ID> AWS Secret Access Key [**************xxxx]: <Your AWS secret access key> Default region name: [us-east-1]: <Your desired Region for deployment> Default output format [None]: <Your desired output format> ```	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Clone the repository.</td><td>
1. Clone the repository that's provided with this pattern. You can use HTTPS or Secure Shell (SSH).
HTTPS:
<pre><code>git clone https://github.com/aws-samples/terraform-ec2-image-builder-container-hardening-pipeline</code></pre>
SSH:
<pre><code>git clone git@github.com:aws-samples/terraform-ec2-image-builder-container-hardening-pipeline.git</code></pre>
2. Navigate to your local directory that contains this solution:
<pre><code>cd terraform-ec2-image-builder-container-hardening-pipeline</code></pre>
</td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required

Update variables.	Update the variables in the `hardening-pipeline .tfvars` file to match your environment and your desired configuration. You must provide your own `account_id` . However, you should also modify the rest of the variables to fit your desired deployment. All variables are required.
	
	`account_id = "<DEPLOYMENT-ACCOUNT-ID>"`
	`aws_region = "us-east-1"`
	`vpc_name = "example-hardening-pipeline-vpc"`
	`kms_key_alias = "image-builder-container-key"`
	`ec2_iam_role_name = "example-hardening-instance-role"`
	`hardening_pipeline_role_name = "example-hardening-pipeline-role"`
	`aws_s3_ami_resources_bucket = "example-hardening-ami-resources-bucket-0123"`
	`image_name = "example-hardening-al2-container-image"`
		AWS DevOps

Task	Description	Skills required

	`ecr_name = "example-hardening-container-repo"`
	`recipe_version = "1.0.0"`
	`ebs_root_vol_size = 10`
	
	Here's a description of each variable:
	
	• `account_id` – The AWS account number that you want to deploy the solution into.
	• `aws_region` – The AWS Region that you want to deploy the solution into.
	• `vpc_name` – The name for your VPC infrastructure.
	• `kms_key_alias` – The AWS KMS key name to be used by the EC2 Image Builder infrastructure configuration.
	• `ec2_iam_role_name` – The name for the role that will be used as the EC2 instance profile.
	• `hardening_pipeline_role_name` – The name for the role that will be used to deploy the hardening pipeline.
	• `aws_s3_ami_resources_bucket` – The name	

Task	Description	Skills required
	for an S3 bucket that will host all files necessary to build the pipeline and container images. • `image_name` – The container image name. This value must be between 3 and 50 characters and should contain alphanumeric characters and hyphens only. • `ecr_name` – The name of the Amazon ECR registry to store the container images in. • `recipe_version` – The version of the image recipe. The default value is 1.0.0. • `ebs_root_vol_size` – The size (in gigabytes) of the Amazon Elastic Block Store (Amazon EBS) root volume. The default value is 10 gigabytes.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Initialize Terraform.</td><td>After you update your variable values, you can initialize the Terraform configuration directory. Initializing a configuration directory downloads and installs the AWS provider, which is defined in the configuration.<pre><code>terraform init</code></pre>You should see a message that says Terraform has been successfully initialized and identifies the version of the provider that was installed.</td><td>AWS DevOps</td></tr>
<tr><td>Deploy the infrastructure and create a container image.</td><td>Use the following command to initialize, validate, and apply the Terraform modules to the environment by using the variables defined in your <code>.tfvars</code> file:<pre><code>terraform init &&
 terraform validate &&
 terraform apply -var-
file *.tfvars -auto-app
rove</code></pre></td><td>AWS DevOps</td></tr>
</table>

Task	Description	Skills required
Customize the container.	You can create a new version of a container recipe after EC2 Image Builder deploys the pipeline and initial recipe. You can add any of the 31+ components available within EC2 Image Builder to customize the container build. For more information, see the *Components* section of [Create a new version of a container recipe](#) in the EC2 Image Builder documenta tion.	AWS administrator

Validate resources

Task	Description	Skills required

Validate AWS infrastructure provisioning.	After you have successfu lly completed your first Terraform `apply` command, if you're provisioning locally, you should see this snippet in your local machine's terminal:
	
	`Apply complete!`
	`Resources: 43 added, 0 changed, 0 destroyed.`
		AWS DevOps

Validate individual AWS infrastructure resources.	To validate the individual resources that were deployed, if you're provisioning locally,	AWS DevOps

Task	Description	Skills required
	you can run the following command: `terraform state list` This command returns a list of 43 resources.	

Remove resources

Task	Description	Skills required
Remove the infrastructure and container image.	When you’ve finished working with your Terraform configuration, you can run the following command to remove resources: `terraform init && terraform validate && terraform destroy -var-file *.tfvars -auto-approve`	AWS DevOps

Troubleshooting

Issue	Solution
Error validating provider credentials	When you run the Terraform `apply` or `destroy` command from your local machine, you might encounter an error similar to the following:

Issue	Solution
	`Error: configuring Terraform AWS Provider: error validating provider credentials: error calling sts:GetCallerIdentity: operation error STS: GetCallerIdentity, https response error StatusCode: 403, RequestID: 123456a9-fbc1-40ed-b8d8-513d0133ba7f, api error InvalidClientTokenId: The security token included in the request is invalid.` This error is caused by the expiration of the security token for the credentials used in your local machine's configuration. To resolve the error, see Set and view configuration settings in the AWS CLI documentation.

Related resources

- Terraform EC2 Image Builder Container Hardening Pipeline (GitHub repository)
- EC2 Image Builder documentation
- AWS Control Tower Account Factory for Terraform (AWS blog post)
- Backend state S3 bucket (Terraform documentation)
- Installing or updating the latest version of the AWS CLI (AWS CLI documentation)
- Download Terraform

Centralize IAM access key management in AWS Organizations by using Terraform

Aarti Rajput, Chintamani Aphale, T.V.R.L.Phani Kumar Dadi, Pratap Kumar Nanda, Pradip kumar Pandey, and Mayuri Shinde, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more

Enforcing security rules for keys and passwords is an essential task for every organization. One important rule is to rotate AWS Identity and Access Management (IAM) keys at regular intervals to enforce security. AWS access keys are generally created and configured locally whenever teams want to access AWS from the AWS Command Line Interface (AWS CLI) or from applications outside AWS. To maintain strong security across the organization, old security keys must be changed or deleted after the requirement has been met or at regular intervals. The process of managing key rotations across multiple accounts in an organization is time-consuming and tedious. This pattern helps you automate the rotation process by using Account Factory for Terraform (AFT) and AWS services.

The pattern provides these benefits:

- Manages your access key IDs and secret access keys across all the accounts in your organization from a central location.
- Automatically rotates the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` environment variables.
- Enforces renewal if user credentials are compromised.

The pattern uses Terraform to deploy AWS Lambda functions, Amazon EventBridge rules, and IAM roles. An EventBridge rule runs at regular intervals and calls a Lambda function that lists all user access keys based on when they were created. Additional Lambda functions create a new access key ID and secret access key, if the previous key is older than the rotation period you define (for example, 45 days), and notify a security administrator by using Amazon Simple Notification Service (Amazon SNS) and Amazon Simple Email Service (Amazon SES). Secrets are created in AWS Secrets Manager for that user, the old secret access key is stored in Secrets Manager, and permissions for accessing the old key are configured. To ensure that the old access key is no longer used, it is disabled after an inactive period (for example, 60 days, which would be 15 days after the keys were rotated in our example). After an inactive buffer period (for example, 90 days, or 45 days after the keys were rotated in our example), the old access keys are deleted from AWS Secrets Manager. For a detailed architecture and workflow, see the Architecture section.

Prerequisites and limitations

- A landing zone for your organization that's built by using AWS Control Tower (version 3.1 or later)
- Account Factory for Terraform (AFT) configured with three accounts:
  - Organization management account manages the entire organization from a central location.
  - AFT management account hosts the Terraform pipeline and deploys the infrastructure into the deployment account.
  - Deployment account deploys this complete solution and manages IAM keys from a central location.
- Terraform version 0.15.0 or later for provisioning the infrastructure in the deployment account.
- An email address that's configured in Amazon Simple Email Service (Amazon SES).
- (Recommended) To enhance security, deploy this solution inside a private subnet (deployment account) within a virtual private cloud (VPC). You can provide the details of the VPC and subnet when you customize the variables (see *Customize parameters for the code pipeline* in the Epics section).

Architecture

AFT repositories

This pattern uses Account Factory for Terraform (AFT) to create all required AWS resources and the code pipeline to deploy the resources in a deployment account. The code pipeline runs in two repositories:

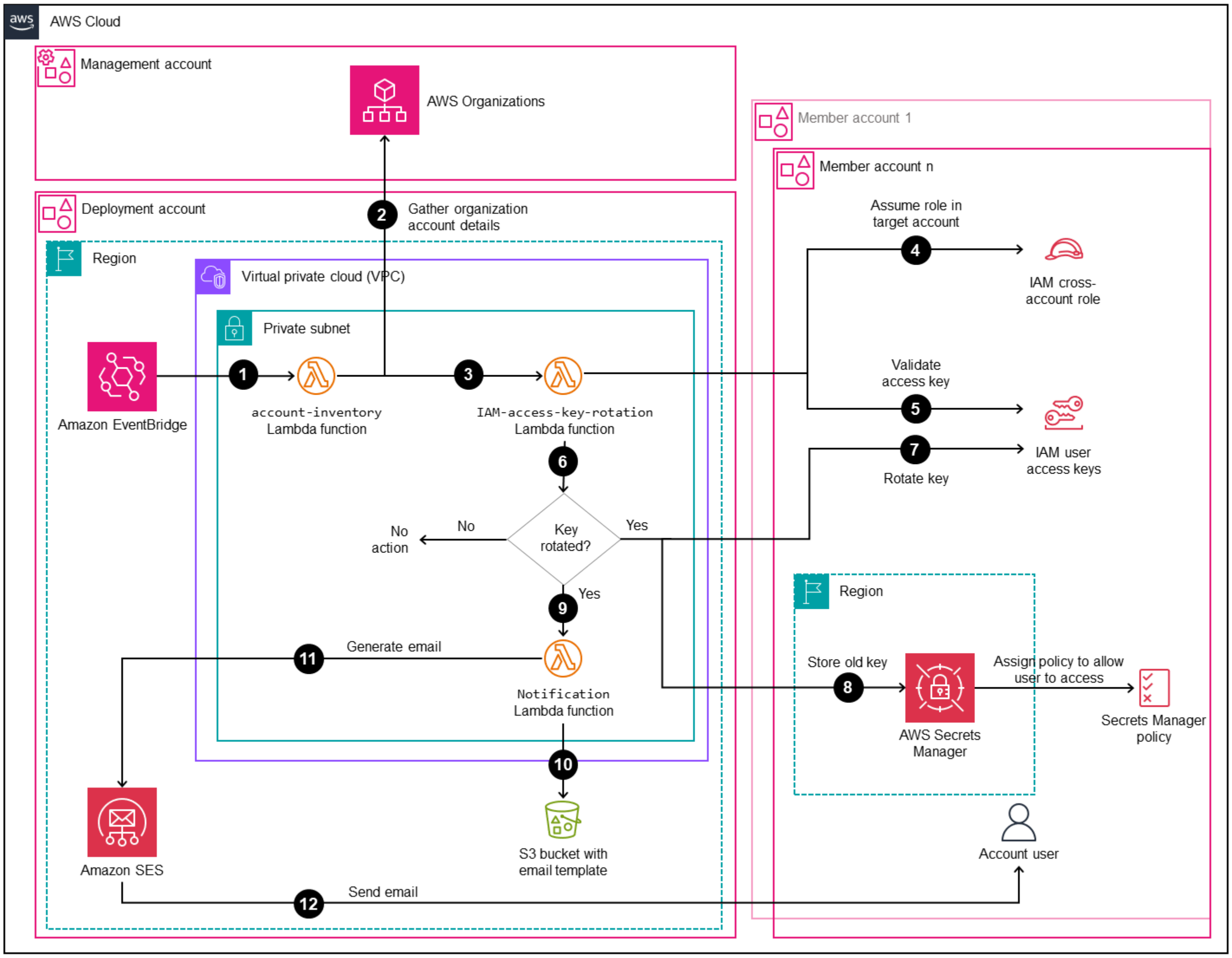
- **Global customization** contains Terraform code that will run across all accounts registered with AFT.
- **Account customizations** contains Terraform code that will run in the deployment account.

Resource details

AWS CodePipeline jobs create the following resources in the deployment account:

- AWS EventBridge rule and configured rule
- `account-inventory` Lambda function
- `IAM-access-key-rotation` Lambda function
- `Notification` Lambda function
- Amazon Simple Storage Service (Amazon S3) bucket that contains an email template
- Required IAM policy

Architecture



The diagram illustrates the following:

1. An EventBridge rule calls the `account-inventory` Lambda function every 24 hours.
2. The `account-inventory` Lambda function queries AWS Organizations for a list of all AWS account IDs, account names, and account emails.
3. The `account-inventory` Lambda function initiates an `IAM-access-key-auto-rotation` Lambda function for each AWS account and passes the metadata to it for additional processing.
4. The `IAM-access-key-auto-rotation` Lambda function uses an assumed IAM role to access the AWS account. The Lambda script runs an audit against all users and their IAM access keys in the account.
5. The IAM key rotation threshold (rotation period) is configured as an environment variable when the `IAM-access-key-auto-rotation` Lambda function is deployed. If the rotation period is modified, the `IAM-access-key-auto-rotation` Lambda function is redeployed with an updated environment variable. You can configure parameters to set the rotation period, the inactive period for old keys, and the inactive buffer after which old keys will be deleted (see *Customize parameters for the code pipeline* in the Epics section).
6. The `IAM-access-key-auto-rotation` Lambda function validates the age of the access key based on its configuration. If the IAM access key's age hasn't exceeded the rotation period you defined, the Lambda function takes no further action.
7. If the IAM access key's age has exceeded the rotation period you defined, the `IAM-access-key-auto-rotation` Lambda function creates a new key and rotates the existing key.
8. The Lambda function saves the old key in Secrets Manager and limits permissions to the user whose access keys deviated from security standards. The Lambda function also creates a resource-based policy that allows only the specified IAM principal to access and retrieve the secret.
9. The `IAM-access-key-rotation` Lambda function calls the `Notification` Lambda function.
10. The `Notification` Lambda function queries the S3 bucket for an email template and dynamically generates email messages with the relevant activity metadata.
11. The `Notification` Lambda function calls Amazon SES for further action.
12. Amazon SES sends email to the account owner's email address with the relevant information.

Tools

AWS services

- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them. This patern requires IAM roles and permissions.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically.
- Amazon Simple Email Service (Amazon SES) helps you send and receive emails by using your own email addresses and domains.

Other tools

- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The instructions and code for this pattern are available in the GitHub IAM access key rotation repository. You can deploy the code in the AWS Control Tower central deployment account to manage key rotation from a central location.

Best practices

- For IAM, see security best practices in the IAM documentation.
- For key rotation, see guidelines for updating access keys in the IAM documentation.

Epics

Set up source files

Task	Description	Skills required
Clone the repository.	1. Clone the IAM access key rotation GitHub repository:	DevOps engineer

Task	Description	Skills required

	`$ git clone https://github.com/aws-samples/centralized-iam-key-management-aws-organizations-terraform.git`
	
	2. Confirm that your local copy of the repository contains three folders:
	
	`$ cd Iam-Access-keys-Rotation`
	`$ ls`
	`org-account-customization`
	`global-account-customization`
	`account-customization`	

Configure accounts

Task	Description	Skills required
Configure the bootstrapping account.	As part of the AFT bootstrapping process, you should have a folder called `aft-bootstrap` on your local machine. 1. Copy all Terraform files manually from your local GitHub org-account-customization folder to your `aft-bootstrap` folder.	DevOps engineer

Task	Description	Skills required

	2. Run Terraform commands to configure the global cross-account role in the AWS Control Tower management account:
	```$ cd aft-bootstrap```
	```$ terraform init```
	```$ terraform apply –auto-approve```	

Configure global customizations.	As part of the [AFT folder](#) setup, you should have a folder called `aft-global-customizations` on your local machine.
	1. Manually copy all Terraform files from your local GitHub [global-account-customization](#) folder to your `aft-global-customizations/terraform` folder.
	2. Push the code to AWS CodeCommit:
	```$ git add *```
	```$ git commit -m "message"```
	```$ git push```
		DevOps engineer

Task	Description	Skills required

Configure account customizations.	As part of the [AFT folder setup](#), you have be a folder called `aft-account-customizations` on your local machine.
	
	1. Create a folder with your vended account number.
	2. Manually copy manual all Terraform files from your local GitHub [account-customization](#) folder to your `aft-account-customizations/<vended account>/terraform` folder.
	3. Push the code to AWS CodeCommit:
	
	`$ git add *`
	`$ git commit -m "message"`
	`$ git push`
		DevOps engineer

Customize parameters for the code pipeline

Task	Description	Skills required
Customize non-Terraform code pipeline parameters for all accounts.	Create a file called `input.auto.tfvars` in the `aft-global-customizations/terraform/` folder and provide the required input data. See the	DevOps engineer

Task	Description	Skills required
	file in the GitHub repository for default values.	

Task	Description	Skills required
Customize code pipeline parameters for the deployment account.	Create a file called `input.auto.tfvars` in the `aft-account-customizations/<AccountName>/terraform/` folder and push the code to AWS CodeCommit. Pushing code to AWS CodeCommit automatically initiates the code pipeline. Specify values for parameters based on your organization's requirements, including the following (see [the file in the Github repository](#) for default values): • `s3_bucket_name` – A unique bucket name for the email template. • `s3_bucket_prefix` – A folder name inside the S3 bucket. • `admin_email_address` – The email address for the administrator who should receive the notification. • `org_list_account` – The account number of the management account. • `rotation_period` – The number of days after which	DevOps engineer

Task	Description	Skills required
	a key should be rotated from active to inactive. • `inactive_period` – The number of days after which rotated keys should be deactivated. This value must be greater than the value of `rotation_period`. • `inactive_buffer` – The grace period between the rotation and the deactivation of a key. • `recovery_grace_period` – The grace period between the deactivation and the deletion of a key. • `dry_run_flag` – Set to **true** if you want to send a notification to the administrator for testing purposes, without rotating keys. • `store_secrets_in_central_account` – Set to **true** if you want to store the secret in the deployment account. If the variable is set to **false** (default), the secret will be stored in the member account. • `credential_replication_region` – The	

Task	Description	Skills required
	AWS Region where you want to deploy the Lambda function and the S3 buckets for the email template. • `run_lambda_in_vpc` – Set to **true** to run the Lambda function inside the VPC. • `vpc_id` – The VPC ID of the deployment account, if you want to run the Lambda function inside the VPC. • `vpc_cidr` – The CIDR range for the deployment account. • `subnet_id` – The subnet IDs for the deployment account. • `create_smtp_endpoint` – Set to **true** if you want to enable the email endpoint.	

Validate key rotation

Task	Description	Skills required
Validate the solution.	1. From the AWS Management Console, sign in to the deployment account. 2. Open the [IAM console](https://console.aws.amazon.com/iam/) and check whether user credentials (access key IDs	DevOps engineer

Task	Description	Skills required
	and secret keys) are being rotated as specified. 3. After an IAM key has been rotated, confirm the following: • The old value is stored in AWS Secrets Manager. • The secret name is in the format `Account_<account ID>_User_<username>_AccessKey` . • The user you specified in the `admin_email_address` parameter receives an email notification about the key rotation.	

Extend the solution

Task	Description	Skills required
Customize the email notification date.	If you want to send email notifications on a specific day before you disable the access key, you can update the `IAM-access-key-auto-rotation` Lambda function with those changes: 1. Define a variable called `notify-period` .	DevOps engineer

Task	Description	Skills required
	2. Add an `if` condition in `main.py` before you deactivate the key: `If (keyage>rotation-period-notify-period){ send_to_notifier(context, aws_account_id, account_name, resource_owner, resource_actions[resource_owner], dryrun, config.emailTemplateAudit) }`	

Troubleshooting

Issue	Solution
The `account-inventory` Lambda job fails with `AccessDenied` while listing accounts.	If you encounter this issue, you must validate permissions: 1. Log in to the newly vended account, open the [Amazon CloudWatch console](https://console.aws.amazon.com/cloudwatch/), and and then view the CloudWatch log group `/aws/lambda/account-inventory-lambda`. 2. In the latest CloudWatch logs, identify the account number that is causing the access denied issue. 3. Log in to the AWS Control Tower management account and confirm that the role `allow-list-account` was created.

Issue	Solution
	4. If the role doesn't exist, rerun the Terraform code by using the `terraform apply` command. 5. Choose the **Trusted Account** tab and validate that the same account is trusted.

Related resources

- Terraform Recommended Practices (Terraform documentation)
- Security best practices in IAM (IAM documentation)
- Best practices for key rotation (IAM documentation)

Check an Amazon CloudFront distribution for access logging, HTTPS, and TLS version

SaiJeevan Devireddy and Bijesh Bal, Amazon Web Services

Summary

This pattern checks an Amazon CloudFront distribution to make sure that it uses HTTPS, uses Transport Layer Security (TLS) version 1.2 or later, and has access logging enabled. CloudFront is a service provided by Amazon Web Services (AWS) that speeds up the distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called *edge locations*. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

This pattern provides an AWS Lambda function that is initiated when Amazon CloudWatch Events detects the CloudFront API call CreateDistribution, CreateDistributionWithTags, or UpdateDistribution. The custom logic in the Lambda function evaluates all CloudFront distributions that were created or updated in the AWS account. It sends a violation notification by using Amazon Simple Notification Service (Amazon SNS) if it detects the following violations:

- Global checks:
  - Custom certificate doesn't use TLS version 1.2
  - Logging is disabled for distribution
- Origin checks:
  - Origin isn't configured with TLS version 1.2
  - Communication with origin is allowed on a protocol other than HTTPS
- Behavior checks:
  - Default behavior communication is allowed on a protocol other than HTTPS
  - Custom behavior communication is allowed on a protocol other than HTTPS

Prerequisites and limitations

Prerequisites

- An active AWS account
- An email address where you want to receive the violation notifications

Limitations

- This security control doesn't check for existing Cloudfront distributions unless an update has been made to the distribution.
- CloudFront is considered a global service and isn't tied to a specific AWS Region. However, Amazon CloudWatch Logs and AWS Cloudtrail API logging for global services occur in the US East (N. Virginia) Region (`us-east-1`). Therefore, this security control for CloudFront must be deployed and maintained in `us-east-1`. This single deployment monitors all distributions for CloudFront. Do not deploy the security control in any other AWS Regions. (Deployment in other Regions will result in a failure to initiate CloudWatch Events and the Lambda function, and no SNS notifications.)
- This solution has gone through extensive testing with CloudFront web content distributions. It does not cover real-time messaging protocol (RTMP) streaming distributions.

Architecture

Target technology stack

- Lambda function
- SNS topic
- Amazon EventBridge rule

Target architecture

Automation and scale

- If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy the attached template across multiple accounts that you want to monitor.

Tools

AWS services

- AWS CloudFormation – CloudFormation is a service that helps you model and set up AWS resources by using infrastructure as code.
- Amazon EventBridge – EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, routing that data to targets such as Lambda functions.
- AWS Lambda – Lambda supports running code without provisioning or managing servers.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS – Amazon SNS coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

The attached code includes:

- A .zip file that contains the Lambda code (index.py)
- A CloudFormation template (.yml file) that you run to deploy the Lambda code

Epics

Upload the security control

Task	Description	Skills required
Create the S3 bucket for the Lambda code.	On the Amazon S3 console, create a S3 bucket with an	Cloud architect

Task	Description	Skills required
	unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket must be in the Region where you are planning to deploy the Lambda code.	
Upload the Lambda code to the S3 bucket.	Upload the Lambda code (cloudfront_ssl_log_lambda.zip file) that's provided in the *Attachments* section to the S3 bucket you created in the previous step.	Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Deploy the CloudFormation template.	On the AWS CloudFormation console, in the same AWS Region as the S3 bucket, deploy the CloudFormation template (cloudfront-ssl-logging.yml) that's provided in the *Attachments* section.	Cloud architect
Specify the S3 bucket name.	For the **S3 Bucket** parameter, specify the name of the S3 bucket that you created in the first epic.	Cloud architect

Task	Description	Skills required
Specify the Amazon S3 key name for the Lambda file.	For the **S3 Key** parameter , specify the Amazon S3 location of the Lambda code .zip file in your S3 bucket. Do not include leading slashes (for example, you can enter lambda.zip or controls/lambda.zip).	Cloud architect
Provide a notification email address.	For the **Notification email** parameter, provide an email address where you would like to receive the violation notifications.	Cloud architect
Define the logging level.	For the **Lambda Logging level** parameter, define the logging level for your Lambda function. Choose one of the following values: • **INFO** to get detailed informational messages on the application's progress. • **ERROR** to get informati on about error events that could still allow the application to continue running. • **WARNING** to get informati on about potentially harmful situations.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the CloudFormation template has been deployed successfully, a new SNS topic is created and a subscript ion message is sent to the email address you provided. You must confirm this email subscription to receive violation notifications.	Cloud architect

Related resources

- AWS CloudFormation information
- Creating a stack on the AWS CloudFormation console (CloudFormation documentation)
- CloudFront logging (CloudFront documentation)
- Amazon S3 information
- AWS Lambda information

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Check for single-host network entries in security group ingress rules for IPv4 and IPv6

SaiJeevan Devireddy, Ganesh Kumar, and John Reynolds, Amazon Web Services

Summary

This pattern provides a security control that notifies you when Amazon Web Services (AWS) resources do not meet your specifications. It provides an AWS Lambda function that looks for single-host network entries in both Internet Protocol version 4 (IPv4) and IPv6 security group source address fields. The Lambda function is initiated when Amazon CloudWatch Events detects the Amazon Elastic Compute Cloud (Amazon EC2) AuthorizeSecurityGroupIngress API call. The custom logic in the Lambda function evaluates the subnet mask of the CIDR block of the security group ingress rule. If the subnet mask is determined to be anything other than /32 (IPv4) or /128 (IPv6), the Lambda function sends a violation notification by using Amazon Simple Notification Service (Amazon SNS).

Prerequisites and limitations

Prerequisites

- An active AWS account
- An email address where you want to receive the violation notifications

Limitations

- This security monitoring solution is regional and must be deployed in each AWS Region that you want to monitor.

Architecture

Target technology stack

- Lambda function
- SNS topic
- Amazon EventBridge rule

Target architecture

Automation and scale

- If you are using AWS Organizations, you can use [AWS Cloudformation StackSets](#) to deploy this template across multiple accounts that you want to monitor.

Tools

AWS services

- [AWS CloudFormation](#) is a service that helps you model and set up AWS resources by using infrastructure as code.
- [Amazon EventBridge](#) delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, and routes that data to targets such as Lambda functions.
- [AWS Lambda](#) supports running code without provisioning or managing servers.
- [Amazon Simple Storage Service (Amazon S3)](#) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- [Amazon SNS](#) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

The attached code includes:

- A .zip file that contains the Lambda security control code (`index.py`)

- A CloudFormation template (`security-control.yml` file) that you run to deploy the Lambda code

Epics

Upload the security control

Task	Description	Skills required
Create the S3 bucket for the Lambda code.	On the [Amazon S3 console](https://console.aws.amazon.com/s3/), create a S3 bucket with an unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket must be in the AWS Region where you want to deploy the security group ingress check.	Cloud architect
Upload the Lambda code to the S3 bucket.	Upload the Lambda code (`security-control-lambda.zip` file) that's provided in the *Attachments* section to the S3 bucket that you created in the previous step.	Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Change the Python version.	Download the CloudFormation template (`security-control.yml`) that's	Cloud architect

Task	Description	Skills required
	provided in the *Attachments* section. Open the file and modify the Python version to reflect the latest version supported by Lambda (currently Python 3.9). For example, you can search for `python` in the code and change the value for `Runtime` from `python3.6` to `python3.9`. For the latest information about Python runtime version support, see the [AWS Lambda documentation]().	
Deploy the AWS CloudFormation template.	On the AWS CloudFormation console, in the same AWS Region as the S3 bucket, deploy the CloudFormation template (`security-control.yml`).	Cloud architect
Specify the S3 bucket name.	For the **S3 Bucket** parameter, specify the name of the S3 bucket that you created in the first epic.	Cloud architect

Task	Description	Skills required
Specify the Amazon S3 key name for the Lambda file.	For the **S3 Key** parameter , specify the Amazon S3 location of the Lambda code .zip file in your S3 bucket. Do not include leading slashes (for example, you can enter `lambda.zip` or `controls/lambda.zip`).	Cloud architect
Provide a notification email address.	For the **Notification email** parameter, provide an email address where you would like to receive the violation notifications.	Cloud architect
Define the logging level.	For the **Lambda Logging level** parameter, define the logging level for your Lambda function. Choose one of the following values: • **INFO** to get detailed informational messages on the application's progress. • **ERROR** to get information about error events that could still allow the application to continue running. • **WARNING** to get information about potentially harmful situations.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the CloudFormation template has been deployed successfully, a new SNS topic is created and a subscript ion message is sent to the email address you provided. You must confirm this email subscription to receive violation notifications.	Cloud architect

Related resources

- AWS CloudFormation information
- Creating a stack on the AWS CloudFormation Console (AWS CloudFormation documentation)
- Security groups for your VPC (Amazon VPC documentation)
- Amazon S3 information
- AWS Lambda information

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Choose an Amazon Cognito authentication flow for enterprise applications

Michael Daehnert and Fabian Jahnke, Amazon Web Services

Summary

[Amazon Cognito](https://aws.amazon.com/cognito/) provides authentication, authorization, and user management for web and mobile applications. It offers beneficial features for authentication of federated identities. To get it up and running, technical architects need to decide how they want to use those features.

Amazon Cognito supports multiple flows for authentication requests. These flows define how your users can verify their identity. The decision about which authentication flow to use depends on specific requirements of your application and can become complex. This pattern helps you decide which authentication flow is the best fit for your enterprise application. It assumes that you already have a basic knowledge of Amazon Cognito, OpenID Connect (OIDC), and federation, and it guides you through details about different federated authentication flows.

This solution is intended for technical decision makers. It helps you understand the different authentication flows and map them to your application requirements. Technical leads should gather the required insights to start the Amazon Cognito integrations. Because enterprise organizations mainly focus on SAML federation, this pattern includes descriptions for Amazon Cognito user pools with SAML federation.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Identity and Access Management (IAM) roles and permissions with full access to Amazon Cognito
- (Optional) Access to your identity provider (IdP), such as Microsoft Entra ID, Active Directory Federation Service (AD FS), or Okta
- A high level of expertise for your application
- Basic knowledge of Amazon Cognito, OpenID Connect (OIDC), and federation

Limitations

- This pattern focuses on Amazon Cognito user pools and identity providers. For information about Amazon Cognito identity pools, see the [Additional information] section.

Architecture

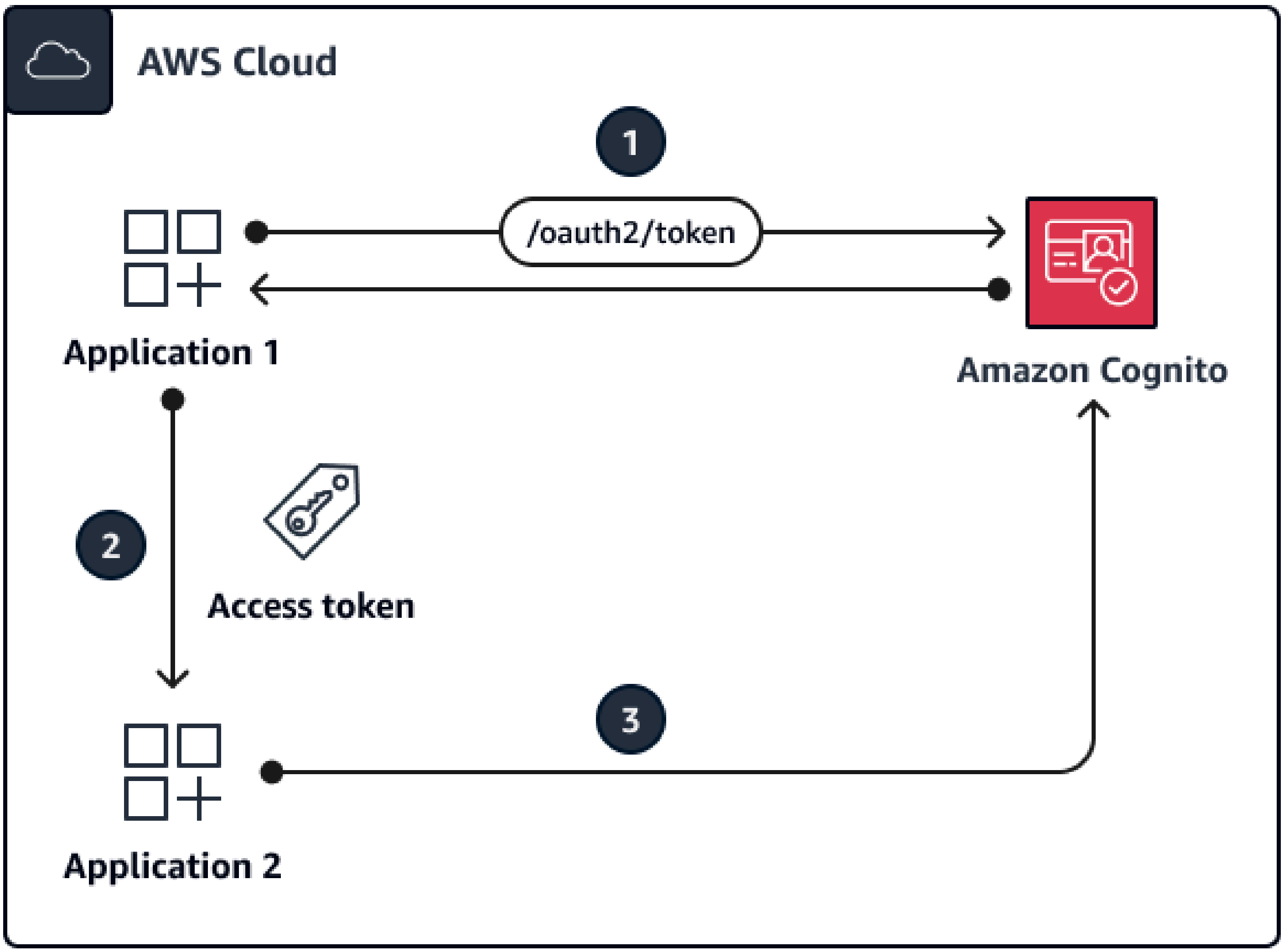
Use the following table to help you choose an authentication flow. More information about each flow is provided in this section.

Do you need machine-to-machine authentication?	Is your app a web-based application where the frontend is rendered on the server?	Is your app a single-page application (SPA) or mobile-based frontend application?	Does your application require refresh tokens for a "keep me signed in" feature?	Does the frontend offer a browser-based redirect mechanism?	Recommended Amazon Cognito flow
Yes	No	No	No	No	Client Credentials flow
No	Yes	No	Yes	Yes	Authorization Code flow
No	No	Yes	Yes	Yes	Authorization Code flow with Proof Key for Code Exchange (PKCE)
No	No	No	No	No	Resource Owner Password flow*

* Resource Owner Password flow should be used only if absolutely necessary. For more information, see the *Resource Owner Password flow* section in this pattern.

Client Credentials flow

The Client Credentials flow is the shortest of the Amazon Cognito flows. It should be used if systems or services communicate with each other without any user interaction. The requesting system uses the client ID and the client secret to retrieve an access token. Because both systems work without user interaction, no additional consent step is required.



The diagram illustrates the following:

1. Application 1 sends an authentication request with the client ID and client secret to the Amazon Cognito endpoint, and it retrieves an access token.
2. Application 1 uses this access token for every subsequent call to Application 2.

3. Application 2 validates the access token with Amazon Cognito.

This flow should be used:

- For communications between applications with no user interaction

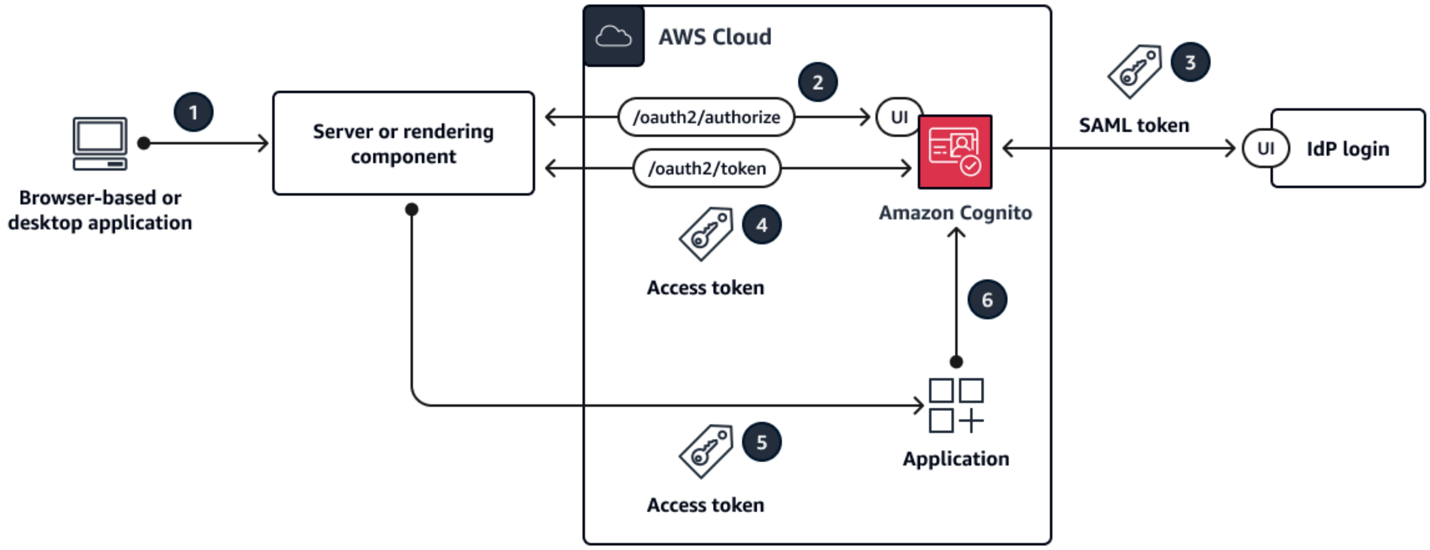
This flow should not be used:

- For any communication in which user interactions are possible

Authorization Code flow

The Authorization Code flow is for classic web-based authentication. In this flow, the backend handles all of the token exchange and storage. The browser-based client does not see the actual tokens. This solution is used for applications written in frameworks such as .NET Core, Jakarta Faces, or Jakarta Server Pages (JSP).

The Authorization Code flow is a redirection-based flow. The client must be able to interact with the web browser or a similar client. The client is redirected to an authentication server and authenticates against this server. If the client authenticates successfully, it is redirected back to the server.



The diagram illustrates the following:

1. The client sends a request to the web server.

2. The web server redirects the client to Amazon Cognito by using an HTTP 302 status code. The client automatically follows this redirect to the configured IdP login.
3. The IdP checks for an existing browser session on the IdP side. If none exists, the user receives a prompt to authenticate by providing their username and password. The IdP responds with a SAML token to Amazon Cognito.
4. Amazon Cognito returns success with a JSON web token (JWT), specifically a code token. The web server calls /oauth2/token to exchange the code token for an access token. The web server sends the client ID and client secret to Amazon Cognito for validation.
5. The access token is used for every subsequent call to other applications.
6. Other applications validate the access token with Amazon Cognito.

This flow should be used:

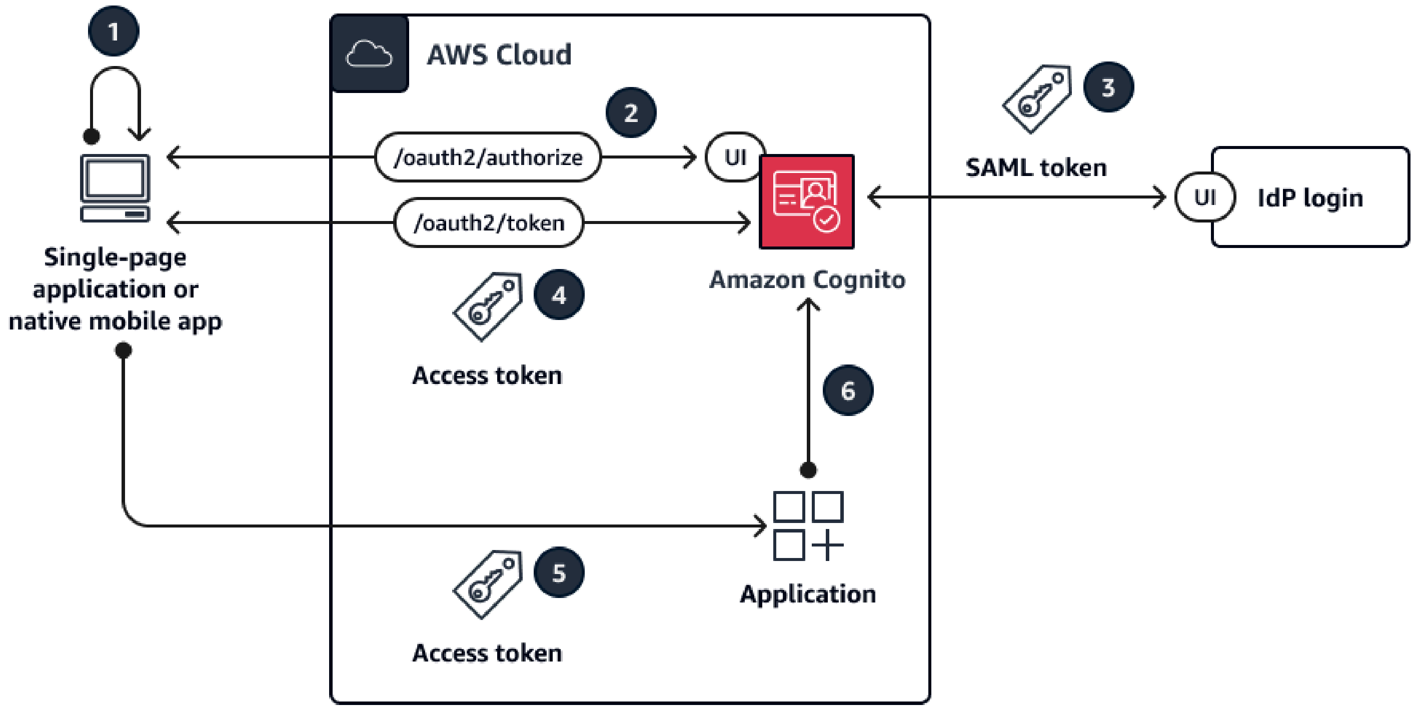
- If the user is able to interact with the web browser or client. The application code is run and rendered on the server to make sure that no secrets are exposed to the browser.

This flow should not be used:

- For single-page applications (SPAs) or mobile apps because they're rendered on the client and shouldn't use client secrets.

Authorization Code flow with PKCE

Authorization Code flow with Proof Key for Code Exchange (PKCE) should be used for single-page applications and mobile applications. It is the successor of the Implicit flow and is more secure because it uses PKCE. PKCE is an extension to the OAuth 2.0 authorization code grant for public clients. PKCE guards against the redemption of intercepted authorization codes.



The diagram illustrates the following:

1. The application creates a code verifier and code challenge. These are well defined, unique values that are sent to Amazon Cognito for future reference.
2. The application calls the /oauth2/authorization endpoint of Amazon Cognito. It automatically redirects the user to the configured IdP login.
3. The IdP checks for an existing session. If none exists, the user receives a prompt to authenticate by providing their username and password. The IdP responds with a SAML token to Amazon Cognito.
4. After Amazon Cognito returns success with a code token, the web server calls /oauth2/token to exchange the code token for an access token.
5. The access token is used for every subsequent call to other applications.
6. The other applications validate the access token with Amazon Cognito.

This flow should be used:

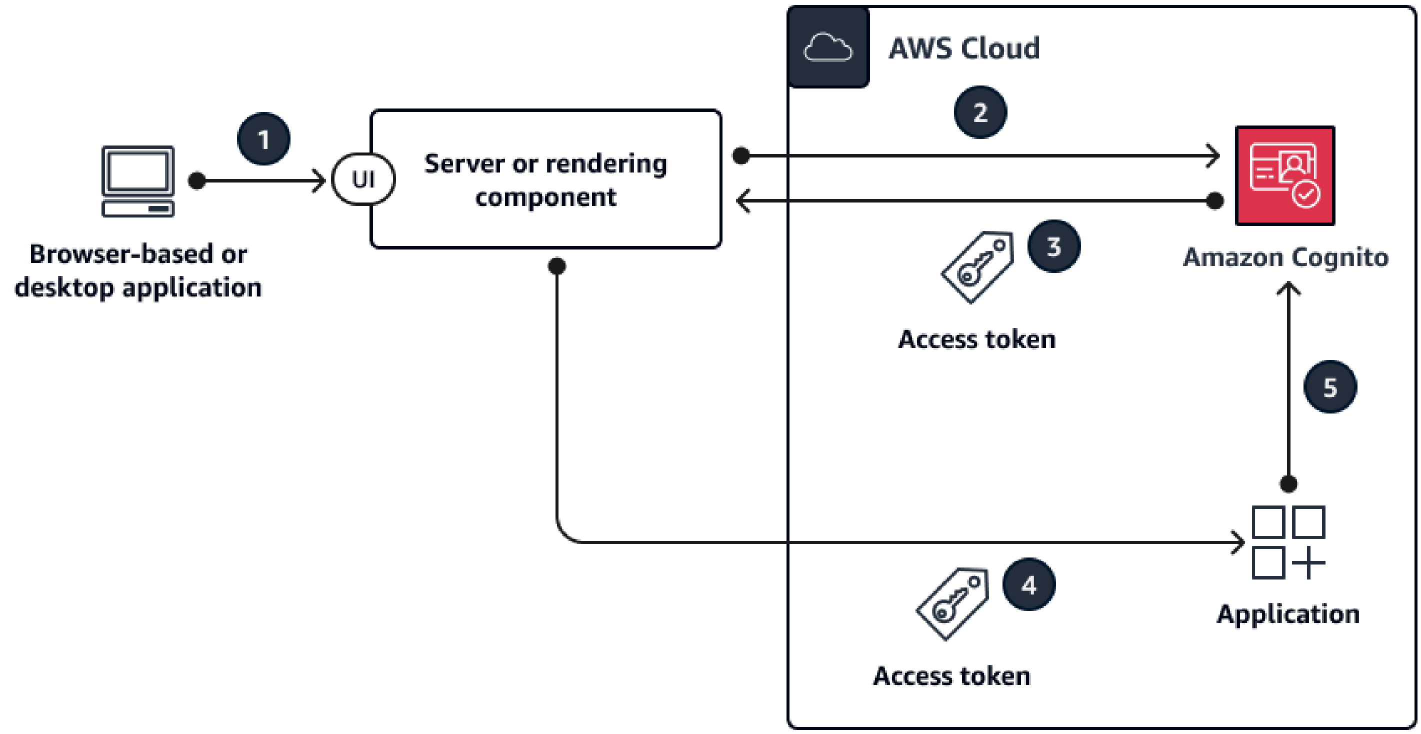
- For SPAs or mobile applications

This flow should not be used:

- If the application backend handles authentication

Resource Owner Password flow

The Resource Owner Password flow is intended for applications with no redirect capabilities. It is built by creating a login form in your own application. The login is checked on Amazon Cognito through a CLI or SDK call instead of relying on redirect flows. Federation is not possible in this authentication flow because federation requires browser-based redirects.



The diagram illustrates the following:

1. The user enters their credentials on a login form provided by the application.
2. The AWS Command Line Interface (AWS CLI) makes an admin-initiated-auth call to Amazon Cognito.

> **Note**
> Alternatively, you can use AWS SDKs instead of the AWS CLI.

3. Amazon Cognito returns an access token.
4. The access token is used for every subsequent call to other applications.

5. The other applications validate the access token with Amazon Cognito.

This flow should be used:

- When migrating existing clients that use direct authentication logic (such as basic access authentication or digest access authentication) to OAuth by converting the stored credentials to an access token

This flow should not be used:

- If you want to use federated identities
- If your application supports redirects

Tools

AWS services

- Amazon Cognito provides authentication, authorization, and user management for web and mobile apps.

Other tools

- JSON web token (JWT) debugger is a web-based JWT validation tool.

Epics

Assess your application

Task	Description	Skills required
Define authentication requirements.	Assess your application according to your specific authentication requirements.	App developer, App architect
Align requirements with authentication flows.	In the Architecture section, use the decision table and explanations of each flow to	App developer, General AWS, App architect

Task	Description	Skills required
	choose your Amazon Cognito authentication flow.	

Set up the Amazon Cognito user pool

Task	Description	Skills required
Create a user pool.	1. Sign in to the AWS Management Console, and then open the [Amazon Cognito console](https://console.aws.amazon.com/cognito/). 2. Create a new Cognito user pool. For instructions, see [Amazon Cognito user pools](https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html). 3. Update the user pool settings and attributes as needed. For example, set a [password policy](https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-settings-policies.html) for the user pool. Do not create app clients yet.	General AWS
(Optional) Configure an identity provider.	1. Create a SAML identity provider in the Amazon Cognito user pool. For instructions, see [Adding and managing SAML identity providers in a user pool](https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-managing-saml-idp.html). 2. Configure your third-party SAML identity provider to work with federation for Amazon Cognito user pools. For more informati	General AWS, Federation administrator

Task	Description	Skills required
	on, see [Configuring your third-party SAML identity provider](). If you're using AD FS, see [Building AD FS Federation for your Web App using Amazon Cognito User Pools]() (AWS blog post).	
Create an app client.	1. Create an app client for the user pool. For instructions, see [Creating an app client](). Note the following: • Change settings as needed, such as token expirations. • If your authentication flow doesn't require a client secret, then clear the **Generate client secret** checkbox. 2. Choose **App client settings** to change its integrati on to a user pool login (username and password) or federated login through a SAML-based IdP. 3. Enable your IdP by defining URLs and defining OAuth flows or scopes as needed.	General AWS

Integrate the application with Amazon Cognito

Task	Description	Skills required
Exchange Amazon Cognito integration details.	Depending on your authentication flow, share Amazon Cognito information with the application, such as the user pool ID and app client ID.	App developer, General AWS
Implement Amazon Cognito authentication.	This depends on your chosen authentication flow, your programming language, and the frameworks you're using. For some links to get started, see the Related resources section.	App developer

Related resources

AWS documentation

- User pool authentication flow
- Verifying a JSON web token
- Access AWS services from an ASP.NET Core app using Amazon Cognito identity pools
- Frameworks and SDKs:
  - Amazon Amplify authentication
  - Amazon Cognito Identity Provider examples (AWS SDK for Java 2.x documentation)
  - Authenticating users with Amazon Cognito (AWS SDK for .NET documentation)

AWS blog posts

- Authorization@Edge using cookies: Protect your Amazon CloudFront content from being downloaded by unauthenticated users
- Building AD FS Federation for your Web App using Amazon Cognito User Pools

Implementation partners

- AWS Partners for authentication solutions

Additional information

FAQ

Why is the Implicit flow deprecated?

Since the release of the OAuth 2.1 framework, the Implicit flow is marked as deprecated for security reasons. As an alternative, please use the Authorization Code flow with PKCE described in the Architecture section.

What if Amazon Cognito doesn't offer some functionality I require?

AWS Partners offer different integrations for authentication and authorization solutions. For more information, see AWS Partners for authentication solutions.

What about Amazon Cognito identity pool flows?

Amazon Cognito user pools and federated identities are for authentication. Amazon Cognito identity pools are used for authorization of AWS resources access by requesting temporary AWS credentials. The ID token and access token exchange for identity pools isn't discussed in this pattern. For more information, see What's the difference between Amazon Cognito user pools and identity pools and Common Amazon Cognito scenarios.

Next steps

This pattern provides an overview of Amazon Cognito authentication flows. As a next step, the detailed implementation for the application's programming language needs to be chosen. Multiple languages offer SDKs and frameworks, which you can use with Amazon Cognito. For helpful references, see the Related resources section.

Create AWS Config custom rules by using AWS CloudFormation Guard policies

Andrew Lok, Nicole Brown, Kailash Havildar, and Tanya Howell, Amazon Web Services

Summary

AWS Config rules help you evaluate your AWS resources and their target configuration state. There are two types of AWS Config rules: managed and custom. Your can create custom rules with AWS Lambda functions or with AWS CloudFormation Guard (GitHub), a policy-as-code language.

Rules created with Guard provide more granular control than managed rules, and they are typically easier to configure than fully custom Lambda rules. This approach provides engineers and architects the ability to build rules without needing to know Python, NodeJS, or Java, which are required to deploy custom rules through Lambda.

This pattern provides workable templates, code samples, and deployment approaches to help you adopt custom rules with Guard. By using this pattern, an administrator can use AWS Config to build custom compliance rules that have configuration item attributes. For example, developers can use Guard policies against AWS Config configuration items to continuously monitor the state of deployed AWS and non-AWS resources, detect rule violations, and automatically initiate remediation.

Objectives

After reading this pattern, you should be able to:

- Understand how Guard policy code interacts with the AWS Config service.
- Deploy *Scenario 1*, which is an AWS Config custom rule that uses Guard syntax to validate compliance for encrypted volumes. This rule verifies that the drive is in use and verifies that the drive type is gp3.
- Deploy *Scenario 2*, which is an AWS Config custom rule that uses Guard syntax to validate Amazon GuardDuty compliance. This rule verifies that GuardDuty recorders have Amazon Simple Storage Service (Amazon S3) Protection and Amazon Elastic Kubernetes Service (Amazon EKS) Protection enabled.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Config, set up in your AWS account

Limitations

- Guard custom rules are only able to query key-value pairs in a target configuration item JSON record

Architecture

You apply the Guard syntax to an AWS Config rule as a custom policy. AWS Config captures the hierarchical JSON of each of the resources specified. The JSON of the AWS Config configuration item contains key-value pairs. These attributes are used in the Guard syntax as variables that are assigned to their corresponding value.

The following is an explanation of the Guard syntax. The variables from the configuration item JSON are used and prepended with a % character.

```
# declare variable
let <variable name> = <'value'>

# create rule and assign condition and policy
    rule <rule name> when
        <CI json key> == <"CI json value"> {
            <top level CI json key>.<next level CI json key> == %<variable name>
        }
```

Scenario 1: Amazon EBS volumes

Scenario 1 deploys an AWS Config custom rule that uses Guard syntax to validate compliance for encrypted volumes. This rule verifies that the drive is in use and verifies that the drive type is gp3.

The following is an example of an AWS Config configuration item for scenario 1. There are three key-value pairs in this configuration item that used as variables in the Guard policy:

`volumestatus`, `volumeencryptionstatus`, and `volumetype`. Also, the `resourceType` key is used as a filter in the Guard policy.


```
{
  "version": "1.3",
  "accountId": "111111111111",
  "configurationItemCaptureTime": "2023-01-15T19:04:45.402Z",
  "configurationItemStatus": "ResourceDiscovered",
  "configurationStateId": "4444444444444",
  "configurationItemMD5Hash": "",
  "arn": "arn:aws:ec2:us-west-2:111111111111:volume/vol-222222222222",
  "resourceType": "AWS::EC2::Volume",
  "resourceId": "vol-222222222222",
  "awsRegion": "us-west-2",
  "availabilityZone": "us-west-2b",
  "resourceCreationTime": "2023-01-15T19:03:22.247Z",
  "tags": {},
  "relatedEvents": [],
  "relationships": [
    {
      "resourceType": "AWS::EC2::Instance",
      "resourceId": "i-33333333333333333",
      "relationshipName": "Is attached to Instance"
    }
  ],
  "configuration": {
    "attachments": [
      {
        "attachTime": "2023-01-15T19:03:22.000Z",
        "device": "/dev/xvda",
        "instanceId": "i-33333333333333333",
        "state": "attached",
        "volumeId": "vol-222222222222",
        "deleteOnTermination": true,
        "associatedResource": null,
        "instanceOwningService": null
      }
    ],
    "availabilityZone": "us-west-2b",
    "createTime": "2023-01-15T19:03:22.247Z",
    "encrypted": false,
    "kmsKeyId": null,
    "outpostArn": null,
```

```
    "size": 8,
    "snapshotId": "snap-55555555555555555",
    "state": "in-use",
    "volumeId": "vol-222222222222",
    "iops": 100,
    "tags": [],
    "volumeType": "gp2",
    "fastRestored": null,
    "multiAttachEnabled": false,
    "throughput": null,
    "sseType": null
  },
  "supplementaryConfiguration": {}
}
```


The following is an example of using Guard syntax to define the variables and rules in scenario 1. In the following example:

- The first three lines define the variables by using the `let` command. They are assigned a name and value that is derived from the attributes of the configuration item.
- The `compliancecheck` rule block adds a when conditional dependency that looks for a `resourceType` key-value pair that matches `AWS::EC2::Volume`. If a match is found, the rule proceeds through the rest of the JSON attributes and looks for matches on the following three conditions: `state`, `encrypted`, and `volumeType`.

```
let volumestatus = 'available'
let volumetype = 'gp3'
let volumeencryptionstatus = true

    rule compliancecheck when
        resourceType == "AWS::EC2::Volume" {
            configuration.state == %volumestatus
            configuration.encrypted == %volumeencryptionstatus
            configuration.volumeType == %volumetype
        }
```

For the complete Guard custom policy that implements this custom rule, see [awsconfig-guard-cft.yaml] or [awsconfig-guard-tf-ec2vol.json] in the GitHub code repository. For HashiCorp Terraform code that deploys this custom policy in Guard, see [awsconfig-guard-tf-example.json] in the code repository.

Scenario 2: GuardDuty compliance

Scenario 2 deploys an AWS Config custom rule that uses Guard syntax to validate Amazon GuardDuty compliance. This rule verifies that GuardDuty recorders have Amazon S3 Protection and Amazon EKS Protection enabled. It also verifies that GuardDuty findings are published every 15 minutes. This scenario could be deployed across all AWS accounts and AWS Regions in an organization (in AWS Organizations).

The following is an example of an AWS Config configuration item for scenario 2. There are three key-value pairs in this configuration item that used as variables in the Guard policy: `FindingPublishingFrequency`, `S3Logs`, and `Kubernetes`. Also, the `resourceType` key is used as a filter in the policy.

```
{
  "version": "1.3",
  "accountId": "111111111111",
  "configurationItemCaptureTime": "2023-11-27T13:34:28.888Z",
  "configurationItemStatus": "OK",
  "configurationStateId": "7777777777777",
  "configurationItemMD5Hash": "",
  "arn": "arn:aws:guardduty:us-
west-2:111111111111:detector/66666666666666666666666666666666",
  "resourceType": "AWS::GuardDuty::Detector",
  "resourceId": "66666666666666666666666666666666",
  "resourceName": "66666666666666666666666666666666",
  "awsRegion": "us-west-2",
  "availabilityZone": "Regional",
  "resourceCreationTime": "2020-02-17T02:48:04.511Z",
  "tags": {},
  "relatedEvents": [],
  "relationships": [],
  "configuration": {
    "Enable": true,
    "FindingPublishingFrequency": "FIFTEEN_MINUTES",
    "DataSources": {
      "S3Logs": {
        "Enable": true
      },
      "Kubernetes": {
        "AuditLogs": {
          "Enable": true
        }
```

```
      }
    },

    "Id": "66666666666666666666666666666666",
    "Tags": []
  },
  "supplementaryConfiguration": {
    "CreatedAt": "2020-02-17T02:48:04.511Z"
  }
}
```

The following is an example of using Guard syntax to define the variables and rules in scenario 2. In the following example:

- The first three lines define the variables by using the `let` command. They are assigned a name and value that is derived from the attributes of the configuration item.
- The `compliancecheck` rule block adds a when conditional dependency that looks for a `resourceType` key-value pair that matches `AWS::GuardDuty::Detector`. If a match is found, the rule proceeds through the rest of the JSON attributes and looks for matches on the following three conditions: `S3Logs.Enable`, `Kubernetes.AuditLogs.Enable`, and `FindingPublishingFrequency`.

```
let s3protection = true
let kubernetesprotection = true
let publishfrequency = 'FIFTEEN_MINUTES'

    rule compliancecheck when
        resourceType == "AWS::GuardDuty::Detector" {
            configuration.DataSources.S3Logs.Enable == %s3protection
            configuration.DataSources.Kubernetes.AuditLogs.Enable ==
 %kubernetesprotection
            configuration.FindingPublishingFrequency == %publishfrequency
        }
```

For the complete Guard custom policy that implements this custom rule, see [awsconfig-guard-cft-gd.yaml] in the GitHub code repository. For HashiCorp Terraform code that deploys this custom policy in Guard, see [awsconfig-guard-tf-gd.json] in the code repository.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time.

Other tools

- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources.

Code repository

The code for this pattern is available in the GitHub AWS Config with AWS CloudFormation Guard repository. This code repository contains samples for both of the scenarios described in this pattern.

Epics

Creating AWS Config custom rules

Task	Description	Skills required
(Optional) Select key-value pairs for the rule.	Complete these steps if you are defining a custom Guard policy. If you are using one of the sample policies for scenario 1 or 2, skip these steps. 1. Sign in to the AWS Management Console and open the AWS Config console.	AWS administrator, Security engineer

Task	Description	Skills required
	2. In the left navigation, choose **Resources**. 3. In the resource inventory , choose the type of resource that you want to create an AWS Config custom rule for. 4. Choose **View details**. 5. Choose **View Configura tion Item (JSON)**. This section expands to show the configuration item in JSON format. 6. Identify the key-value pairs that you would like to build an AWS Config custom rule for.	
Create the custom rule.	Using the key-value pairs that you identified previously or using one of the provided sample Guard policies, follow the instructions in [Creating AWS Config Custom Policy Rules](https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config_develop-rules_cfn-guard.html) to create a custom rule.	AWS administrator, Security engineer

Task	Description	Skills required
Validate the custom rule.	Do one of the following to validate the custom Guard rule: • Enter the following command in the AWS Command Line Interface (AWS CLI). `cfn-guard validate -r guard-s3.guard -d s3bucket-prod-pass.json` • Follow the instructions in *Detective mode* in Evaluating Your Resources with AWS Config Rules to deploy the rule in AWS Config. Confirm that the Guard syntax matches correctly to corresponding resources in the target account or file.	AWS administrator, Security engineer

Troubleshooting

Issue	Solution
Test the Guard policy outside of AWS Config	Unit testing can be done on your local device or in an integrated development environment (IDE), such as an AWS Cloud9 IDE. To perform unit testing, do the following: 1. Install the AWS CloudFormation Guard CLI and its dependencies.

Issue	Solution

	2. Save a JSON-formatted CI sample to your workstation as a .json file.
	3. Save the GuardDuty policy to your workstation as a .guard file.
	4. In the Guard CLI, enter the following command to validate the sample JSON file by using the Guard policy.
	`cfn-guard validate \`
	`-r guard-s3.guard \`
	`-d s3bucket-prod-pass.json`

Debug an AWS Config custom rule	In your Guard policy, change the `EnableDebugLogDelivery` value to `true`. The default value is `false`. The log messages are stored in Amazon CloudWatch.

Related resources

AWS documentation

- Creating AWS Config Custom Policy Rules (AWS Config documentation)
- Writing AWS CloudFormation Guard rules (Guard documentation)

AWS blog posts and workshops

- Introducing AWS CloudFormation Guard 2.0 (AWS blog post)

Other resources

- AWS CloudFormation Guard (GitHub)
- AWS CloudFormation Guard CLI documentation (GitHub)

Create a consolidated report of Prowler security findings from multiple AWS accounts

Mike Virgilio, Jay Durga, and Andrea Di Fabio, Amazon Web Services

Summary

[Prowler](https://github.com/prowler-cloud/prowler) (GitHub) is an open-source command line tool that can help you assess, audit, and monitor your Amazon Web Services (AWS) accounts for adherence to security best practices. In this pattern, you deploy Prowler in a centralized AWS account in your organization, managed by AWS Organizations, and then use Prowler to perform a security assessment of all of the accounts in the organization.

While there are many methods to deploy and utilize Prowler for an assessment, this solution has been designed for rapid deployment, full analysis of all accounts in the organization or defined target accounts, and accessible reporting of the security findings. In this solution, when Prowler completes the security assessment of all accounts in the organization, it consolidates the results. It also filters out any expected error messages, such as errors related to restrictions that prevent Prowler from scanning Amazon Simple Storage Service (Amazon S3) buckets in accounts provisioned through AWS Control Tower. The filtered, consolidated results are reported in a Microsoft Excel template that is included with this pattern. You can use this report to identify potential improvements for the security controls in your organization.

This solution was designed with the following in mind:

- The AWS CloudFormation templates reduce the effort required to deploy the AWS resources in this pattern.
- You can adjust the parameters in the CloudFormation templates and **prowler_scan.sh** script at the time of deployment to customize the templates for your environment.
- Prowler assessment and reporting speeds are optimized through parallel processing of AWS accounts, aggregated results, consolidated reporting with recommended remediations, and automatically generated visualizations.
- The user doesn't need to monitor the scan progress. When the assessment is complete, the user is notified through an Amazon Simple Notification Service (Amazon SNS) topic so that they can retrieve the report.
- The report template helps you read and assess only the relevant results for your entire organization.

Prerequisites and limitations

Prerequisites

- An AWS account for hosting security services and tools, managed as a member account of an organization in AWS Organizations. In this pattern, this account is referred to as the *security account*.
- In the security account, you must have a private subnet with outbound internet access. For instructions, see VPC with servers in private subnets and NAT in the Amazon Virtual Private Cloud (Amazon VPC) documentation. You can establish internet access by using an NAT gateway that is provisioned in a public subnet.
- Access to the AWS Organizations management account or an account that has delegated administrator permissions for CloudFormation. For instructions, see Register a delegated administrator in the CloudFormation documentation.
- Enable trusted access between AWS Organizations and CloudFormation. For instructions, see Enable trusted access with AWS Organizations in the CloudFormation documentation.

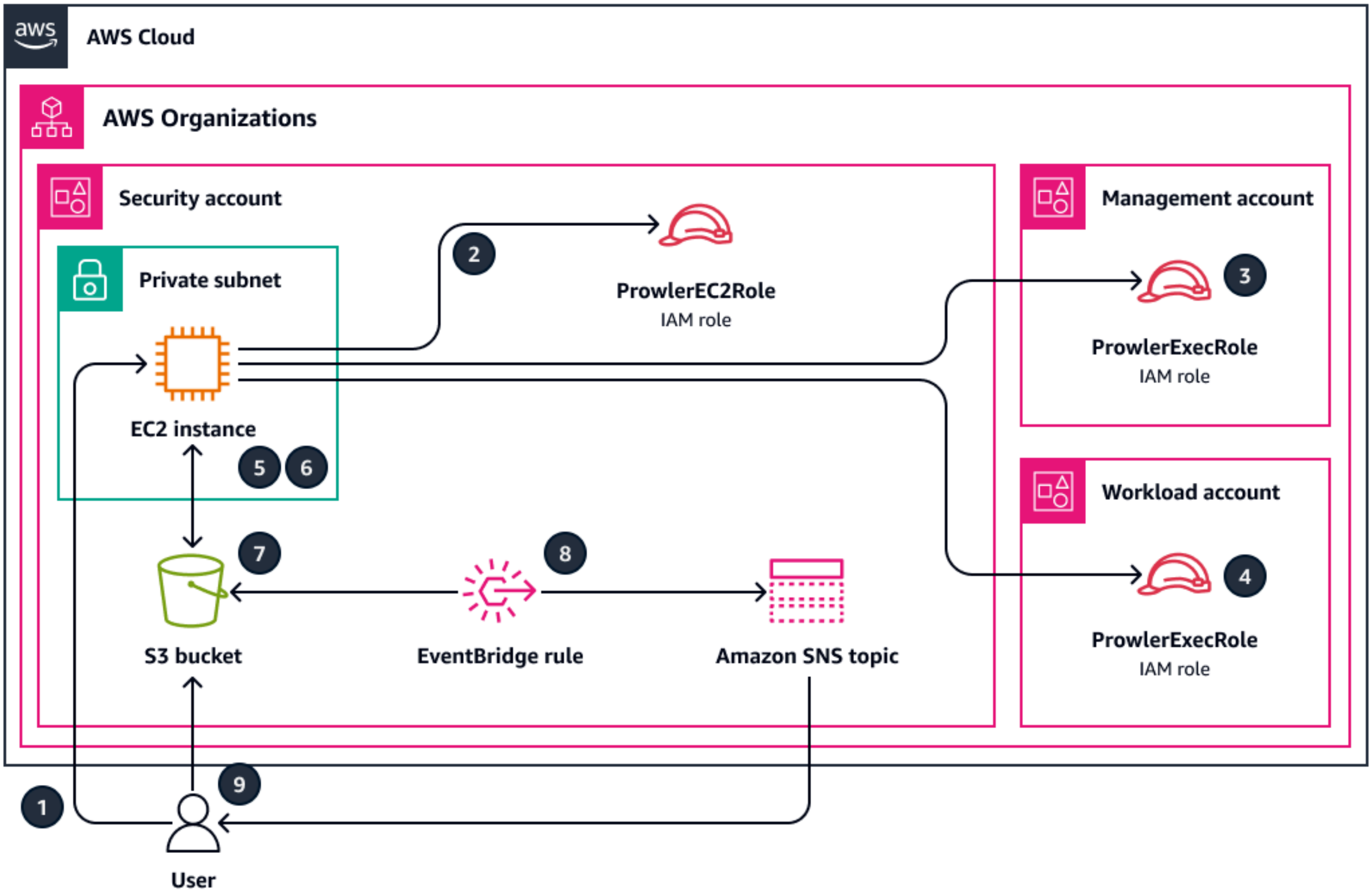
Limitations

- The target AWS accounts must be managed as an organization in AWS Organizations. If you are not using AWS Organizations, you can update the **IAM-ProwlerExecRole.yaml** CloudFormation template and the **prowler_scan.sh** script for your environment. Instead, you provide a list of AWS account IDs and Regions where you want to run the script.
- The CloudFormation template is designed to deploy the Amazon Elastic Compute Cloud (Amazon EC2) instance in a private subnet that has outbound internet access. The AWS Systems Manager Agent (SSM Agent) requires outbound access to reach the AWS Systems Manager service endpoint, and you need outbound access to clone the code repository and install dependencies. If you want to use a public subnet, you must modify the **prowler-resources.yaml** template to associate an Elastic IP address with the EC2 instance.

Product versions

- Prowler version 4.0 or later

Architecture



The diagram shows the following process:

1. Using Session Manager, a capability of AWS Systems Manager, the user authenticates to the EC2 instance and runs the **prowler_scan.sh** script. This shell script performs steps 2–8.
2. The EC2 instance assumes the `ProwlerEC2Role` IAM role, which grants permissions to access the S3 bucket and to assume the `ProwlerExecRole` IAM roles in the other accounts in the organization.
3. The EC2 instance assumes the `ProwlerExecRole` IAM role in the organization's management account and generates a list of the accounts in the organization.
4. The EC2 instance assumes the `ProwlerExecRole` IAM role in the organization's member accounts (called *workload accounts* in the architecture diagram) and performs a security assessment in each account. The findings are stored as CSV and HTML files on the EC2 instance.

> **Note**
>
> HTML files are an output of the Prowler assessment. Due to the nature of HTML, they aren't concatenated, processed, or used directly in this pattern. However, these might be useful for individual account report review.

5. The EC2 instance processes all of the CSV files to remove known, expected errors and consolidates the remaining findings into a single CSV file.
6. The EC2 instance packages the individual account results and aggregated results into a zip file.
7. The EC2 instance uploads the zip file to the S3 bucket.
8. An EventBridge rule detects the file upload and uses an Amazon SNS topic to send an email to the user notifying them that the assessment is complete.
9. The user downloads the zip file from the S3 bucket. The user imports the results into the Excel template and reviews the results.

Tools

AWS services

- Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

- [AWS Systems Manager](#) helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale. This pattern uses Session Manger, a capability of Systems Manager.

Other tools

- [Prowler](#) is an open-source command-line tool that helps you assess, audit, and monitor your accounts for adherence to AWS security best practices and other security frameworks and standards.

Code repository

The code for this pattern is available in the GitHub [Multi-Account Security Assessment via Prowler](#) repository. The code repository contains the following files:

- **prowler_scan.sh** – This bash script is used to start a Prowler security assessment of multiple AWS accounts, in parallel. As defined in the **Prowler-resources.yaml** CloudFormationtemplate, this script is automatically deployed to the `usr/local/prowler` folder on the EC2 instance.
- **Prowler-Resources.yaml** – You use this CloudFormation template to create a stack in the security account in the organization. This template deploys all of the required resources for this account in order to support the solution. This stack must be deployed before the **IAM-ProwlerExecRole.yaml** template. We do not recommend that you deploy these resources in an account that hosts critical production workloads.

> **Note**
>
> If this stack is deleted and redeployed, you must rebuild the `ProwlerExecRole` stack set in order to rebuild the cross-account dependencies between the IAM roles.

- **IAM-ProwlerExecRole.yaml** – You use this CloudFormation template to create a stack set that deploys the `ProwlerExecRole` IAM role in all accounts in the organization, including the management account.
- **prowler-report-template.xlsm** – You use this Excel template to process the Prowler findings. The pivot tables in the report provide search capabilities, charts, and consolidated findings.

Epics

Prepare for deployment

Task	Description	Skills required
Clone the code repository.	1. In a command-line interface, change your working directory to the location where you want to store the sample files. 2. Enter the following command: `git clone https://github.com/aws-samples/multi-account-security-assessment-via-prowler.git`	AWS DevOps
Review the templates.	1. In the cloned repository, open the **Prowler-Resources.yaml** and **IAM-ProwlerExecRole.yaml** files. 2. Review the resources created by these templates and adjust the templates as needed for your environment. For more information, see [Working with templates](#) in the CloudFormation documentation. 3. Save and close the **Prowler-Resources.yaml**	AWS DevOps

Task	Description	Skills required
	and **IAM-ProwlerExecRole.yaml** files.	

Create the CloudFormation stacks

Task	Description	Skills required
Provision resources in the security account.	Using the **prowler-resources.yaml** template, you create a CloudFormation stack that deploys all of the required resources in the security account. For instructions, see [Creating a stack](#) in the CloudFormation documentation. Note the following when deploying this template: 1. On the **Specify template** page, choose **Template is ready**, and then upload the **prowler-resources.yaml** file. 2. On the **Specify stack details** page, in the **Stack name** box, enter `Prowler-Resources` . 3. In the **Parameters** section, enter the following: • `VPCId` – Select a VPC in the account. • `SubnetId` –– Select a private subnet that has internet access.	AWS DevOps

Task	Description	Skills required

	Note: If you select a public subnet, the EC2 instance won't be assigned a public IP address because the CloudFormation template, by default, doesn't provision and attach an Elastic IP address.
	• `InstanceType` – Select an instance size based on the number of parallel assessments:
	  • For 10, choose `r6i.large` .
	  • For 12, choose `r6i.xlarge` .
	  • For 14–18, choose `r6i.2xlarge` .
	• `InstanceImageId` – Leave the default for Amazon Linux.
	• `KeyPairName` – If you're using SSH for access, specify the name of an existing key pair.
	• `PermittedSSHInbound` – If you're using SSH for access, specify a permitted CIDR block. If you're not using SSH,	

Task	Description	Skills required
	keep the default value of `127.0.0.1` . • `BucketName` – The default value is `prowler-output-<accountID>-<region>` . You can modify this as needed. If you specify a custom value, the account ID and Region are automatically appended to the specified value. • `EmailAddress` – Specify an email address for an Amazon SNS notification when Prowler completes the assessment and uploads the .zip file to the S3 bucket. *Note:* The SNS subscription configuration must be confirmed prior to Prowler completing the assessment or a notification will not be sent. • `IAMProwlerEC2Role` – Keep the default unless your naming conventions require a different name for this IAM role.	

Task	Description	Skills required
	• `IAMProwlerExecRole` – Keep the default unless another name will be used when deploying the **IAM-ProwlerExecRole.yaml** file. • `Parallelism` – Specify the number of parallel assessments to perform. Make sure that the value in the `InstanceType` parameter supports this number of parallel assessments. • `FindingOutput` – If you want to exclude pass results, select `FailOnly`. This significantly reduces the output size and focuses on the checks that might need to be resolved. If you want to include pass results, select `FailAndPass`. 4. On the **Review** page, select **The following resource(s) require capabilities: [AWS::IAM::Role]**, and then choose **Create Stack**. 5. After the stack has been successfully created, in the CloudFormation console, on the **Outputs** tab, copy	

Task	Description	Skills required
	the `ProwlerEC2Role` Amazon Resource Name (ARN). You use this ARN later when deploying the **IAM-ProwlerExecRole.yaml** file.	

Task	Description	Skills required
Provision the IAM role in the member accounts.	In the AWS Organizations management account or an account with delegated administrator permissions for CloudFormation, use the **IAM-ProwlerExecRole.yaml** template to create a CloudFormation stack set. The stack set deploys the `ProwlerExecRole` IAM role in all member accounts in the organization. For instructions, see [Create a stack set with service-managed permissions](#) in the CloudFormation documentation. Note the following when deploying this template: 1. Under **Prepare template**, choose **Template is ready**, and then upload the **IAM-ProwlerExecRole.yaml** file. 2. On the **Specify StackSet details** page, name the stack set `IAM-ProwlerExecRole`. 3. In the **Parameters** section, enter the following: • `AuthorizedARN` – Enter the `ProwlerEC2Role` ARN, which you copied when you	AWS DevOps

Task	Description	Skills required
	created the `Prowler-Resources` stack. • `ProwlerExecRoleName` – Keep the default value of `ProwlerExecRole` unless another name was used when deploying the **Prowler-Resources.yaml** file. 4. Under **Permissions**, choose **Service-managed permissions**. 5. On the **Set deployment options** page, under **Deployment targets**, choose **Deploy to organization** and accept all defaults. *Note:* If you want the stacks deployed to all member accounts simultaneously, set **Maximum concurrent accounts** and **Failure tolerance** to a high value, such as `100`. 6. Under **Deployment regions**, choose the AWS Region where the EC2 instance for Prowler is deployed. Because IAM resources are global and not Regional, this deploys	

Task	Description	Skills required
	the IAM role in all active Regions. 7. On the **Review** page, select **I acknowledge that AWS CloudFormation might create IAM resources with custom names**, and then choose **Create StackSet**. 8. Monitor the **Stack instances** tab (for individua l account status) and the **Operations** tab (for overall status) to determine when the deployment is complete.	

Task	Description	Skills required
Provision the IAM role in the management account.	Using the **IAM-ProwlerExecRole.yaml** template, you create a CloudFormation stack that deploys the `ProwlerExecRole` IAM role in the management account of the organization. The stack set you created previously doesn't deploy the IAM role in the management account. For instructions, see Creating a stack in the CloudFormation documentation. Note the following when deploying this template: 1. On the **Specify template** page, choose **Template is ready**, and then upload the **IAM-ProwlerExecRole.yaml** file. 2. On the **Specify stack details** page, in the **Stack name** box, enter `IAM-ProwlerExecRole` . 3. In the **Parameters** section, enter the following: • `AuthorizedARN` – Enter the `ProwlerEC2Role` ARN, which you copied when you created the `Prowler-Resources` stack.	AWS DevOps

Task	Description	Skills required
	• `ProwlerExecRoleName` – Keep the default value of `ProwlerExecRole` unless another name was used when deploying the **Prowler-Resources.yaml** file. 4. On the **Review** page, select **The following resource(s) require capabilities: [AWS::IAM::Role]**, and then choose **Create Stack**.	

Perform the Prowler security assessment

Task	Description	Skills required
Run the scan.	1. Sign in to the security account in the organization. 2. Using Session Manager, connect to the EC2 instance for Prowler that you previously provisioned. For instructions, see [Connect to your Linux instance using Session Manager](). If you're unable to connect, see the [Troubleshooting]() section of this pattern.	AWS administrator

Task	Description	Skills required

	3. Navigate to `usr/local/prowler`, and then open the **prowler_scan.sh** file.
	4. Review and modify the adjustable parameters and variables in this script as needed for your environme nt. For more informati on about customization options, see the comments at the beginning of the script.
	
	For example, instead of getting a list of all member accounts in the organization from the management account, you can modify the script to specify the AWS account IDs or AWS Regions that you want to scan, or you can reference an external file that contains these parameters.
	5. Save and close the **prowler_scan.sh** file.
	6. Enter the following commands. This runs the **prowler_scan.sh** script.
	
	`sudo -i`
	`screen`
	`cd /usr/local/`
	`prowler`	

Task	Description	Skills required
	`./prowler_scan.sh` Note the following: • The screen command permits the script to continue running in the event that the connectio n times out or you lose console access. • After the scan starts, you can force a screen detach by pressing **Ctrl+A D**. The screen detaches, and you can close the instance connection and allow the assessment to proceed. • To resume a detached session, connect to the instance, enter `sudo -i` then enter `screen -r`. • To monitor progress of the individual account assessments, you can navigate to the `usr/local/prowler` directory and enter the command `tail -f output/stdout-<account-id>` . 7. Wait for Prowler to complete scans in all accounts. The script assesses multiple accounts	

Task	Description	Skills required
	at the same time. When the assessment is complete in all accounts, you receive a notification if you specified an email address when you deployed the **Prowler-Resources.yaml** file.	
Retrieve the Prowler findings.	1. Download the `prowler-output-<assessDate>.zip` file from the `prowler-output-<accountID>-<region>` bucket. For instructions, see [Downloading an object](#) in the Amazon S3 documentation. 2. Delete all objects in the bucket, including the file you downloaded. This is a best practice for cost optimization and to make sure that you can delete the `Prowler-Resources` CloudFormation stack at any time. For instructions, see [Deleting objects](#) in the Amazon S3 documentation.	General AWS

Task	Description	Skills required
Stop the EC2 instance.	To prevent billing while the instance is idle, stop the EC2 instance that runs Prowler. For instructions, see [Stop and start your instances]() in the Amazon EC2 documentation.	AWS DevOps

Create a report of the findings

Task	Description	Skills required
Import the findings.	1. In Excel, open the **prowler-report-template.xlsx** file, and then choose the **Prowler CSV** worksheet. 2. Delete all of the sample data, including the header row. If you are asked whether to delete the query associated with the data being removed, choose **No**. Deleting the query can affect the functionality of the pivot tables in the Excel template. 3. Extract the contents of the zip file you download from the S3 bucket. 4. In Excel, open the **prowler-fullorgresults-accessdeniedfiltered.txt**. We recommended that you	General AWS

Task	Description	Skills required
	use this file because the most common, non-actionable errors have already been removed, such as `Access Denied` errors related to attempted scans of AWS Control Tower resources. If you want the unfiltered findings, open the **prowler-fullorgresults.txt** file instead. 5. Select column A. 6. If you're using Windows, enter **Ctrl+C**, or if you're using MacOS, enter **Cmd+C**. This copies all of the data to the clipboard. 7. In the Excel report template, on the **Prowler CSV** worksheet, select cell A1. 8. If you're using Windows, enter **Ctrl+V**, or if you're using MacOS, enter **Cmd+V**. This pastes the findings into the report. 9. Confirm all of the cells containing pasted data are selected. If not, select column A. 10. On the **Data** tab, choose **Text to Columns**.	

Task	Description	Skills required
	11. In the wizard, do the following: • For step 1, choose **Delimited**. • For step 2, for **Delimiters**, choose **Semicolon**. In the **Data preview** pane, confirm that the data is being separated into columns. • For step 3, choose **Finish**. 12. Confirm that the text data is delimited across multiple columns. 13. Save the Excel report with a new name. 14. Search and delete any `Access Denied` errors in the findings. For instructions about how to remove these programmatically, see *Programmatically removing errors* in the [Additional information](#) section.	

Task	Description	Skills required
Finalize the report.	1. Choose the **Findings** worksheet, and then select cell A17. This cell is the header of the pivot table. 2. In the ribbon, under **PivotTable Tools**, choose **Analyze**, and then under **Refresh**, choose **Refresh All**. This updates the pivot tables with the new data set. 3. By default, Excel does not properly display AWS account numbers. To fix the number formatting, do the following: • On the **Findings** worksheet, open the context (right-click) menu for column A, and then choose **Format Cells**. • Choose **Number**, and in **Decimal places**, enter `0`. • Choose **OK**. *Note:* If an AWS account number starts with one or more zeros, Excel automatically removes the zeros. If you see an account number that is less than 12 digits in the report, the missing digits are zeros	General AWS

Task	Description	Skills required
	at the beginning of the number. 4. (Optional) You can collapse fields to make the findings easier to read. Do the following: • On the **Findings** worksheet, if you move the cursor to the line between rows 18 and 19 (the space between the critical header and the first finding), the cursor icon changes to a small arrow pointing down. • Click to select all finding fields. • Open the context (right-click) menu, find **Expand/Collapse**, and then choose **Collapse**. 5. For details about the assessment, review the **Findings**, **Severity**, and **Pass Fail** worksheets.	

(Optional) Update Prowler or the resources in the code repository

Task	Description	Skills required
Update Prowler.	If you want to update Prowler to the latest version, do the following:	General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td></td><td>

Connect to the EC2 instance for Prowler by using Session Manager. For instructions, see Connect to your Linux instance using Session Manager.
Enter the following command.
<pre><code>sudo -i
pip3 install --upgrade
 prowler</code></pre>

</td><td></td></tr>
</table>

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Update the prowler_scan.sh script.</td><td>If you want to update the prowler_scan.sh script to the latest version in the repo, do the following:

1. Connect to the EC2 instance for Prowler by using Session Manager. For instructions, see Connect to your Linux instance using Session Manager.
2. Enter the following command.
<pre>sudo -i</pre>3. Navigate to the Prowler script directory.
<pre>cd /usr/local/prowler</pre>4. Enter the following command to stash the local script so that you can merge custom changes into the newest version.
<pre>git stash</pre>5. Enter the following command to get the latest version of the script.
<pre>git pull</pre></td><td>General AWS</td></tr>
</table>

Task	Description	Skills required
	6. Enter the following command to merge the custom script with the latest version of the script. `git stash pop` **Note** You might receive warnings related to any locally generated files that are not in the GitHub repo, such as finding reports. You can ignore these as long as the **prowler_scan.sh** shows that the locally stashed changes are merged back in.	

(Optional) Clean up

Task	Description	Skills required
Delete all deployed resources.	You can leave the resources deployed in the accounts. If you shut down the EC2 instance when it is not in use and keep the S3 bucket empty, this reduces the costs	AWS DevOps

Task	Description	Skills required
	of maintaining the resources for future scans. If you want to deprovision all resources, do the following: 1. Delete the `IAM-ProwlerExecRole` stack provisioned in the management account. For instructions, see [Deleting a stack](#) in the CloudFormation documentation. 2. Delete the `IAM-ProwlerExecRole` stack set provisioned in the organization's management account or in the delegated administrator account. For instructions, see [Delete a stack set](#) in the CloudFormation documentation. 3. Delete all objects in the `prowler-output` S3 bucket. For instructions, see [Deleting objects](#) in the Amazon S3 documentation. 4. Delete the `Prowler-Resources` stack provisioned in the security account. For instructions, see [Deleting a stack](#)	

Task	Description	Skills required
	in the CloudFormation documentation.	

Troubleshooting

Issue	Solution
Unable to connect to the EC2 instance by using Session Manager.	The SSM Agent must be able to communicate with the Systems Manager endpoint. Do the following: 1. Validate the subnet where the EC2 instance is deployed has internet access. 2. Reboot the EC2 instance.
When deploying the stack set, the CloudFormation console prompts you to `Enable trusted access with AWS Organizations to use service-managed permissions` .	This indicates that trusted access has not been enabled between AWS Organizations and CloudFormation. Trusted access is required to deploy the service-managed stack set. Choose the button to enable trusted access. For more information, see Enable trusted access in the CloudFormation documentation.

Related resources

AWS documentation

- Implementing security controls on AWS (AWS Prescriptive Guidance)

Other resources

- Prowler (GitHub)

Additional information

Programmatically removing errors

If the results contain `Access Denied` errors, you should remove them from the findings. These errors are typically due to external influencing permissions that prevent Prowler from assessing a particular resource. For example, some checks fail when reviewing S3 buckets provisioned through AWS Control Tower. You can programmatically extract these results and save the filtered results as a new file.

The following commands remove rows that contain a single text string (a pattern) and then output the results to a new file.

- For Linux or MacOS (Grep)

```
grep -v -i "Access Denied getting bucket" myoutput.csv > myoutput_modified.csv
```

- For Windows (PowerShell)

```
Select-String -Path myoutput.csv -Pattern 'Access Denied getting bucket' -NotMatch >
 myoutput_modified.csv
```

The following commands removes rows that match more than one text string and then output the results to a new file.

- For Linux or MacOS (Uses an escaped pipe between strings)

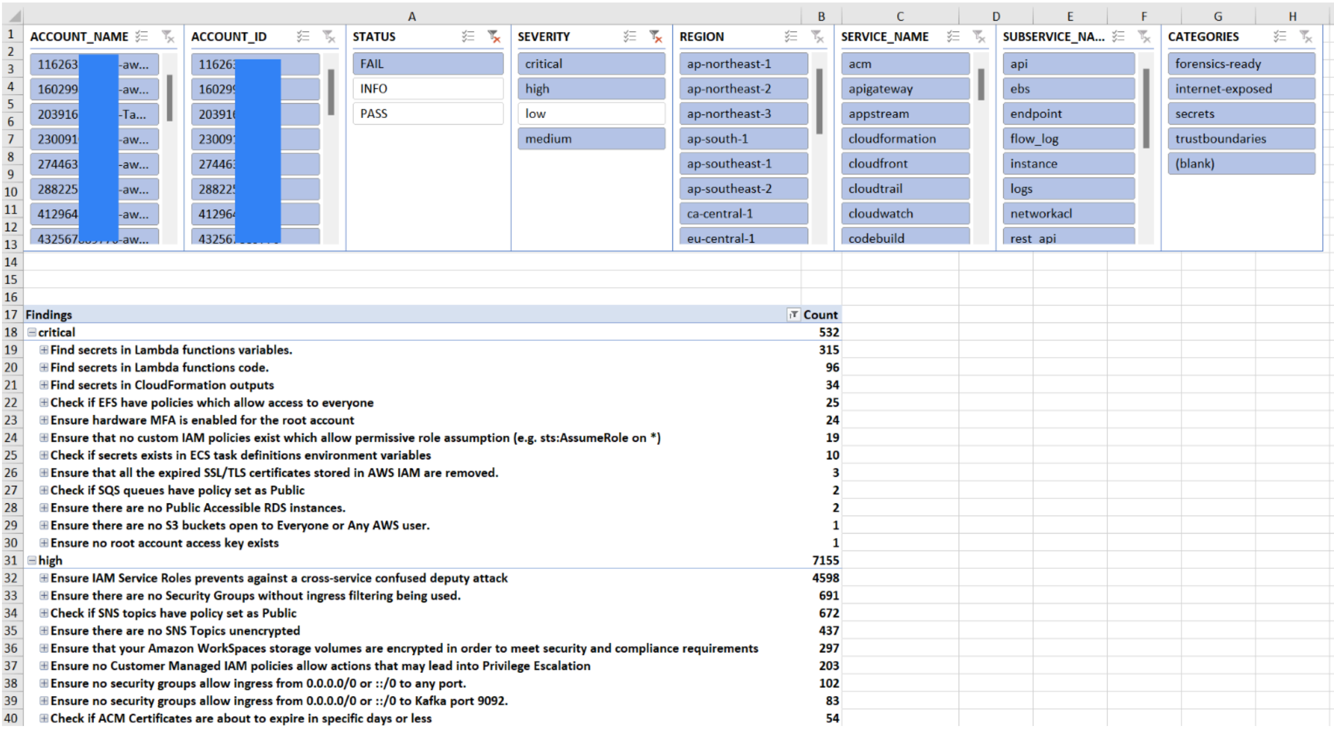
```
grep -v -i 'Access Denied getting bucket\|Access Denied Trying to Get' myoutput.csv >
 myoutput_modified.csv
```

- For Windows (Uses a comma between strings)

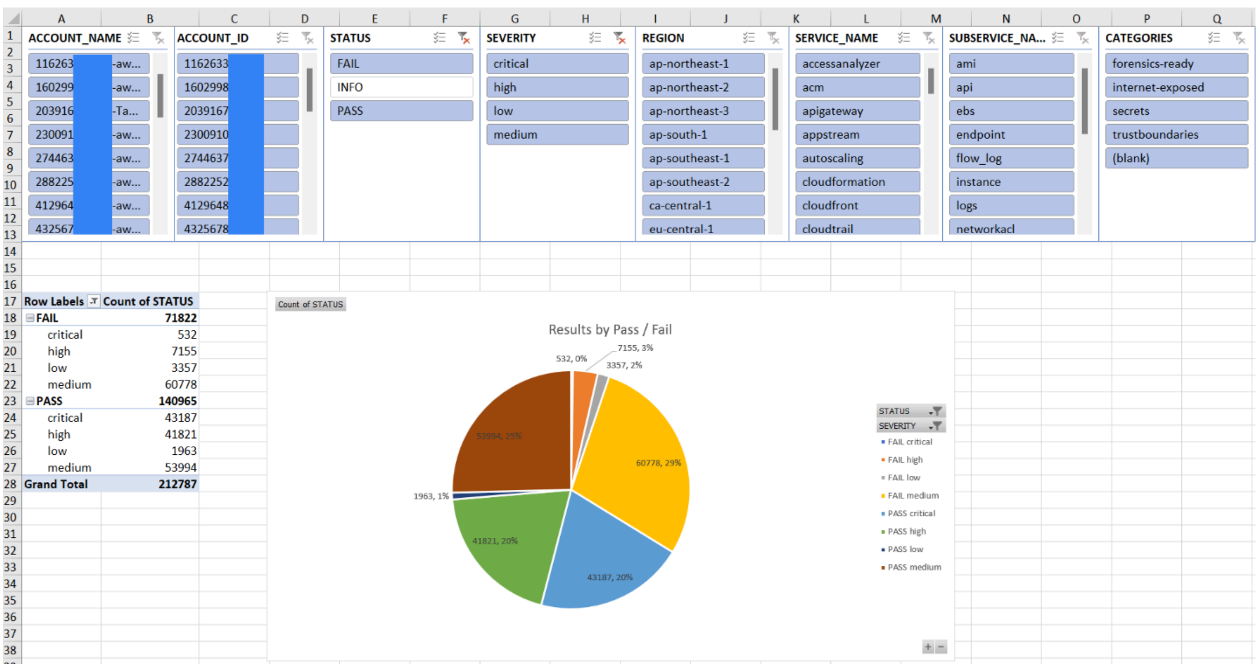
```
Select-String -Path myoutput.csv -Pattern 'Access Denied getting bucket', 'Access
 Denied Trying to Get' -NotMatch > myoutput_modified.csv
```

Report examples

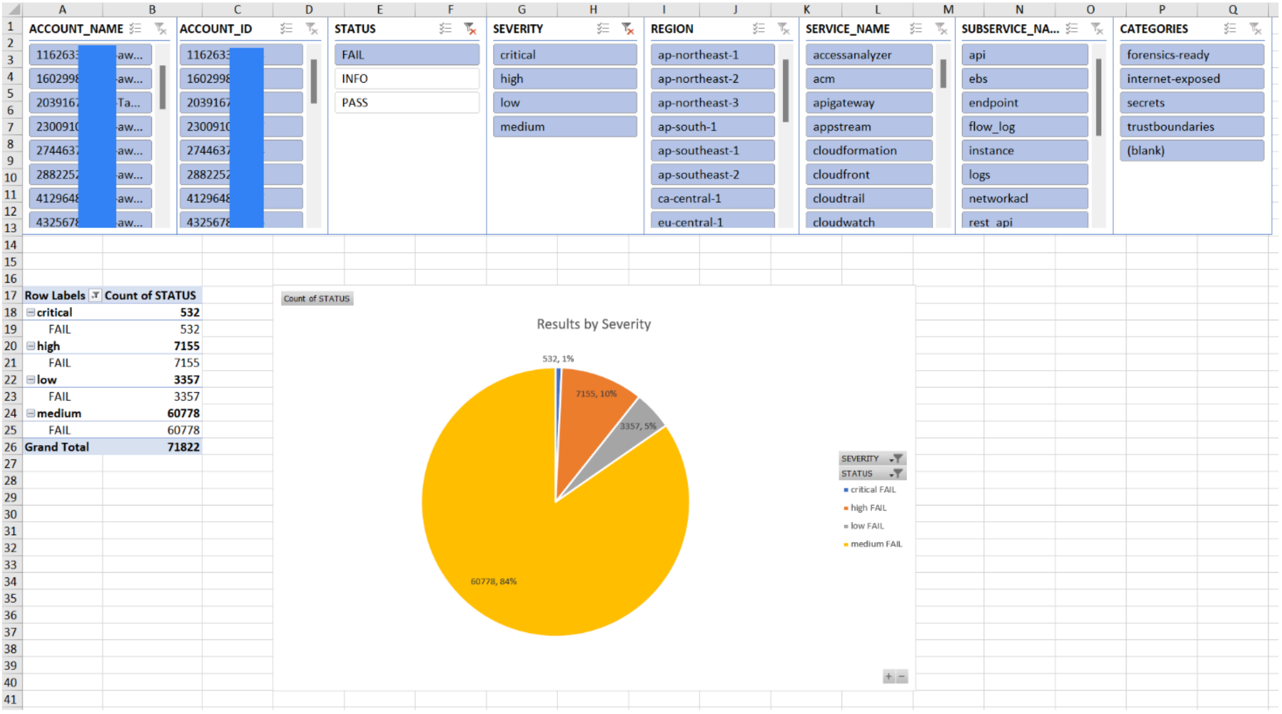
The following image is an example of the **Findings** worksheet in the report of consolidated Prowler findings.



The following image is an example of the **Pass Fail** worksheet in the report of consolidated Prowler findings. (By default, pass results are excluded from the output.)



The following image is an example of the **Severity** worksheet in the report of consolidated Prowler findings.


ACCOUNT_NAME
ACCOUNT_ID
STATUS
FAIL
INFO
PASS
SEVERITY
critical
high
low
medium
REGION
ap-northeast-1
ap-northeast-2
ap-northeast-3
ap-south-1
ap-southeast-1
ap-southeast-2
ca-central-1
eu-central-1
SERVICE_NAME
accessanalyzer
acm
apigateway
appstream
cloudformation
cloudfront
cloudtrail
cloudwatch
SUBSERVICE_NA...
api
ebs
endpoint
flow_log
instance
logs
networkacl
rest_api
CATEGORIES
forensics-ready
internet-exposed
secrets
trustboundaries
(blank)
Row Labels
Count of STATUS
critical 532
FAIL 532
high 7155
FAIL 7155
low 3357
FAIL 3357
medium 60778
FAIL 60778
Grand Total 71822
Count of STATUS
Results by Severity
532, 1%
7155, 10%
3357, 5%
60778, 84%
SEVERITY
STATUS
critical FAIL
high FAIL
low FAIL
medium FAIL

Delete unused Amazon EBS volumes by using AWS Config and AWS Systems Manager

Sankar Sangubotla, Amazon Web Services

Summary

The lifecycle of an Amazon Elastic Block Store (Amazon EBS) volume is typically independent from the lifecycle of the Amazon Elastic Compute Cloud (Amazon EC2) instance to which it is attached. Unless you select **Delete on Termination** option at the time of launch, terminating the Amazon EC2 instance detaches the Amazon EBS volume but doesn't delete it. Especially in development and testing environments where it's common to launch and terminate Amazon EC2 instances, this can result in a large number of unutilized Amazon EBS volumes. Amazon EBS volumes accrue charges in your AWS account, regardless of whether they're being used. Deleting these volumes can help you optimize costs for your AWS accounts. In addition, deleting unused Amazon EBS volumes is a security best practice to prevent access to any unused, potentially sensitive, data in those volumes.

AWS Config can help you manually or automatically remediate noncompliant resources. This pattern describes how to configure an AWS Config rule and automatic remediation action that deletes unused Amazon EBS volumes in the account. The remediation action is a predefined runbook for Automation, a capability of AWS Systems Manager. You can configure the runbook to create a snapshot of the volume before deleting it.

Prerequisites and limitations

Prerequisites

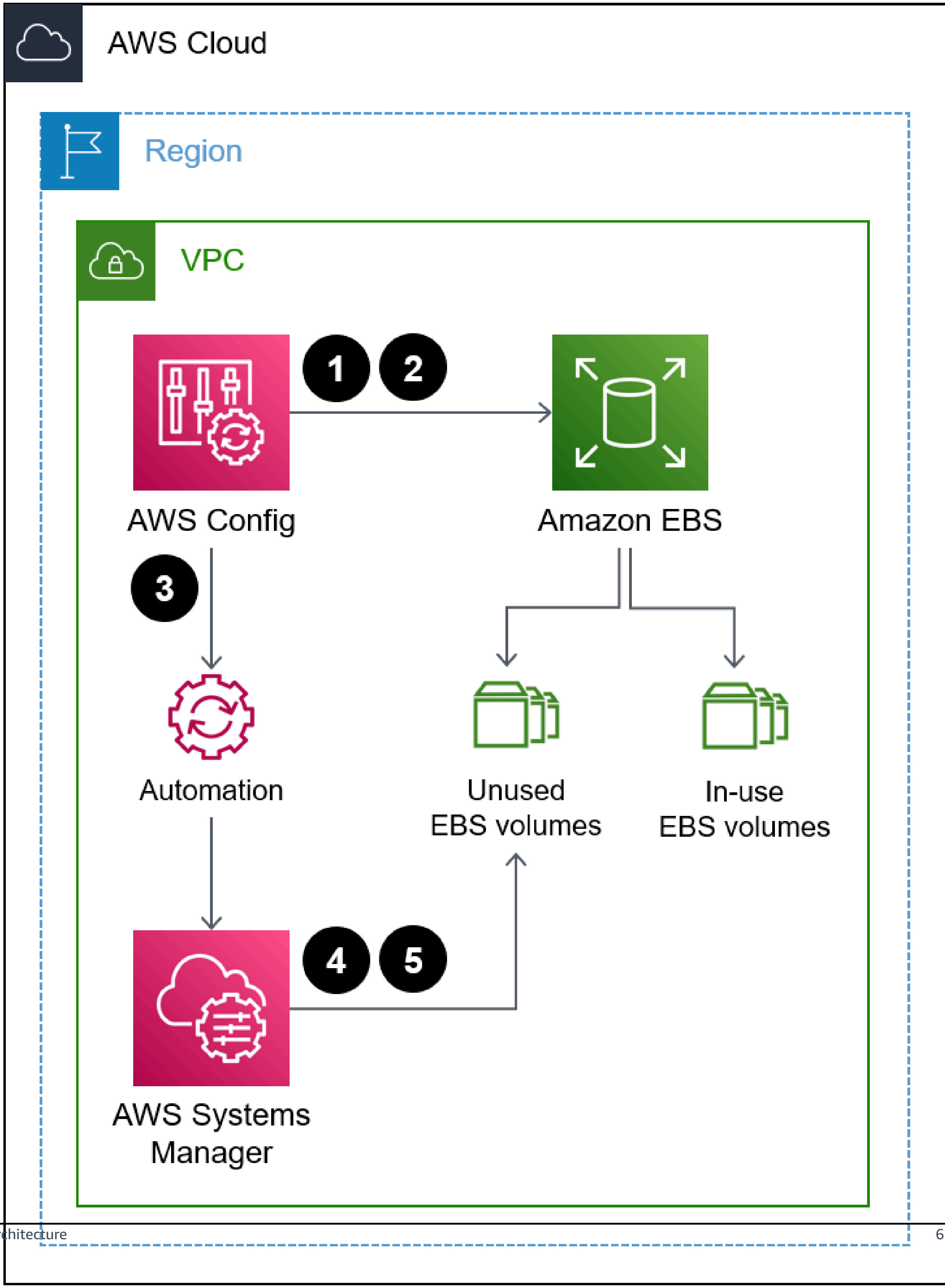
- An active AWS account.
- AWS Identity and Access Management (IAM) permissions to run the `AWSConfigRemediation-DeleteUnusedEBSVolume` runbook for Automation, a capability of Systems Manager. For more information, see *Required IAM permissions* in AWSConfigRemediation-DeleteUnusedEBSVolume.
- One or more unused Amazon EBS volumes.

Limitations

- The unused Amazon EBS volumes must be in the `available` state.

Architecture

Target architecture


AWS Cloud
Region
VPC
1
2
AWS Config
Amazon EBS
3
Automation
Unused
EBS volumes
In-use
EBS volumes
4
5
AWS Systems
Manager

1. The AWS Config rule evaluates the Amazon EBS volumes.
2. The rule returns a list of compliant and noncompliant resources. Amazon EBS volumes that are in the `available` state, which are unused volumes, are determined to be noncompliant.
3. AWS Config automatically starts the Automation runbook.
4. If configured, Systems Manager creates snapshots of the unused volumes before deleting them.
5. Systems Manager deletes the unused Amazon EBS volumes.

Automation and scale

You can apply this solution across all accounts in your organization. For more information, see Managing rules across all accounts in your organization in the AWS Config documentation.

Tools

- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.
- AWS Systems Manager Automation simplifies common maintenance, deployment, and remediation tasks for many AWS services.

Epics

Configure the AWS Config rule

Task	Description	Skills required
Create a role for the Automation runbook.	Create a role called `AssumeRole` . Systems Manager Automation uses this role to run the runbook. For instructions, see Configuring a service role (assume role) access for automations in the	AWS systems administrator

Task	Description	Skills required
	Systems Manager documentation.	
Turn on the AWS Config recorder.	Follow the instructions in Setting Up AWS Config with the Console in the AWS Config documentation to make sure that AWS Config is running and it is configured to record Amazon EBS volumes.	AWS systems administrator
Run the rule.	1. Follow the instructions in Evaluating your resources in the AWS Config documentation to run the `ec2-volume-inuse-check` rule. Wait for the evaluation to complete. 2. On the **Rules** page, select the `ec2-volume-inuse-check` rule, and then for **Resources in scope**, choose **Noncompliant**. 3. Confirm that there are one or more unused Amazon EBS volumes in the evaluation results.	AWS systems administrator

Configure automatic remediation of unused Amazon EBS volumes

Task	Description	Skills required
Add the automatic remediation action.	1. On the **Rules** page, select the `ec2-volume-inuse-check` rule. 2. Follow the instructions in [Setting up automatic remediation](#) in the AWS Config documentation. Note the following: 3. In the **Remediation action details** section, choose `AWSConfigRemediation-DeleteUnusedEBSVolume` . • Select **Resource ID parameter**, and then in the list, choose **VolumeId**. At runtime, this parameter is substituted with the ID of the noncompliant Amazon EBS volume. • In the **Parameters** section, provide values for the following parameters: • `CreateSnapshot` – (Optional) If set to `true`, the automation creates a snapshot of the Amazon EBS volume before it's deleted.	AWS systems administrator

Task	Description	Skills required
	• `AutomationAssumeRole` – Enter the Amazon Resource Name (ARN) of the `AssumeRole` service role that you created previously.	
Test the automatic remediation for the AWS Config rule.	1. In the AWS Config console, on the **Rules** page, select the `ec2-volume-inuse-check` rule. 2. In the **Action**s menu, choose **Re-evaluate**. 3. Allow the rule to evaluate the non-compliant resources, and then confirm that the unused Amazon EBS volumes are deleted.	AWS systems administrator

Troubleshooting

Issue	Solution
AWS Config doesn't accurately reflect the resource state.	Sometimes, AWS Config doesn't update the state of the resources. Turn the recorder off and then turn it back on again on the AWS Config **Settings** page. The recorder captures the state of the resources. For newly created or deleted resources, it might take some time for the recorder to reflect the current state. For more information about Amazon

Issue	Solution
	EBS volume states, see Volume states in the Amazon EBS documentation.

Related resources

- AWSConfigRemediation-DeleteUnusedEBSVolume runbook
- ec2-volume-inuse-check rule
- Remediating noncompliant AWS resources with AWS Config rules

Deploy and manage AWS Control Tower controls by using AWS CDK and CloudFormation

Iker Reina Fuente and Ivan Girardi, Amazon Web Services

Summary

This pattern describes how to use AWS CloudFormation and AWS Cloud Development Kit (AWS CDK) to implement and administer preventive, detective, and proactive AWS Control Tower controls as infrastructure as code (IaC). A control (also known as a *guardrail*) is a high-level rule that provides ongoing governance for your overall AWS Control Tower environment. For example, you can use controls to require logging for your AWS accounts and then configure automatic notifications if specific security-related events occur.

AWS Control Tower helps you implement preventive, detective, and proactive controls that govern your AWS resources and monitor compliance across multiple AWS accounts. Each control enforces a single rule. In this pattern, you use a provided IaC template to specify which controls you want to deploy in your environment.

AWS Control Tower controls apply to an entire organizational unit (OU), and the control affects every AWS account within the OU. Therefore, when users perform any action in any account in your landing zone, the action is subject to the controls that govern the OU.

Implementing AWS Control Tower controls helps establish a strong security foundation for your AWS landing zone. By using this pattern to deploy the controls as IaC through CloudFormation and AWS CDK, you can standardize the controls in your landing zone and more efficiently deploy and manage them. This solution uses cdk_nag to scan the AWS CDK application during deployment. This tool checks the application for adherence to AWS best practices.

To deploy AWS Control Tower controls as IaC, you can also use HashiCorp Terraform instead of AWS CDK. For more information, see Deploy and manage AWS Control Tower controls by using Terraform.

Intended audience

This pattern is recommended for users who have experience with AWS Control Tower, CloudFormation, AWS CDK, and AWS Organizations.

Prerequisites and limitations

Prerequisites

- Active AWS accounts managed as an organization in AWS Organizations and an AWS Control Tower landing zone. For instructions, see Getting started in the AWS Control Tower documentation.
- AWS Command Line Interface (AWS CLI), installed and configured.
- Node package manager (npm), installed and configured for the AWS CDK.
- Prerequisites for AWS CDK.
- Permissions to assume an existing AWS Identity and Access Management (IAM) role in a deployment account.
- Permissions to assume an IAM role in the organization's management account that that can be used to bootstrap AWS CDK. The role must have permissions to modify and deploy CloudFormation resources. For more information, see Bootstrapping in the AWS CDK documentation.
- Permissions to create IAM roles and policies in the organization's management account. For more information, see Permissions required to access IAM resources in the IAM documentation.

Limitations

- This pattern provides instructions for deploying this solution across AWS accounts, from a deployment account to the organization's management account. For testing purposes, you can deploy this solution directly in the management account, but instructions for this configuration are not explicitly provided.
- For AWS Control Tower controls, this pattern requires the use of global identifiers that are in the following format:

  ```
  arn:<PARTITION>:controlcatalog:::control/<CONTROL_CATALOG_OPAQUE_ID>
  ```

  Previous versions of this pattern used regional identifiers that are no longer supported. We recommend that you migrate from Regional identifiers to global identifiers. Global identifiers help you manage controls and expand the number of controls you can use.

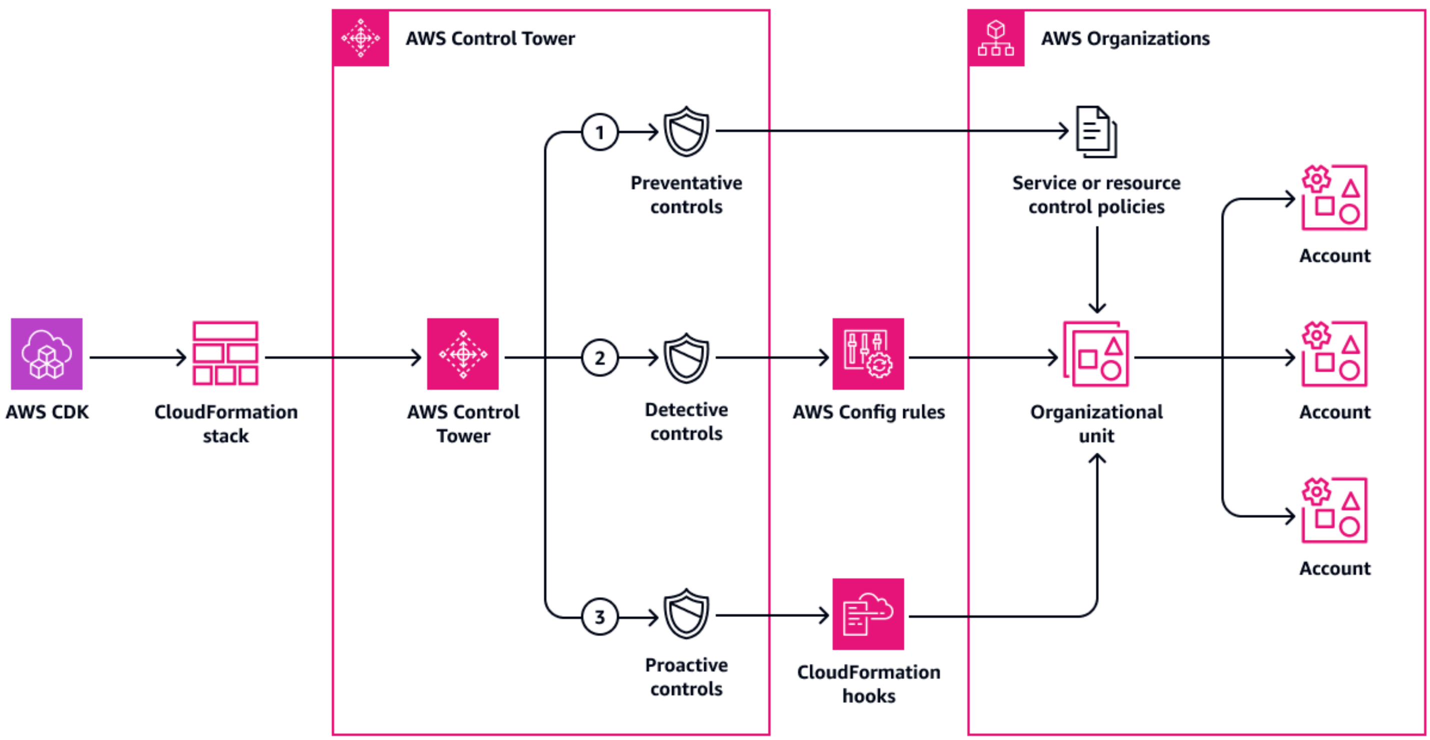
> **Note**
>
> In most cases, the value for `<PARTITION>` is `aws`.

Product versions

- AWS Control Tower version 3.2 or later
- Python version 3.9 or later
- npm version 8.9.0 or later

Architecture

This section provides a high-level overview of this solution and the architecture established by the sample code. The following diagram shows controls deployed across the various accounts in the OU.



AWS Control Tower controls are categorized according to their *behavior* and their *guidance*.

There are three primary types of control behaviors:

1. *Preventive controls* are designed to prevent actions from occurring. These are implemented with service control policies (SCPs) or resource control policies (RCPs) in AWS Organizations. The status of a preventive control is either **enforced** or **not enabled**. Preventive controls are supported in all AWS Regions.
2. *Detective controls* are designed to detect specific events when they occur and log the action in AWS CloudTrail. These are implemented with AWS Config rules. The status of a detective control is either **clear**, **in violation**, or **not enabled**. Detective controls apply only in those AWS Regions supported by AWS Control Tower.
3. *Proactive controls* scan resources that would be provisioned by AWS CloudFormation and check whether they are compliant with your company policies and objectives. Resources that are not compliant will not be provisioned. These are implemented with AWS CloudFormation hooks. The status of a proactive control is **PASS**, **FAIL**, or **SKIP**.

Control *guidance* refers to the recommended practice for how to apply each control to your OUs. AWS Control Tower provides three categories of guidance: *mandatory*, *strongly recommended*, and *elective*. The guidance of a control is independent of its behavior. For more information, see Control behavior and guidance.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code. The AWS CDK Toolkit is the primary tool for interacting with your AWS CDK app.
- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.
- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time.
- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.

Other tools

- cdk_nag is an open-source tool that uses a combination of rule packs to check AWS CDK applications for adherence to best practices.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.
- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub Deploy AWS Control Tower controls using AWS CDK repository. You use the **cdk.json** file to interact with the AWS CDK app, and you use the **package.json** file to install the npm packages.

Best practices

- Follow the principle of least-privilege (IAM documentation). The sample IAM policy and trust policy provided in this pattern include the minimum permissions required, and the AWS CDK stacks created in the management account are restricted by these permissions.
- Follow the Best practices for AWS Control Tower administrators (AWS Control Tower documentation).
- Follow the Best practices for developing and deploying cloud infrastructure with the AWS CDK (AWS CDK documentation).
- When bootstrapping the AWS CDK, customize the bootstrap template to define policies and the trusted accounts that should have the ability to read and write to any resource in the management account. For more information, see Customizing bootstrapping.
- Use code analysis tools, such as cfn_nag, to scan the generated CloudFormation templates. The cfn-nag tool looks for patterns in CloudFormation templates that might indicate the infrastructure is not secure. You can also use cdk-nag to check your CloudFormation templates by using the cloudformation-include module.

Epics

Prepare to enable the controls

Task	Description	Skills required
Create the IAM role in the management account.	1. Create an IAM policy in the management account with the permissions defined in *IAM policy* in the Additional information section. For instructions, see Creating IAM policies in the IAM documentation. Make note of the Amazon Resource Name (ARN) of the policy. The following is an example ARN. `arn:aws:iam::<MANAGEMENT-ACCOUNT-ID>:policy/<POLICY-NAME>` 2. Create an IAM role in the management account, attach the IAM permission policy that you created in the previous step, and attach the custom trust policy in *Trust policy* in the Additional information section. For instructions, see Creating a role using custom trust policies in the IAM documentation. The following is an example ARN for the new role.	DevOps engineer, General AWS

Task	Description	Skills required
	`arn:aws:iam:: <MANAGEMENT-ACCOUNT-ID>:role/<ROLE-NAME>`	

Task	Description	Skills required
Bootstrap AWS CDK.	1. In the management account, assume a role that has permissions to bootstrap AWS CDK. 2. Enter the following command, replacing the following: • `<MANAGEMENT-ACCOUNT-ID>` is the ID of the organization's management account. • `<AWS-CONTROL-TOWER-REGION>` is the AWS Region where AWS Control Tower is deployed. For a complete list of Region codes, see Regional endpoints in *AWS General Reference*. • `<DEPLOYMENT-ACCOUNT-ID>` is the ID of the deployment account. • `<DEPLOYMENT-ROLE-NAME>` is the name of the IAM role you are using the deployment account. • `<POLICY-NAME>` is the name of the policy you created in the management account. `$ npx cdk bootstrap aws://<MANAGEMENT-ACCOUNT-ID>/<AWS-C`	DevOps engineer, General AWS, Python

Task	Description	Skills required
	``` ONTROL-TOWER-REGIO N> \ --trust arn:aws:i am::<DEPLOYMENT-AC COUNT-ID>:role/<DE PLOYMENT-ROLE-NAME> \ --cloudformation- execution-policies arn:aws:iam::<MANA GEMENT-ACCOUNT-ID> :policy/<POLICY-NA ME> ```	
Clone the repository.	In a bash shell, enter the following command. This clones the Deploy AWS Control Tower controls using AWS CDK repository from GitHub. ``` git clone https://g ithub.com/aws-samp les/aws-control-to wer-controls-cdk.git ```	DevOps engineer, General AWS

Task	Description	Skills required
Edit the AWS CDK configura tion file.	1. In the cloned repository, open the **constants.py** file. 2. In the `ACCOUNT_ID` parameter, enter the ID of your management account. 3. In the `<AWS-CONTROL-TOWER-REGION>` parameter, enter the AWS Region where AWS Control Tower is deployed. 4. In the `ROLE_ARN` parameter, enter the ARN of the role you created in the management account. 5. Open [All global identifiers](#) in the AWS Control Tower documentation. 6. In the JSON-formatted list, locate the control that you want to implement, and then copy its global identifier (also known as the {CONTROL_ CATALOG_OPAQUE_ID} value). For example, the global identifier for the **AWS-GR_AUDIT_BUCKE T_ENCRYPTION_ENABL ED** control is `k4izcjxhu kijhajp6ks5mjxk` . 7. In the `GUARDRAIL S_CONFIGURATION` section, in the `Enable-`	DevOps engineer, General AWS

Task	Description	Skills required
	`Control` parameter, enter the global identifier that you copied. Enter the identifier in double quotation marks, and separate multiple identifiers with commas. 8. In the `GUARDRAILS_CONFIGURATION` section, in the `OrganizationalUnitIds` parameter, enter the ID of the organizational unit where you want to enable the control, such as `ou-1111-11111111` . Enter the ID in double quotation marks, and separate multiple IDs with commas. For more information about how to retrieve OU IDs, see [Viewing the details of an OU](#). 9. Save and close the **constants.py** file. For an example of an updated **constants.py** file, see the [Additional information](#) section of this pattern.	

Enable controls in the management account

Task	Description	Skills required
Assume the IAM role in the deployment account.	In the deployment account, assume the IAM role that has permissions to deploy the AWS CDK stacks in the management account. For more information about assuming an IAM role in the AWS CLI, see [Use an IAM role in the AWS CLI]().	DevOps engineer, General AWS

Activate the environment.	If you are using Linux or MacOS:
	
	1. Enter the following command to create a virtual environment.
	`$ python3 -m venv .venv`
	2. After the virtual environment is created, enter the following command to activate it.
	`$ source .venv/bin/activate`
	
	If you are using Windows:
	
	1. Enter the following command to activate a virtual environment.
		DevOps engineer, General AWS

Task	Description	Skills required
	`% .venv\Scripts\activate.bat`	
Install the dependencies.	After the virtual environment is activated, enter the following command to run the **install_deps.sh** script. This script installs the required dependencies. `$ ./scripts/install_deps.sh`	DevOps engineer, General AWS, Python
Deploy the stack.	Enter the following commands to synthesize and deploy the CloudFormation stack. `$ npx cdk synth` `$ npx cdk deploy`	DevOps engineer, General AWS, Python

Related resources

AWS documentation

- About controls (AWS Control Tower documentation)
- Controls library (AWS Control Tower documentation)
- AWS CDK Toolkit commands (AWS CDK documentation)
- Deploy and manage AWS Control Tower controls by using Terraform (AWS Prescriptive Guidance)

Other resources

- Python

Additional information

Example constants.py file

The following is an example of an updated **constants.py** file. This sample enables the **AWS-GR_DISALLOW_CROSS_REGION_NETWORKING** control (global ID: `dvuaav61i5cnfazfelmvn9m6k`) and the **AWS-GR_SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED** control (global ID: `50z1ot237wl8u1lv5ufau6qqo`). For a list of global IDs, see [All global identifiers](#) in the AWS Control Tower documentation.

```
ACCOUNT_ID = 111122223333
AWS_CONTROL_TOWER_REGION = us-east-2
ROLE_ARN = "arn:aws:iam::111122223333:role/CT-Controls-Role"
GUARDRAILS_CONFIGURATION = [
    {
        "Enable-Control": {
            "dvuaav61i5cnfazfelmvn9m6k": {  # AWS-GR_DISALLOW_CROSS_REGION_NETWORKING
                "Parameters": {
                    "ExemptedPrincipalArns": ["arn:aws:iam::111122223333:role/
RoleName"]
                },
                "Tags": [{"key": "Environment", "value": "Production"}]
            },
            ...
        },
        "OrganizationalUnitIds": ["ou-1111-11111111", "ou-2222-22222222"...],
    },
    {
        "Enable-Control": {
            "50z1ot237wl8u1lv5ufau6qqo",  # AWS-
GR_SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED
            ...
        },
        "OrganizationalUnitIds": ["ou-2222-22222222"...],
    },
]
```

IAM policy

The following sample policy allows the minimum actions required to enable or disable AWS Control Tower controls when deploying AWS CDK stacks from a deployment account to the management account.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "controltower:EnableControl",
                "controltower:DisableControl",
                "controltower:GetControlOperation",
                "controltower:ListEnabledControls",
                "organizations:AttachPolicy",
                "organizations:CreatePolicy",
                "organizations:DeletePolicy",
                "organizations:DescribeOrganization",
                "organizations:DescribeOrganizationalUnit",
                "organizations:DetachPolicy",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListChildren",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:ListParents",
                "organizations:ListPoliciesForTarget",
                "organizations:ListRoots",
                "organizations:UpdatePolicy",
                "ssm:GetParameters"
            ],
            "Resource": "*"
        }
    ]
}
```

Trust policy

The following custom trust policy allows a specific IAM role in the deployment account to assume the IAM role in the management account. Replace the following:

- `<DEPLOYMENT-ACCOUNT-ID>` is the ID of the deployment account

- <DEPLOYMENT-ROLE-NAME> is the name of the role in the deployment account that is allowed to assume the role in the management account

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::<DEPLOYMENT-ACCOUNT-ID>:role/<DEPLOYMENT-ROLE-
NAME>"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

Deploy and manage AWS Control Tower controls by using Terraform

Iker Reina Fuente and Ivan Girardi, Amazon Web Services

Summary

This pattern describes how to use AWS Control Tower controls, HashiCorp Terraform, and infrastructure as code (IaC) to implement and administer preventive, detective, and proactive security controls. A control (also known as a *guardrail*) is a high-level rule that provides ongoing governance for your overall AWS Control Tower environment. For example, you can use controls to require logging for your AWS accounts and then configure automatic notifications if specific security-related events occur.

AWS Control Tower helps you implement preventive, detective, and proactive controls that govern your AWS resources and monitor compliance across multiple AWS accounts. Each control enforces a single rule. In this pattern, you use a provided IaC template to specify which controls you want to deploy in your environment.

AWS Control Tower controls apply to an entire organizational unit (OU), and the control affects every AWS account within the OU. Therefore, when users perform any action in any account in your landing zone, the action is subject to the controls that govern the OU.

Implementing AWS Control Tower controls helps establish a strong security foundation for your AWS landing zone. By using this pattern to deploy the controls as IaC through Terraform, you can standardize the controls in your landing zone and more efficiently deploy and manage them.

To deploy AWS Control Tower controls as IaC, you can also use AWS Cloud Development Kit (AWS CDK) instead of Terraform. For more information, see Deploy and manage AWS Control Tower controls by using AWS CDK and AWS CloudFormation.

Intended audience

This pattern is recommended for users who have experience with AWS Control Tower, Terraform, and AWS Organizations.

Prerequisites and limitations

Prerequisites

- Active AWS accounts managed as an organization in AWS Organizations and an AWS Control Tower landing zone. For instructions, see Getting started in the AWS Control Tower documentation.
- AWS Command Line Interface (AWS CLI), installed and configured.
- An AWS Identity and Access Management (IAM) role in the management account that has permissions to deploy this pattern. For more information about the required permissions and a sample policy, see *Least privilege permissions for the IAM role* in the Additional information section of this pattern.
- Permissions to assume the IAM role in the management account.
- Terraform CLI, installed (Terraform documentation).
- Terraform AWS Provider, configured (Terraform documentation).
- Terraform backend, configured (Terraform documentation).

Limitations

- For AWS Control Tower controls, this pattern requires the use of global identifiers that are in the following format:

```
arn:<PARTITION>:controlcatalog:::control/<CONTROL_CATALOG_OPAQUE_ID>
```

  Previous versions of this pattern used regional identifiers that are no longer supported. We recommend that you migrate from Regional identifiers to global identifiers. Global identifiers help you manage controls and expand the number of controls that you can use.

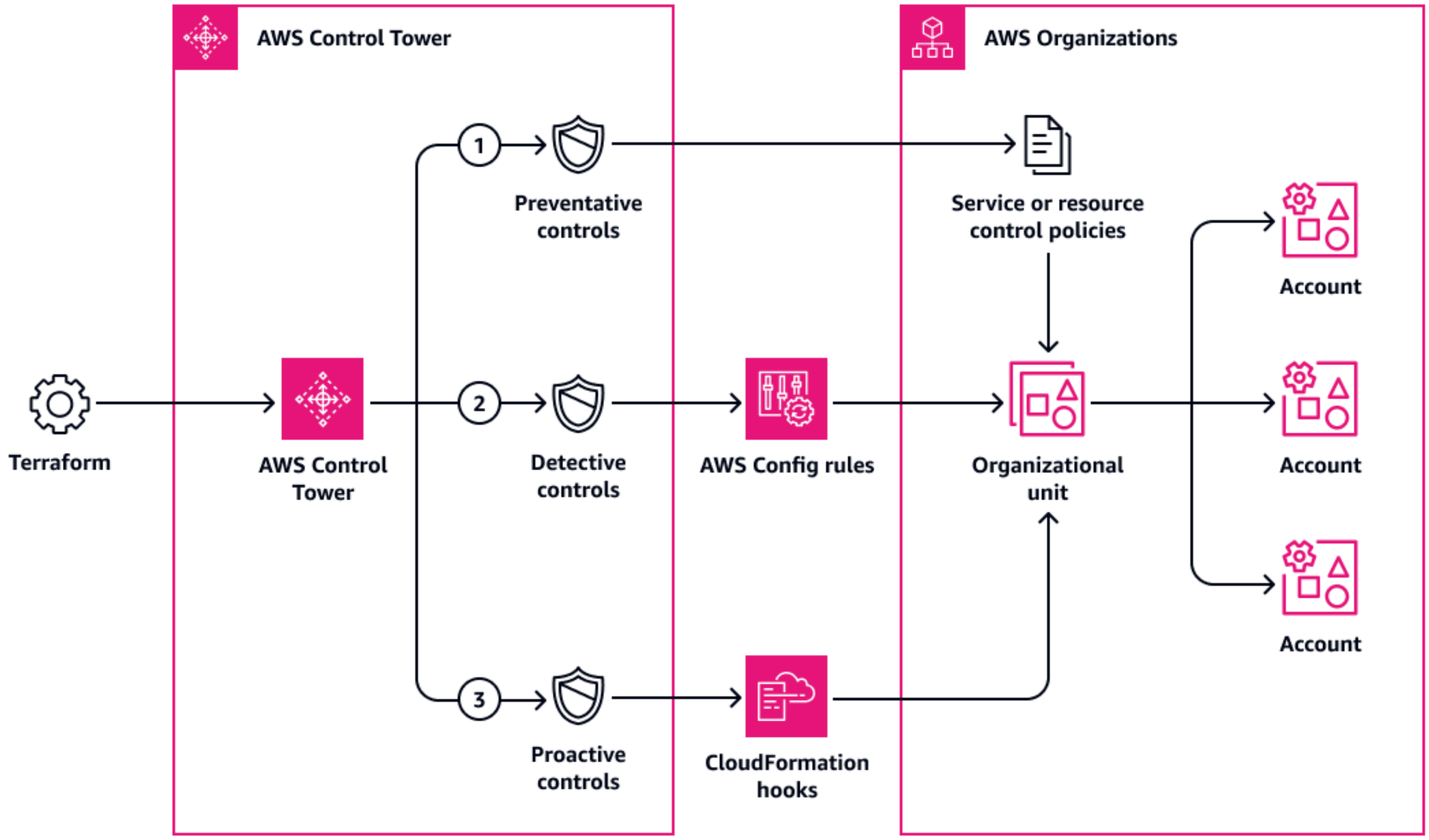
> **Note**
>
> In most cases, the value for `<PARTITION>` is `aws`.

Product versions

- AWS Control Tower version 3.2 or later
- Terraform version 1.5 or later
- Terraform AWS Provider version 4.67 or later

Architecture

This section provides a high-level overview of this solution and the architecture established by the sample code. The following diagram shows controls deployed across the various accounts in the OU.



AWS Control Tower controls are categorized according to their *behavior* and their *guidance*.

There are three primary types of control behaviors:

1. *Preventive controls* are designed to prevent actions from occurring. These are implemented with service control policies (SCPs) or resource control policies (RCPs) in AWS Organizations. The status of a preventive control is either **enforced** or **not enabled**. Preventive controls are supported in all AWS Regions.
2. *Detective controls* are designed to detect specific events when they occur and log the action in AWS CloudTrail. These are implemented with AWS Config rules. The status of a detective control is either **clear**, **in violation**, or **not enabled**. Detective controls apply only in those AWS Regions supported by AWS Control Tower.

3. *Proactive controls* scan resources that would be provisioned by AWS CloudFormation and check whether they are compliant with your company policies and objectives. Resources that are not compliant will not be provisioned. These are implemented with AWS CloudFormation hooks. The status of a proactive control is **PASS**, **FAIL**, or **SKIP**.

Control *guidance* is the recommended practice for how to apply each control to your OUs. AWS Control Tower provides three categories of guidance: *mandatory*, *strongly recommended*, and *elective*. The guidance of a control is independent of its behavior. For more information, see Control behavior and guidance.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time.
- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.

Other tools

- HashiCorp Terraform is an infrastructure as code (IaC) tool that helps you use code to provision and manage cloud infrastructure and resources.

Code repository

The code for this pattern is available in the GitHub Deploy and manage AWS Control Tower controls by using Terraform repository.

Best practices

- The IAM role used to deploy this solution should adhere to the principle of least-privilege (IAM documentation).
- Follow the Best practices for AWS Control Tower administrators (AWS Control Tower documentation).

Epics

Enable controls in the management account

Task	Description	Skills required
Clone the repository.	In a bash shell, enter the following command. This clones the Deploy and manage AWS Control Tower controls by using Terraform repository from GitHub. `git clone https://github.com/aws-samples/aws-control-tower-controls-terraform.git`	DevOps engineer
Edit the Terraform backend configuration file.	1. In the cloned repository, open the **backend.tf** file. 2. Edit the file to set the Terraform backend configuration. The configuration you define in this file depends on your environment. For more information, see Backend configuration (Terraform documentation).	DevOps engineer, Terraform

Task	Description	Skills required
	3. Save and close the **backend.tf** file.	
Edit the Terraform provider configuration file.	1. In the cloned repository, open the **provider.tf** file. 2. Edit the file to set the Terraform provider configuration. For more information, see [Provider configuration](#) (Terraform documentation). Set the AWS Region as the Region where the AWS Control Tower API is available. 3. Save and close the **provider.tf** file.	DevOps engineer, Terraform

Task	Description	Skills required
Edit the configuration file.	1. In the cloned repository, open the **variables.tfvars** file. 2. Open [All global identifiers](#) in the AWS Control Tower documentation. 3. In the JSON-formatted list, locate the control that you want to implement, and then copy its global identifier (also known as the {CONTROL_CATALOG_OPAQUE_ID} value). For example, the global identifier for the **AWS-GR_AUDIT_BUCKET_ENCRYPTION_ENABLED** control is `k4izcjxhukijhajp6ks5mjxk` . 4. In the `controls` section, in the `control_names` parameter, enter the global identifier that you copied. 5. In the `controls` section, in the `organizational_unit_ids` parameter, enter the ID of the organizational unit where you want to enable the control, such as `ou-1111-11111111` . Enter the ID in double quotation marks, and	DevOps engineer, General AWS, Terraform

Task	Description	Skills required
	separate multiple IDs with commas. For more information about how to retrieve OU IDs, see [Viewing the details of an OU](). 6. Save and close the **variables.tfvars** file. For an example of an updated **variables.tfvars** file, see the [Additional information]() section of this pattern.	
Assume the IAM role in the management account.	In the management account, assume the IAM role that has permissions to deploy the Terraform configuration file. For more information about the permissions required and a sample policy, see *Least privilege permissions for the IAM role* in the [Additional information]() section. For more information about assuming an IAM role in the AWS CLI, see [Use an IAM role in the AWS CLI]().	DevOps engineer, General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy the configuration file.</td><td>
1. Enter the following command to initialize Terraform.

```
$ terraform init -
upgrade
```

2. Enter the following command to preview the changes compared the current state.

```
$ terraform plan -
var-file="variabl
es.tfvars"
```

3. Review the configuration changes in the Terraform plan and confirm that you want to implement these changes in the organizat ion.
4. Enter the following command to deploy the resources.

```
$ terraform apply -
var-file="variabl
es.tfvars"
```
</td><td>DevOps engineer, General AWS, Terraform</td></tr>
</table>

(Optional) Disable controls in the AWS Control Tower management account

Task	Description	Skills required
Run the `destroy` command.	Enter the following command to remove the resources deployed by this pattern. `$ terraform destroy -var-file="variables.tfvars"`	DevOps engineer, General AWS, Terraform

Troubleshooting

Issue	Solution
`Error: creating ControlTower Control ValidationException: Guardrail <control ID> is already enabled on organizational unit <OU ID>` error	The control you are trying to enable is already enabled in the target OU. This error can occur if a user manually enabled the control through the AWS Management Console, through AWS Control Tower or through AWS Organizations. To deploy the Terraform configuration file, you can use either of the following options. **Option 1: Update the Terraform current state file** You can import the resource to the Terraform current state file. When you rerun the `apply` command, Terraform will skip this resource. Do the following to import the resource to the current Terraform state: 1. In the AWS Control Tower managemen t account, enter the following command to retrieve a list of the Amazon Resource Names (ARNs) for the OUs, where `<root-`

Issue	Solution

	ID> is the organization root. For more information about retrieving this ID, see Viewing the details of the root.
	`aws organizations list-organizational-units-for-parent --parent-id <root-ID>`
	2. For each OU returned in the previous step, enter the following command, where <OU-ARN> is the ARN of the OU.
	`aws controltower list-enabled-controls --target-identifier <OU-ARN>`
	3. Copy the ARNs and perform the Terraform import in the required module so that it is included in the Terraform state. For instructions, see Import (Terraform documentation).
	4. Repeat the steps in *Deploy the configuration* in the Epics section.
	
	Option 2: Disable the control
	
	If you are working in a non-production environment, you can disable the control in the console. Re-enable it by repeating the steps in *Deploy the configuration* in the Epics section. This approach is not recommended for production environments because there is a period of time when the control will be disabled. If you want to use this option in a production environment, you can implement temporary controls, such as temporarily applying a SCP in AWS Organizations.

Related resources

AWS documentation

- About controls (AWS Control Tower documentation)
- Controls library (AWS Control Tower documentation)
- Deploy and manage AWS Control Tower controls by using AWS CDK and AWS CloudFormation (AWS Prescriptive Guidance)

Other resources

- Terraform
- Terraform CLI documentation

Additional information

Example variables.tfvars file

The following is an example of an updated **variables.tfvars** file. This sample enables the **AWS-GR_ENCRYPTED_VOLUMES** control (global ID: `503uicglhjkokaajywfpt6ros`) and the **AWS-GR_SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED** control (global ID: `50z1ot237wl8u1lv5ufau6qqo`). For a list of global IDs, see All global identifiers in the AWS Control Tower documentation.

The following example also enables controls that have parameters such as **CT.S3.PV.5** (global ID: `7mo7a2h2ebsq71l8k6uzr96ou`) and **CT.SECRETSMANAGER.PV.1** (global ID: `dvhe47fxg5o6lryqrq9g6sxg4`). For a list of controls with parameters, see Controls with parameters in the AWS Control Tower documentation.

```
controls = [
    {
        control_names = [
            "503uicglhjkokaajywfpt6ros", # AWS-GR_ENCRYPTED_VOLUMES
            ...
        ],
        organizational_unit_ids = ["ou-1111-11111111", "ou-2222-22222222"...],
    },
    {
        control_names = [
```

```
            "50z1ot237wl8u1lv5ufau6qqo", # AWS-GR_SUBNET_AUTO_ASSIGN_PUBLIC_IP_DISABLED
            ...
        ],
        organizational_unit_ids = ["ou-1111-11111111"...],
    },
]

controls_with_params = [
  {
    control_names = [
      { "7mo7a2h2ebsq71l8k6uzr96ou" = { # CT.S3.PV.5
        parameters = {
          "ExemptedPrincipalArns" : ["arn:aws:iam::*:role/RoleName"],
          "ExemptedResourceArns" : [],
        }
      } },
      { "dvhe47fxg5o6lryqrq9g6sxg4" = { # CT.SECRETSMANAGER.PV.1
        parameters = {
          "ExemptedPrincipalArns" : ["arn:aws:iam::*:role/RoleName"],
        }
      } },
      ...
    ],
    organizational_unit_ids = ["ou-1111-11111111"...]
  },
  {
    control_names = [
      { "dvuaav61i5cnfazfelmvn9m6k" = { # AWS-GR_DISALLOW_CROSS_REGION_NETWORKING
        parameters = {
          "ExemptedPrincipalArns" : ["arn:aws:iam::*:role/RoleName"],
        }
      } },
      { "41ngl8m5c4eb1myoz0t707n7h" = { # AWS-GR_DISALLOW_VPC_INTERNET_ACCESS
        parameters = {
          "ExemptedPrincipalArns" : ["arn:aws:iam::*:role/RoleName"],
        }
      } },
      ...
    ],
    organizational_unit_ids = ["ou-2222-22222222"...]
  }
]
```

Least privilege permissions for the IAM role

This pattern requires that you assume an IAM role in the management account. Best practice is to assume a role with temporary permissions and limit the permissions according to the principle of least privilege. The following sample policy allows the minimum actions required to enable or disable AWS Control Tower controls.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "controltower:EnableControl",
                "controltower:DisableControl",
                "controltower:GetControlOperation",
                "controltower:ListEnabledControls",
                "organizations:AttachPolicy",
                "organizations:CreatePolicy",
                "organizations:DeletePolicy",
                "organizations:DescribeOrganization",
                "organizations:DetachPolicy",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListChildren",
                "organizations:ListOrganizationalUnitsForParent",
                "organizations:ListParents",
                "organizations:ListPoliciesForTarget",
                "organizations:ListRoots",
                "organizations:UpdatePolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

Deploy the Security Automations for AWS WAF solution by using Terraform

Dr. Rahul Sharad Gaikwad and Tamilselvan P, Amazon Web Services

Summary

AWS WAF is a web application firewall that helps protect applications from common exploits by using customizable rules, which you define and deploy in *web access control lists* (ACLs). Configuring AWS WAF rules can be challenging, especially for organizations that do not have dedicated security teams. To simplify this process, Amazon Web Services (AWS) offers the Security Automations for AWS WAF solution, which automatically deploys a single web ACL with a set of AWS WAF rules that filters web-based attacks. During Terraform deployment, you can specify which protective features to include. After you deploy this solution, AWS WAF inspects web requests to existing Amazon CloudFront distributions or Application Load Balancers, and blocks any requests that don't match the rules.

The Security Automations for AWS WAF solution can be deployed by using AWS CloudFormation according to the instructions in the Security Automations for AWS WAF Implementation Guide. This pattern provides an alternative deployment option for organizations that use HashiCorp Terraform as their preferred infrastructure as code (IaC) tool to provision and manage their cloud infrastructure. When you deploy this solution, Terraform automatically applies the changes in the cloud and deploys and configures the AWS WAF settings and protective features.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2.4.25 or later, installed and configured with necessary permissions. For more information, see Getting started (AWS CLI documentation).
- Terraform version 1.1.9 or later, installed and configured. For more information, see Install Terraform (Terraform documentation).

Architecture

Target architecture

This pattern deploys the Security Automations for AWS WAF solution. For more information about the target architecture, see Architecture overview in the *Security Automations for AWS WAF Implementation Guide*. For more information about the AWS Lambda automations in this deployment, the Application log parser, the AWS WAF log parser, the IP lists parser, and the Access handler, see Component details in the *Security Automations for AWS WAF Implementation Guide*.

Terraform deployment

When you run `terraform apply`, Terraform does the following:

1. Terraform creates AWS Identity and Access Management (IAM) roles and Lambda functions based on the inputs from the **testing.tfvars** file.
2. Terraform creates AWS WAF ACL rules and IP sets based on the inputs from the **testing.tfvars** file.
3. Terraform creates the Amazon Simple Storage Service (Amazon S3) buckets, Amazon EventBridge rules, AWS Glue database tables, and Amazon Athena work groups based on the inputs from the **testing.tfvars** file.
4. Terraform deploys the AWS CloudFormation stack to provision the custom resources.
5. Terraform creates the Amazon API Gateway resources based on the given inputs from **testing.tfvars** file.

Automation and scale

You can use this pattern to create AWS WAF rules for multiple AWS accounts and AWS Regions to deploy the Security Automations for AWS WAF solution throughout your AWS Cloud environment.

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS WAF is a web application firewall that helps you monitor HTTP and HTTPS requests that are forwarded to your protected web application resources.

Other services

- Git is an open-source, distributed version control system.
- HashiCorp Terraform is a command-line interface application that helps you use code to provision and manage cloud infrastructure and resources.

Code repository

The code for this pattern is available in the GitHub AWS WAF Automation Using Terraform repository.

Best practices

- Put static files in separate Amazon S3 buckets.
- Avoid hardcoding variables.
- Limit the use of custom scripts.
- Adopt a naming convention.

Epics

Set up your local workstation

Task	Description	Skills required
Install Git.	Follow the instructions in Getting started (Git website) to install Git on your local workstation.	DevOps engineer
Clone the repository.	On your local workstation, enter the following command to clone the code repository: `git clone https://github.com/aws-samples/aws-waf-automation-terraform-samples.git`	DevOps engineer

Task	Description	Skills required
Update the variables.	1. Navigate into the cloned directory by entering the following command: `cd terraform-aws-waf-automation` 2. In any text editor, open the **testing.tfvars** file. 3. Update the values of the variables in the **testing.t fvars** file. 4. Save and close the file.	DevOps engineer

Provision the target architecture using Terraform

Task	Description	Skills required
Initialize the Terraform configuration.	Enter the following command to initialize your working directory that contains the Terraform configuration files: `terraform init`	DevOps engineer
Preview the Terraform plan.	Enter the following command. Terraform evaluates the configura tion files to determine the target state for the declared resources. It then compares the target state against the current state and creates a plan:	DevOps engineer

Task	Description	Skills required
	`terraform plan -var-file ="testing.tfvars"`	
Verify the plan.	Review the plan and confirm that it configures the required architecture in your target AWS account.	DevOps engineer
Deploy the solution.	1. Enter the following command to apply the plan: `terraform apply -var-file="testing .tfvars"` 2. Enter `yes` to confirm. Terraform creates, updates, or destroys infrastru cture to achieve the target state declared in the configuration files. For more information about the sequence, see *Terraform deployment* in the [Architecture] section of this pattern.	DevOps engineer

Validate and clean up

Task	Description	Skills required
Verify the changes.	1. In the Terraform console, verify that the outputs match the expected results.	DevOps engineer

Task	Description	Skills required
	2. Sign in to the AWS Management Console. 3. Verify the outputs in the Terraform console have been successfully deployed in your AWS account.	
(Optional) Clean up the infrastructure.	If you want to remove all resources and configuration changes made by this solution, do the following: 1. In the Terraform console, enter the following command: `terraform destroy -var-file="testing.tfvars"` 2. Enter `yes` to confirm.	DevOps engineer

Troubleshooting

Issue	Solution
`WAFV2 IPSet: WAFOptimisticLockException` error	If you receive this error when you run the `terraform destroy` command, you must manually delete the IP sets. For instructions, see Deleting an IP set (AWS WAF documentation).

Related resources

AWS references

- Security Automations for AWS WAF Implementation Guide
- Security Automations for AWS WAF (AWS Solutions Library)
- Security Automations for AWS WAF FAQ

Terraform references

- Terraform backend configuration
- Terraform AWS Provider - Documentation and Usage
- Terraform AWS Provider (GitHub repository)

Deploy a pipeline that simultaneously detects security issues in multiple code deliverables

Benjamin Morris, Tim Hahn, Sapeksh Madan, Dina Odum, and Isaiah Schisler, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more.

The Simple Code Scanning Pipeline (SCSP) provides two-click creation of a code analysis pipeline that runs industry-standard open-source security tools in parallel. This enables developers to check the quality and security of their code without having to install tools or even understand how to run them. This helps you reduce vulnerabilities and misconfigurations in code deliverables. It also reduces the amount of time your organization spends installing, researching, and configuring security tools.

Before SCSP, scanning code using this particular suite of tools required developers to locate, manually install, and configure the software analysis tools. Even locally installed, all-in-one tools, such as Automated Security Helper (ASH), require configuring a Docker container in order to run. However, with SCSP, a suite of industry-standard code analysis tools runs automatically in the AWS Cloud. With this solution, you use Git to push your code deliverables, and then you receive a visual output with at-a-glance insights into which security checks failed.

Prerequisites and limitations

- An active AWS account
- One or more code deliverables that you want to scan for security issues
- AWS Command Line Interface (AWS CLI), installed and configured
- Python version 3.0 or later and pip version 9.0.3 or later, installed
- Git, installed
- Install git-remote-codecommit on your local workstation

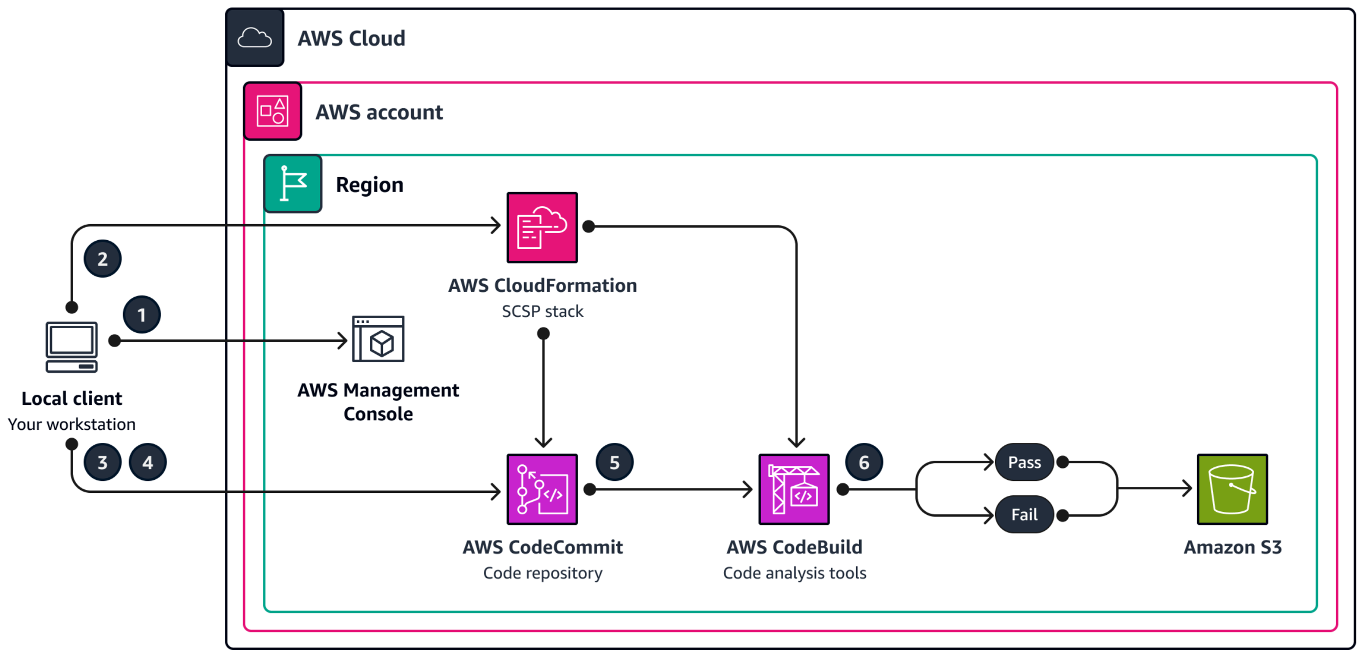
Architecture

Target technology stack

- AWS CodeCommit repository
- AWS CodeBuild project
- AWS CodePipeline pipeline
- Amazon Simple Storage Service (Amazon S3) bucket
- AWS CloudFormation template

Target architecture

The SCSP for static code analysis is a DevOps project designed to give security feedback on deliverable code.



1. In the AWS Management Console, log into the target AWS account. Confirm that you are in the AWS Region where you want to deploy the pipeline.
2. Use the CloudFormation template in the code repository to deploy the SCSP stack. This creates a new CodeCommit repository and CodeBuild project.

> **Note**
>
> As an alternative deployment option, you can use an existing CodeCommit repo by providing the Amazon Resource Name (ARN) of the repository as a parameter during stack deployment.

3. Clone the repository to your local workstation, and then add any files to their respective folders in the cloned repository.
4. Use Git to add, commit, and push the files to the CodeCommit repository.
5. Pushing to the CodeCommit repository initiates a CodeBuild job. The CodeBuild project uses the security tools to scan the code deliverables.
6. Review the output of the pipeline. Security tools that found error-level issues will result in failed actions in the pipeline. Fix these errors or suppress them as false positives. Review details of the tool output in the **Action details** in CodePipeline or in the pipeline's S3 bucket.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.

Other tools

For a complete list of tools that SCSP uses to scan code deliverables, see the SCSP readme in GitHub.

Code repository

The code for this pattern is available in the Simple Code Scanning Pipeline (SCSP) repository in GitHub.

Epics

Deploy the SCSP

Task	Description	Skills required
Create the CloudFormation stack.	1. Sign in to the [AWS Management Console](https://console.aws.amazon.com/). 2. In the console, confirm that you're in the target Region where you want to deploy the solution. For more information, see [Choosing a Region](#). 3. Choose the following link. This opens the Quick create stack wizard in CloudFormation. https://console.aws.amazon.com/cloudformation/home?#/stacks/create/review?templateURL=https://proservetools.s3.amazonaws.com/cft/scsp-pipeline-stack.template.json&stackName=SimpleCodeScanPipeline 4. On the **Quick create stack** wizard, review the parameter settings for your stack and make any modifications as needed for your use case.	AWS DevOps, AWS administrator

Task	Description	Skills required
	5. Select **I acknowledge that AWS CloudFormation might create IAM resources**, and then choose **Create stack**. This creates a CodeCommit repository, a CodePipeline pipeline, several CodeBuild job definitions, and an S3 bucket. Build runs and scanning results are copied into this bucket. After the CloudFormation stack has been completely deployed, SCSP is ready to use.	

Use the pipeline

Task	Description	Skills required
Examine the results of the scan.	1. In the [Amazon S3 console](https://console.aws.amazon.com/s3/), in **Buckets**, choose the **simplecodescanpipeline-deleteresourcespipelinereso** bucket. 2. Choose the **scan_results** directory, and then choose the folder with the most recent scan date stamp. 3. Review the log files in this folder to review any issues detected by the security tools used in the pipeline.	App developer, AWS DevOps

Task	Description	Skills required
	Security tools that found error-level issues will result in `failed` actions in the pipeline. These need to be fixed or suppressed if they are false positives. **Note** You can also view details of the tool output (for both passing and failing scans) in the CodePipeline console, in the **Action details** section.	

Troubleshooting

Issue	Solution
HashiCorp Terraform or AWS CloudFormation files aren't being scanned.	Make sure that Terraform (.tf) and CloudFormation (.yml, .yaml, or .json) files are placed in the appropriate folders in the cloned CodeCommit repository.
The `git clone` command is failing.	Make sure that you have installed `git-remote-codecommit` and that your CLI has access to AWS credentials that have permissions to read the CodeCommit repository.

Issue	Solution
A concurrency error, such as `Project-level concurrent build limit cannot exceed the account-level concurrent build limit of 1 .`	Rerun the pipeline by choosing the **Release Change** button in the CodePipeline console. This is a known issue that seems to be most common during the first few times that the pipeline runs.

Related resources

Provide feedback on the SCSP project.

Additional information

FAQ

Is the SCSP project the same as Automated Security Helper (ASH)?

No. Use ASH when you want a CLI tool that runs code-scanning tools by using containers. Automated Security Helper (ASH) is a tool that is designed to reduce the probability of a security violation in new code, infrastructure, or IAM resource configuration. ASH is a command-line utility that can be run locally. Local use requires a container environment be installed and operational on the system.

Use SCSP when you want an easier setup pipeline than ASH. SCSP requires no local installations. SCSP is designed to run checks individually in a pipeline and display results by tool. SCSP also avoids a lot of the overhead with setting up Docker, and it is operating system (OS) agnostic.

Is SCSP just for security teams?

No, anyone can deploy the pipeline to determine which parts of their code are failing security checks. For example, non-security users can use SCSP to check their code before reviewing with their security teams.

Can I use SCSP if I'm working with another type of repository, such as GitLab, GitHub, or Bitbucket?

You can configure a local git repository to point to two different remote repositories. For example, you could clone an existing GitLab repository, create a SCSP instance (specifying CloudFormation, Terraform, and AWS Config Rules Development Kit (AWS RDK) folders, if needed), and then use `git remote add upstream <SCSPGitLink>` to point the local repository at the SCSP CodeCommit repository as well. This allows for code changes to be sent to SCSP first, validated, then, after any additional updates are made to address findings, pushed to the GitLab, GitHub, or Bitbucket repository. For more information about multiple remotes, see Push commits to an additional Git repository (AWS blog post).

> **Note**
>
> Be careful of drift, such as avoid making changes through web interfaces.

Contributing and adding your own actions

SCSP setup is maintained as a GitHub project, which contains the source code for the SCSP AWS Cloud Development Kit (AWS CDK) application. To add additional checks to the pipeline, the AWS CDK application needs to be updated and then synthesized or deployed into the target AWS account where the pipeline will run. To do this, start by cloning the SCSP GitHub project, and then find the stack definition file in the `lib` folder.

If there's an additional check you would like to add, the `StandardizedCodeBuildProject` class in the AWS CDK code makes it very straightforward to add actions. Provide the name, description, and `install` or `build` commands. AWS CDK creates the CodeBuild project by using sensible default values. In addition to creating the build project, you need to add it to the CodePipeline actions in the build stage. When designing a new check, the action should `FAIL` if the scanning tool detects problems or fails to run. The action should `PASS` if the scanning tool doesn't detect any problems. For an example of configuring a tool, review the code for the `Bandit` action.

For more information about expected input and outputs, see the repository documentation.

If you add custom actions, you need to deploy SCSP by using `cdk deploy` or `cdk synth + CloudFormation deploy`. This is because the **Quick create stack** CloudFormation template is maintained by the repo owners.

Deploy detective attribute-based access controls for public subnets by using AWS Config

Alberto Menendez, Amazon Web Services

Summary

Distributed edge network architectures rely on network edge security that runs alongside the workloads in their virtual private clouds (VPCs). This provides unprecedented scalability in comparison to the more common, centralized approach. Although deploying public subnets in workload accounts can provide benefits, it also introduces new security risks because it increases the attack surface. We recommend that you deploy only Elastic Load Balancing resources, such as Application Load Balancers, or NAT gateways in the public subnets of these VPCs. Using load balancers and NAT gateways in dedicated public subnets helps you implement fine-grained control for inbound and outbound traffic.

We recommend that you implement both preventative and detective controls to limit the types of resources that can be deployed in public subnets. For more information about using attribute-based access control (ABAC) to deploy preventative controls for public subnets, see [Deploy preventative attribute-based access controls for public subnets](). Although effective for most situations, these preventative controls might not address all possible use cases. Therefore, this pattern builds on the ABAC approach and helps you configure alerts about noncompliant resources that are deployed in public subnets. The solution checks whether elastic network interfaces belong to a resource that is not allowed in public subnets.

To achieve this, this pattern uses [AWS Config custom rules]() and [ABAC](). The custom rule processes the configuration of an elastic network interface whenever it is created or modified. At a high level, this rule performs two actions to determine whether the network interface is compliant:

1. To determine whether the network interface is in scope of the rule, the rule checks whether the subnet has specific [AWS tags]() that indicate it is a public subnet. For example, this tag might be `IsPublicFacing=True`.
2. If the network interface is deployed in a public subnet, the rule checks which AWS service created this resource. If the resource is not an Elastic Load Balancing resource or NAT gateway, it marks the resource as noncompliant.

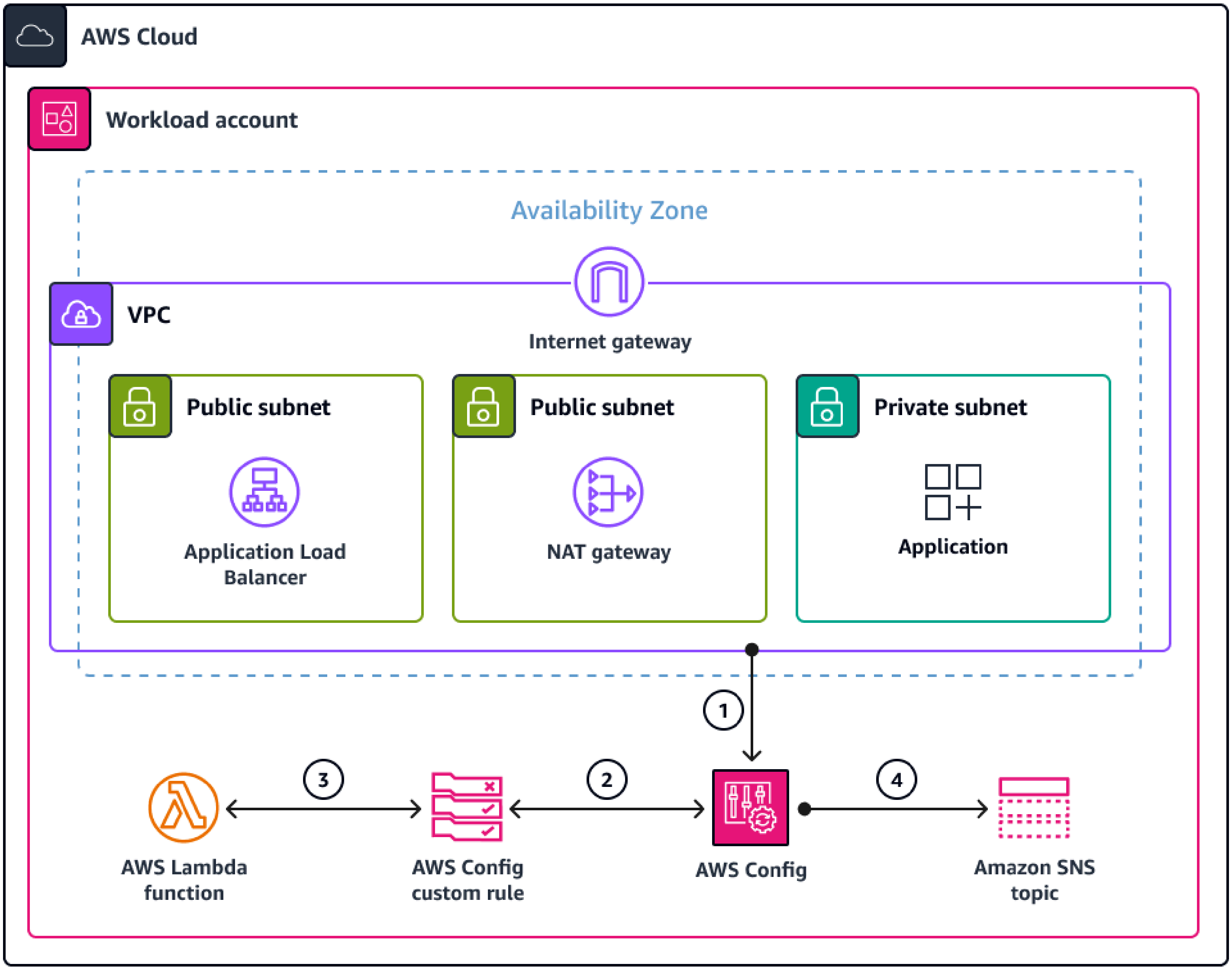
Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Config, set up in the workload account
- Permissions to deploy the required resources in the workload account
- A VPC with public subnets
- Tags properly applied to identify the target public subnets
- (Optional) An organization in AWS Organizations
- (Optional) A central security account that is the delegated administrator for AWS Config and AWS Security Hub

Architecture

Target architecture



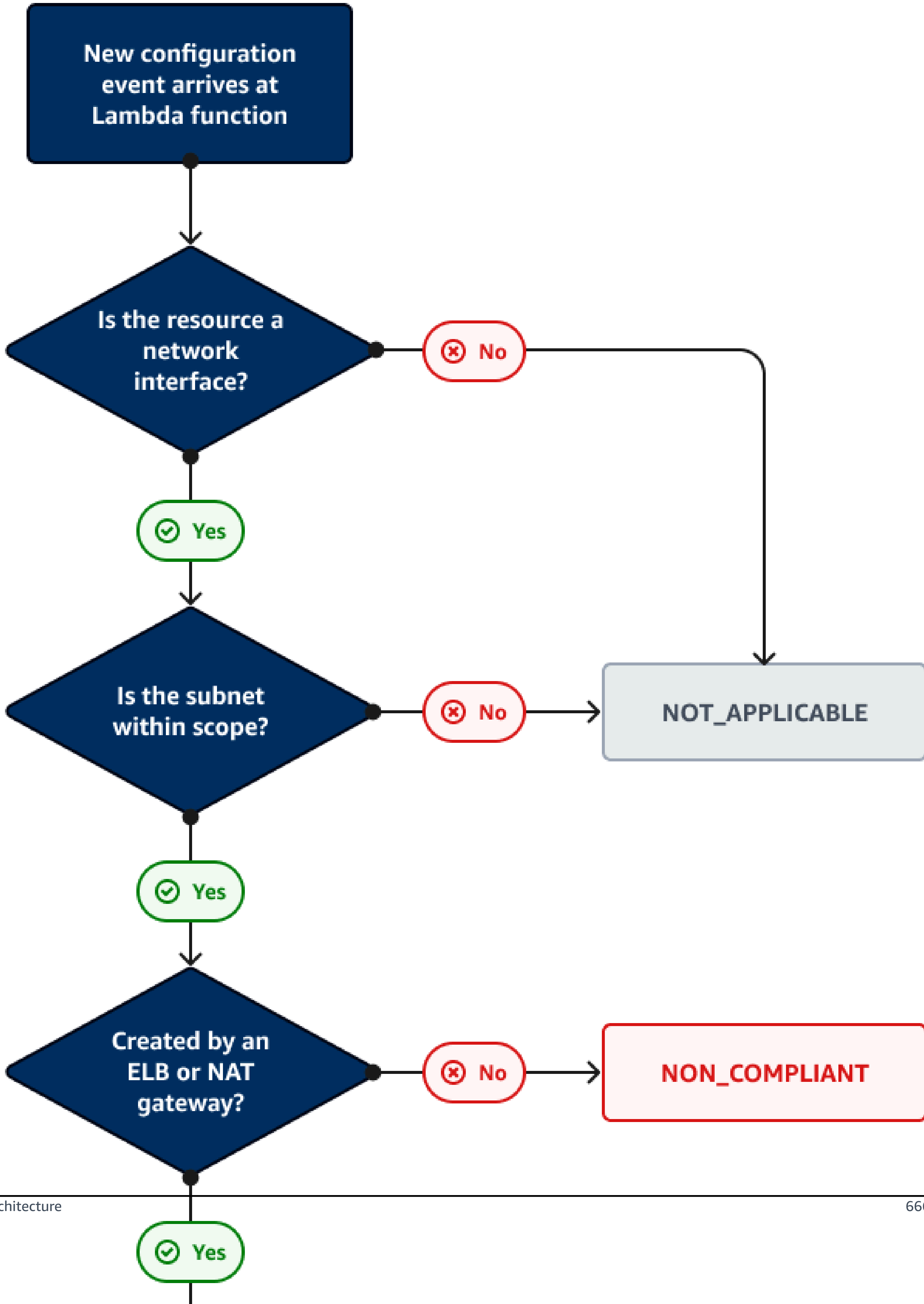
The diagram illustrates the following:

1. When an elastic network interface resource (`AWS::EC2::NetworkInterface`) is deployed or modified, AWS Config captures the event and the configuration.
2. AWS Config matches this event against the custom rule used to evaluate the configuration.
3. The AWS Lambda function associated with this custom rule is invoked. The function evaluates the resource and applies the specified logic to determine if the resource configuration is `COMPLIANT`, `NON_COMPLIANT` or `NOT_APPLICABLE`.
4. If a resource is determined to be `NON_COMPLIANT`, AWS Config sends an alert through Amazon Simple Notification Service (Amazon SNS).

Note: If this account is a member account in AWS Organizations, you can send compliance data to a central security account through AWS Config or AWS Security Hub.

Lambda function evaluation logic

The following diagram shows the logic applied by the Lambda function to evaluate the compliance of the elastic network interface.


New configuration event arrives at Lambda function
Is the resource a network interface?
No
Yes
Is the subnet within scope?
No
NOT_APPLICABLE
Yes
Created by an ELB or NAT gateway?
No
NON_COMPLIANT
Yes

Automation and scale

This pattern is a detective solution. You can also complement it with a remediation rule to automatically resolve any noncompliant resources. For more information, see Remediating Noncompliant Resources with AWS Config Rules.

You can scale this solution by:

- Enforcing application of the corresponding AWS tags that you establish to identify public-facing subnets. For more information, see Tag policies in the AWS Organizations documentation.
- Configuring a central security account that applies the AWS Config custom rule to every workload account in the organization. For more information, see Automate configuration compliance at scale in AWS (AWS blog post).
- Integrating AWS Config with AWS Security Hub in order to capture, centralize, and notify at scale. For more information, see Configuring AWS Config in the AWS Security Hub documentation.

Tools

- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time.
- Elastic Load Balancing distributes incoming application or network traffic across multiple targets. For example, you can distribute traffic across Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, and IP addresses in one or more Availability Zones.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Best practices

For more examples and best practices for developing custom AWS Config rules, see the official [AWS Config Rules Repository] on GitHub.

Epics

Deploy the solution

Task	Description	Skills required
Create the Lambda function.	1. Sign in to the AWS Management Console, and then open the AWS Lambda console. 2. On the **Functions** page, choose **Create function**. 3. Select **Author from scratch**. 4. In the **Basic information** pane, for **Function name**, enter a name. 5. For **Runtime**, choose **Python 3.12**. 6. Leave **architecture** set to **x86_64**. 7. Choose **Create function**. 8. Choose the **Code** tab. 9. In the file explorer, choose **lambda_function.py**. 10. Paste the sample code provided in the Additional information section of this pattern into the **lambda_function.py** tab. Customize the sample code	General AWS

Task	Description	Skills required
	to identify any custom evaluation logic in the `evaluate_change_notification_compliance` function. 11. Choose **Deploy**.	

Task	Description	Skills required

Add permissions to the Lambda function's execution role.	1. In the navigation pane, choose **Functions**.
	2. Choose the function you just created.
	3. Choose **Configuration**, and then choose **Permissions**.
	4. Choose the role name to open the role in the AWS Identity and Access Management (IAM) console.
	5. Under **Permissions policies**, choose **Add permissions**, and then choose **Create inline policy**.
	6. Choose **JSON**.
	7. Paste the following policy into the policy editor. This allows the Lambda function to:
	• Get the details of the subnet tags.
	• Send the compliance outcome back to AWS Config.
	
	`{`
	`    "Version": "2012-10-17",`
	`    "Statement": [`
	`        {`
		General AWS

Task	Description	Skills required
	``` "Action": ["config:PutEvaluations", "ec2:DescribeSubnets"], "Resource": "*", "Effect": "Allow" }] } ``` 8. Choose **Next**. 9. Enter a name for the policy, and then choose **Create policy**.	
Retrieve the Lambda function Amazon Resource Name (ARN).	1. Open the Lambda console. 2. In the navigation pane, choose **Functions**. 3. Choose the function you just created. 4. In the **Function overview** section, under **Function ARN**, copy the value.	General AWS

Task	Description	Skills required
Create the AWS Config custom rule.	1. Open the [AWS Config console](https://console.aws.amazon.com/config/). 2. On the **Rules** page, choose **Add rule**. 3. On the **Specify rule type** page, choose **Create custom Lambda rule**, and then choose **Next**. 4. On the **Configure rule** page, do the following: a. Enter a name and description. b. For **AWS Lambda function ARN**, paste the ARN that you previously copied. c. For **Trigger type**, choose **When configuration changes**. d. For **Scope of changes**, select **Resources**. e. For **Resource type**, choose **AWS EC2 NetworkInterface**. f. Choose **Next**. 5. On the **Review and create** page, verify your rule, and then choose **Save**.	General AWS

Task	Description	Skills required
Configure notifications.	1. Follow the instructions in Creating an Amazon SNS topic in order to create an Amazon SNS topic. 2. Follow the instructions in Subscribing to an Amazon SNS topic to configure an endpoint that receives notifications for the Amazon SNS topic. 3. Follow the instructions in How can I be notified when an AWS resource is non-compliant using AWS Config to configure a custom Amazon EventBridge rule for your noncompliant resources.	General AWS

Test the solution

Task	Description	Skills required
Create a compliant resource.	1. Use the following instructions to create one of the supported resources in a public subnet: • Create a NAT gateway • Getting started with Network Load Balancers • Create an Application Load Balancer	General AWS

Task	Description	Skills required
	2. After the resource is created, the AWS Config custom rule evaluates the elastic network interfaces associated with resource. It marks these network interfaces as `COMPLIANT` . You can view the resources in AWS Config by following these steps: a. Open the [AWS Config console](https://console.aws.amazon.com/config/). b. On the **Rules** page, choose your rule. c. On the **Rule detail** page, go to the bottom of the page. d. Under **Resources in scope**, select **Compliant** . Confirm that you see the IDs of the network interfaces that were created. e. For more detail about the network interface configuration, choose the resource ID.	

Task	Description	Skills required
Create a noncompliant resource.	1. Use the following instructions to create a noncompliant resource in a public subnet: • [Launch an Amazon EC2 instance](#) • [Creating an Amazon Relational Database Service (Amazon RDS) database instance](#) • [Create a VPC endpoint](#) 2. After the resource is created, the AWS Config custom rule evaluates the elastic network interfaces associated with resource. It marks these network interfaces as `NON_COMPLIANT` . You can view the resources in AWS Config by following these steps: a. Open the [AWS Config console](#). b. On the **Rules** page, choose your rule. c. On the **Rule detail** page, go to the bottom of the page. d. Under **Resources in scope**, select **NonCompliant**. Confirm that you see the IDs of	General AWS

Task	Description	Skills required
	the network interfaces that were created. e. For more detail about the network interface configuration, choose the resource ID. 3. Confirm that you receive the notification at the endpoint that you configured in Amazon SNS.	
Create a resource that is not applicable.	1. In a private subnet, create any resource that requires an elastic network interface. 2. After the resource is created, the AWS Config custom rule evaluates the elastic network interfaces associated with resource. It marks these network interfaces as `NOT_APPLICABLE` . These resources are not shown in the AWS Config console.	General AWS

Related resources

AWS documentation

- Setting up AWS Config
- AWS Config custom rules
- ABAC for AWS
- Deploy preventative attribute-based access controls for public subnets

Other AWS resources

- [Automate configuration compliance at scale in AWS]
- [Distributed Inspection Architectures with Gateway Load Balancer]

Additional information

The following is a sample Lambda function that is provided for demonstration purposes.

```
import boto3
import json
import os

# Init clients
config_client = boto3.client('config')
ec2_client = boto3.client('ec2')

def lambda_handler(event, context):

    # Init values
    compliance_value = 'NOT_APPLICABLE'
    invoking_event = json.loads(event['invokingEvent'])
    configuration_item = invoking_event['configurationItem']

    status = configuration_item['configurationItemStatus']
    eventLeftScope = event['eventLeftScope']

    # First check if the event configuration applies. Ex. resource event is not delete
    if (status == 'OK' or status == 'ResourceDiscovered') and not eventLeftScope:
        compliance_value = evaluate_change_notification_compliance(configuration_item)


    config_client.put_evaluations(
       Evaluations=[
           {
               'ComplianceResourceType': invoking_event['configurationItem']
['resourceType'],
               'ComplianceResourceId': invoking_event['configurationItem']
['resourceId'],
               'ComplianceType': compliance_value,
               'OrderingTimestamp': invoking_event['configurationItem']
['configurationItemCaptureTime']
```

```
            },
        ],
        ResultToken=event['resultToken'])

# Function with the logs to evaluate the resource
def evaluate_change_notification_compliance(configuration_item):
    is_in_scope = is_in_scope_subnet(configuration_item['configuration']['subnetId'])

    if (configuration_item['resourceType'] != 'AWS::EC2::NetworkInterface') or not
 is_in_scope:
        return 'NOT_APPLICABLE'

    else:
        alb_condition = configuration_item['configuration']['requesterId'] in ['amazon-
elb']
        nlb_condition = configuration_item['configuration']['interfaceType'] in
 ['network_load_balancer']
        nat_gateway_condition = configuration_item['configuration']['interfaceType'] in
 ['nat_gateway']

        if alb_condition or nlb_condition or nat_gateway_condition:
            return 'COMPLIANT'
    return 'NON_COMPLIANT'

# Function to check if elastic network interface is in public subnet
def is_in_scope_subnet(eni_subnet):

    subnet_description = ec2_client.describe_subnets(
        SubnetIds=[eni_subnet]
    )

    for subnet in subnet_description['Subnets']:
        for tag in subnet['Tags']:
            if tag['Key'] == os.environ.get('TAG_KEY') and tag['Value'] ==
 os.environ.get('TAG_VALUE'):
                return True

    return False
```

Deploy preventative attribute-based access controls for public subnets

Joel Alfredo Nunez Gonzalez and Samuel Ortega Sancho, Amazon Web Services

Summary

In centralized network architectures, inspection and edge virtual private clouds (VPCs) concentrate all inbound and outbound traffic, such as traffic to and from the internet. However, this can create bottlenecks or result in reaching the limits of AWS service quotas. Deploying network edge security alongside the workloads in their VPCs provides unprecedented scalability in comparison to the more common, centralized approach. This is called a *distributed edge* architecture.

Although deploying public subnets in workload accounts can provide benefits, it also introduces new security risks because it increases the attack surface. We recommend that you deploy only Elastic Load Balancing (ELB) resources, such as Application Load Balancers, or NAT gateways in the public subnets of these VPCs. Using load balancers and NAT gateways in dedicated public subnets helps you implement fine-grained control for inbound and outbound traffic.

Attribute-based access control (ABAC) is the practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS. ABAC can provide guardrails for public subnets in workload accounts. This helps application teams be agile, without compromising the security of the infrastructure.

This pattern describes how to help secure public subnets by implementing ABAC through a service control policy (SCP) in AWS Organizations and policies in AWS Identity and Access Management (IAM). You apply the SCP to either a member account of an organization or to an organizational unit (OU). These ABAC policies permit users to deploy NAT gateways in the target subnets and prevent them from deploying other Amazon Elastic Compute Cloud (Amazon EC2) resources, such as EC2 instances and elastic network interfaces.

Prerequisites and limitations

Prerequisites

- An organization in AWS Organizations
- Administrative access to the AWS Organizations root account

- In the organization, an active member account or OU for testing the SCP

Limitations

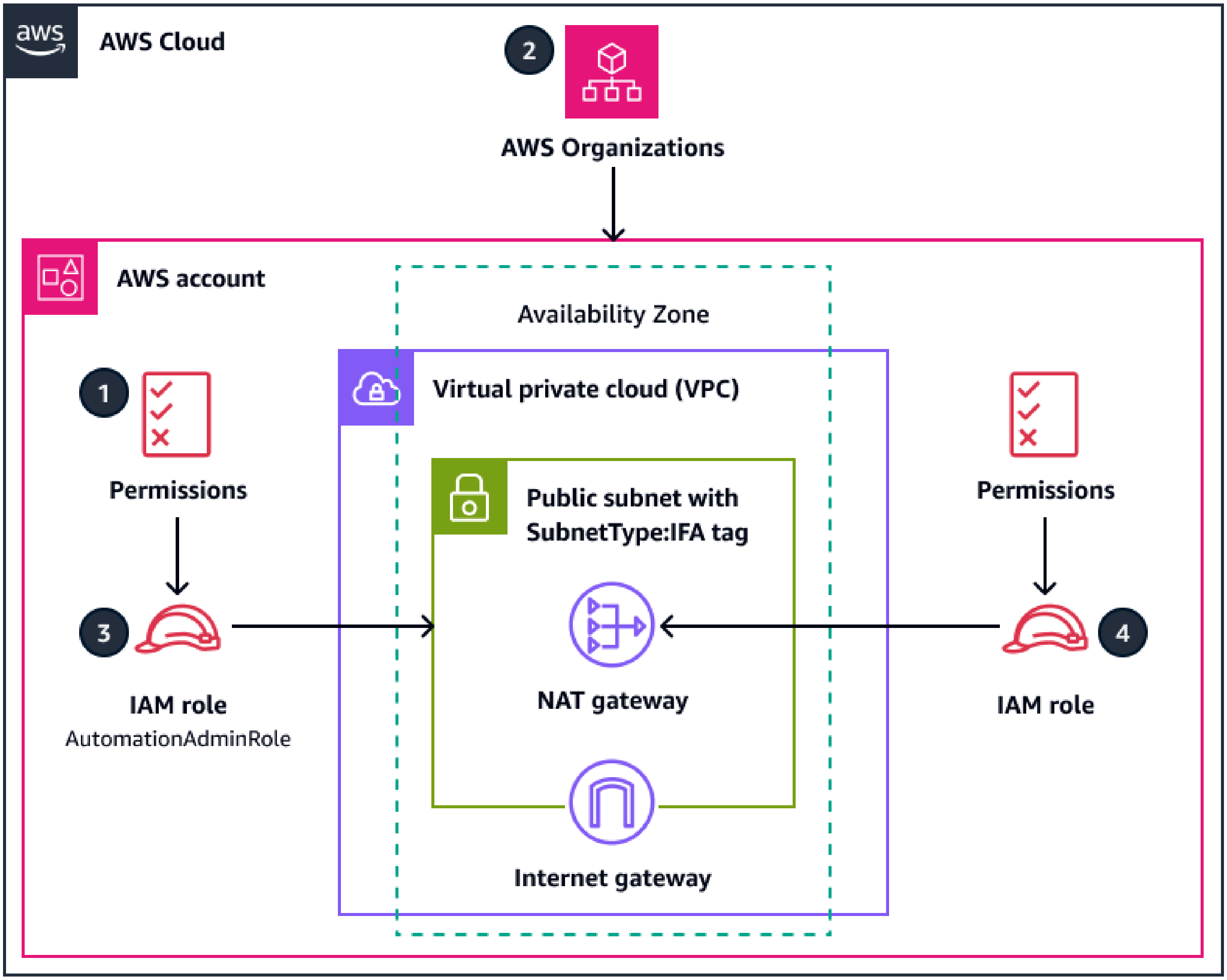
- The SCP in this solution doesn't prevent AWS services that use a service-linked role from deploying resources in the target subnets. Examples of these services are Elastic Load Balancing (ELB), Amazon Elastic Container Service (Amazon ECS), and Amazon Relational Database Service (Amazon RDS). For more information, see [SCP effects on permissions](#) in the AWS Organizations documentation. Implement security controls to detect these exceptions.

Architecture

Target technology stack

- SCP applied to an AWS account or OU in AWS Organizations
- The following IAM roles:
  - `AutomationAdminRole` – Used to modify subnet tags and create VPC resources after implementing the SCP
  - `TestAdminRole` – Used to test whether the SCP is preventing other IAM principals, including those with administrative access, from performing the actions reserved for the `AutomationAdminRole`

Target architecture



1. You create the `AutomationAdminRole` IAM role in the target account. This role has permissions to manage networking resources. Note the following permissions that are exclusive to this role:
   - This role can create VPCs and public subnets.
   - This role can modify the tag assignments for the target subnets.
   - This role can manage its own permissions.
2. In AWS Organizations, you apply the SCP to the target AWS account or OU. For a sample policy, see [Additional information](#) in this pattern.
3. A user or a tool in the CI/CD pipeline can assume the `AutomationAdminRole` role to apply the `SubnetType` tag to the target subnets.

4. By assuming other IAM roles, authorized IAM principals in your organization can manage NAT gateways in the target subnets and other permitted networking resources in the AWS account, such as route tables. Use IAM policies to grant these permissions. For more information, see Identity and access management for Amazon VPC.

Automation and scale

To help protect public subnets, the corresponding AWS tags must be applied. After applying the SCP, NAT gateways are the only kind of Amazon EC2 resource that authorized users can create in subnets that have the `SubnetType:IFA` tag. (`IFA` means *internet-facing assets*.) The SCP prevents the creation of other Amazon EC2 resources, such as instances and elastic network interfaces. We recommend that you use a CI/CD pipeline that assumes the `AutomationAdminRole` role to create VPC resources so that these tags are properly applied to public subnets.

Tools

AWS services

- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage. In AWS Organizations, you can implement service control policies (SCPs), which are a type of policy that you can use to manage permissions in your organization.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Epics

Apply the SCP

Task	Description	Skills required
Create a test admin role.	Create an IAM role named `TestAdminRole` in the target AWS account. Attach	AWS administrator

Task	Description	Skills required
	the **AdministratorAccess** AWS managed IAM policy to the new role. For instructions, see [Creating a role to delegate permissions to an IAM user](#) in the IAM documentation.	

Task	Description	Skills required
Create the automation admin role.	1. Create an IAM role named `AutomationAdminRole` in the target AWS account. 2. Attach the **AdministratorAccess** AWS managed IAM policy to the new role. The following is an example of a trust policy that you could use to test the role from the 111122223333 account. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal": { "AWS": [ "arn:aws:iam::111122223333:root" ] }, "Action": "sts:AssumeRole", "Condition": {} } ] }`	AWS administrator

Task	Description	Skills required
Create and attach the SCP.	1. Using the sample code provided in the Additional information section, create a security control policy. For instructions, see Creating an SCP in the AWS Organizations documentation. 2. Attach the SCP to the target AWS account or OU. For instructions, see Attaching and detaching service control policies in the AWS Organizations documentation.	AWS administrator

Test the SCP

Task	Description	Skills required
Create a VPC or subnet.	1. Assume the `TestAdminRole` role in the target AWS account. 2. Try to create a VPC or a new public subnet in an existing VPC. For instructions, see Create a VPC, subnets, and other VPC resources in the Amazon VPC documentation. You shouldn't be able to create these resources.	AWS administrator

Task	Description	Skills required
	3. Assume the `AutomationAdminRole` role, and retry the previous step. Now, you should be able to create the networking resources.	
Manage tags.	1. Assume the `TestAdminRole` role in the target AWS account. 2. Add a `SubnetType:IFA` tag to an available public subnet. You should be able to add this tag. For instructions about how to add tags through the AWS Command Line Interface (AWS CLI), see create-tags in the *AWS CLI Command Reference*. 3. Without changing your credentials, attempt to modify the `SubnetType:IFA` tag assigned to this subnet. You shouldn't be able to modify this tag. 4. Assume the `AutomationAdminRole` role, and retry the previous steps. This role should be able to add and modify this tag.	AWS administrator

Task	Description	Skills required
Deploy resources in the target subnets.	1. Assume the `TestAdminRole` role. 2. For a public subnet that has the `SubnetType:IFA` tag, try to create an EC2 instance. For instructions, see [Launch an instance](#) in the Amazon EC2 documentation. In this subnet, you shouldn't be able to create, modify, or delete any Amazon EC2 resources except NAT gateways. 3. In the same subnet, create a NAT gateway. For instructions, see [Create a NAT gateway](#) in the Amazon VPC documentation. You should be able to create, modify, or delete NAT gateways in this subnet.	AWS administrator

Task	Description	Skills required
Manage the AutomationAdminRole role.	1. Assume the `TestAdminRole` role. 2. Try to modify the `AutomationAdminRole` role. For instructions, see [Modifying a role](#) in the IAM documentation. You shouldn't be able to modify this role. 3. Assume the `AutomationAdminRole` role, and retry the previous step. Now, you should be able to modify the role.	AWS administrator

Clean up

Task	Description	Skills required
Clean up deployed resources.	1. Detach the SCP from the AWS account or OU. For instructions, see [Detaching an SCP](#) in the AWS Organizations documentation. 2. Delete the SCP. For instructions, see [Deleting an SCP](#) (AWS Organizations documentation). 3. Delete the `AutomationAdminRole` role and the `TestAdminRole` role. For instructions, see	AWS administrator

Task	Description	Skills required
	Deleting roles in the IAM documentation. 4. Delete all networking resources, such as VPCs and subnets, that you created for this solution.	

Related resources

AWS documentation

- Attaching and detaching SCPs
- Creating, updating, and deleting SCPs
- Deploy detective attribute-based access controls for public subnets by using AWS Config
- Detective controls
- Service authorization reference
- Tagging AWS resources
- What is ABAC for AWS?

Additional AWS references

- Securing resource tags used for authorization using a Service Control Policy in AWS Organizations (AWS blog post)

Additional information

The following service control policy is an example that you can use to test this approach in your organization.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCActions",
```

```
      "Effect": "Deny",
      "Action": [
        "ec2:CreateVPC",
        "ec2:CreateRoute",
        "ec2:CreateSubnet",
        "ec2:CreateInternetGateway",
        "ec2:DeleteVPC",
        "ec2:DeleteRoute",
        "ec2:DeleteSubnet",
        "ec2:DeleteInternetGateway"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:*"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:PrincipalARN": ["arn:aws:iam::*:role/AutomationAdminRole"]
        }
      }
    },
    {
      "Sid": "AllowNATGWOnIFASubnet",
      "Effect": "Deny",
      "NotAction": [
        "ec2:CreateNatGateway",
        "ec2:DeleteNatGateway"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:subnet/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIfExists": {
          "aws:ResourceTag/SubnetType": "IFA"
        },
        "StringNotLike": {
          "aws:PrincipalARN": ["arn:aws:iam::*:role/AutomationAdminRole"]
        }
      }
    },
    {
      "Sid": "DenyChangesToAdminRole",
      "Effect": "Deny",
      "NotAction": [
        "iam:GetContextKeysForPrincipalPolicy",
```

```
        "iam:GetRole",
        "iam:GetRolePolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListInstanceProfilesForRole",
        "iam:ListRolePolicies",
        "iam:ListRoleTags"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AutomationAdminRole"
      ],
      "Condition": {
        "StringNotLike": {
          "aws:PrincipalARN": ["arn:aws:iam::*:role/AutomationAdminRole"]
        }
      }
    }
  ]
}
```

Detect Amazon RDS and Aurora database instances that have expiring CA certificates

Stephen DiCato and Eugene Shifer, Amazon Web Services

Summary

As a security best practice, it is recommended that you encrypt data in transit between application servers and relational databases. You can use SSL or TLS to encrypt a connection to a database (DB) instance or cluster. These protocols help provide confidentiality, integrity, and authenticity between an application and database. The database uses a server certificate, which is issued by a certificate authority (CA) and is used to perform server identity verification. SSL or TLS verifies the authenticity of the certificate by validating its digital signature and ensuring it is not expired.

In the AWS Management Console, Amazon Relational Database Service (Amazon RDS) and Amazon Aurora provide notifications about DB instances that require certificate updates. However, to check for these notifications, you must log into each AWS account and navigate to the service console in each AWS Region. This task becomes more complex if you need to assess certificate validity across many AWS accounts that are managed as an organization in AWS Organizations.

By provisioning the infrastructure as code (IaC) provided in this pattern, you can detect expiring CA certificates for all Amazon RDS and Aurora DB instances in your AWS account or AWS organization. The AWS CloudFormation template provisions an AWS Config rule, an AWS Lambda function, and the necessary permissions. You can deploy it into a single account as a stack, or you can deploy it across the entire AWS organization as a stack set.

Prerequisites and limitations

Prerequisites

- An active AWS account
- If you're deploying into a single AWS account:
  - Ensure that you have permissions to create CloudFormation stacks.
  - Enable AWS Config in the target account.
  - (Optional) Enable AWS Security Hub in the target account.
- If you're deploying into an AWS organization:

- Ensure that you have permissions to create CloudFormation stack sets.
- Enable Security Hub with AWS Organizations integration.
- Enable AWS Config in the accounts where you are deploying this solution.
- Designate an AWS account to be the delegated administrator for AWS Config and Security Hub.

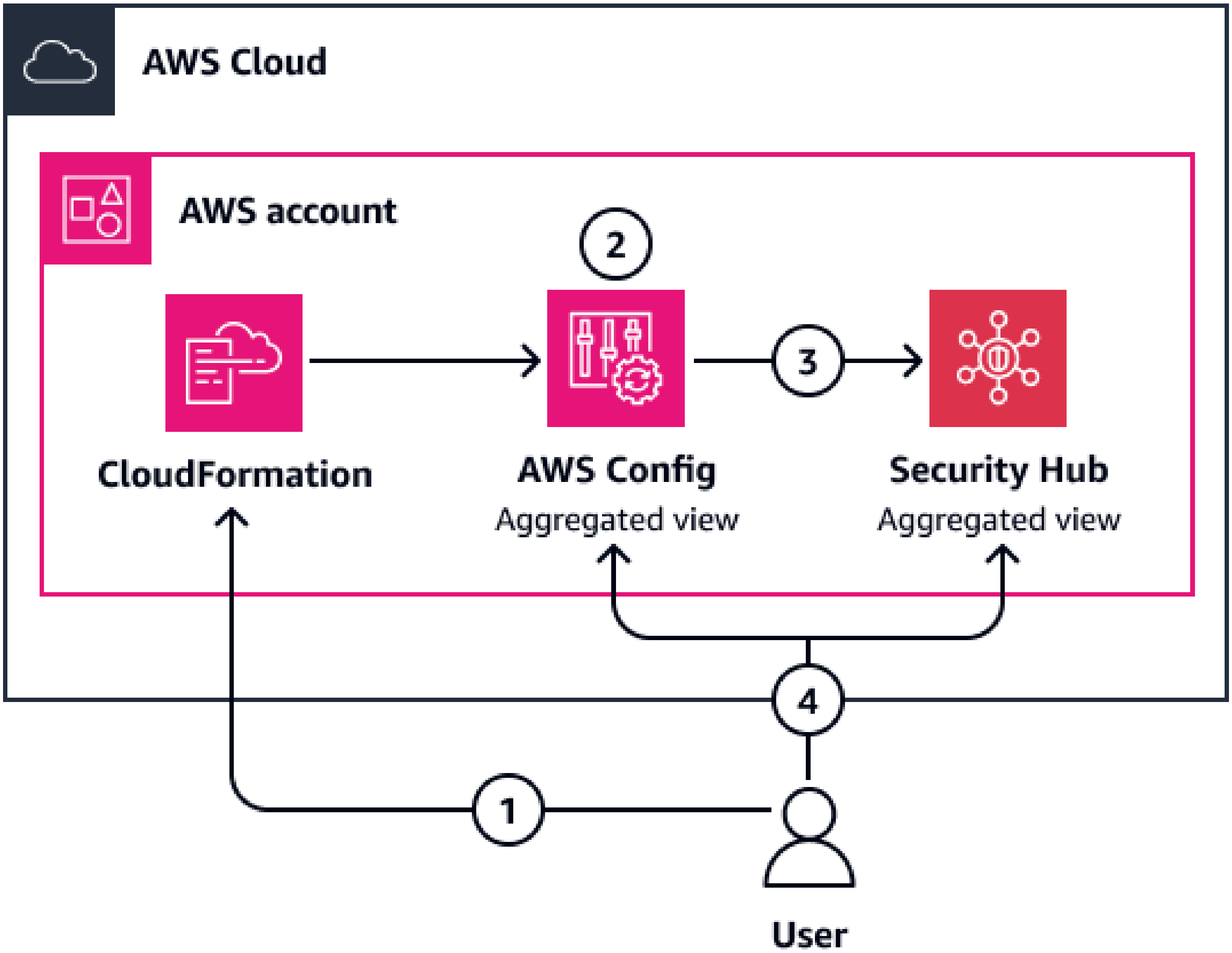
Limitations

- If you're deploying to an individual account that doesn't have Security Hub enabled, you can use AWS Config to evaluate the findings.
- If you're deploying to an organization that doesn't have a delegated administrator for AWS Config and Security Hub, you must log into the individual member accounts to view the findings.
- If you use AWS Control Tower to manage and govern the accounts in your organization, deploy the IaC in this pattern by using Customizations for AWS Control Tower (CfCT). Using the CloudFormation console will create configuration drift from AWS Control Tower guardrails and require that you re-enroll the organizational units (OUs) or managed accounts.
- Some AWS services aren't available in all AWS Regions. For Region availability, see the Service endpoints and quotas page, and choose the link for the service.

Architecture

Deploying into an individual AWS account

The following architecture diagram shows the deployment of the AWS resources within a single AWS account. It's implemented by using a CloudFormation template directly through the CloudFormation console. If Security Hub is enabled, you can view the results in either AWS Config or Security Hub. If Security Hub is not enabled, you can view the results only in the AWS Config console.

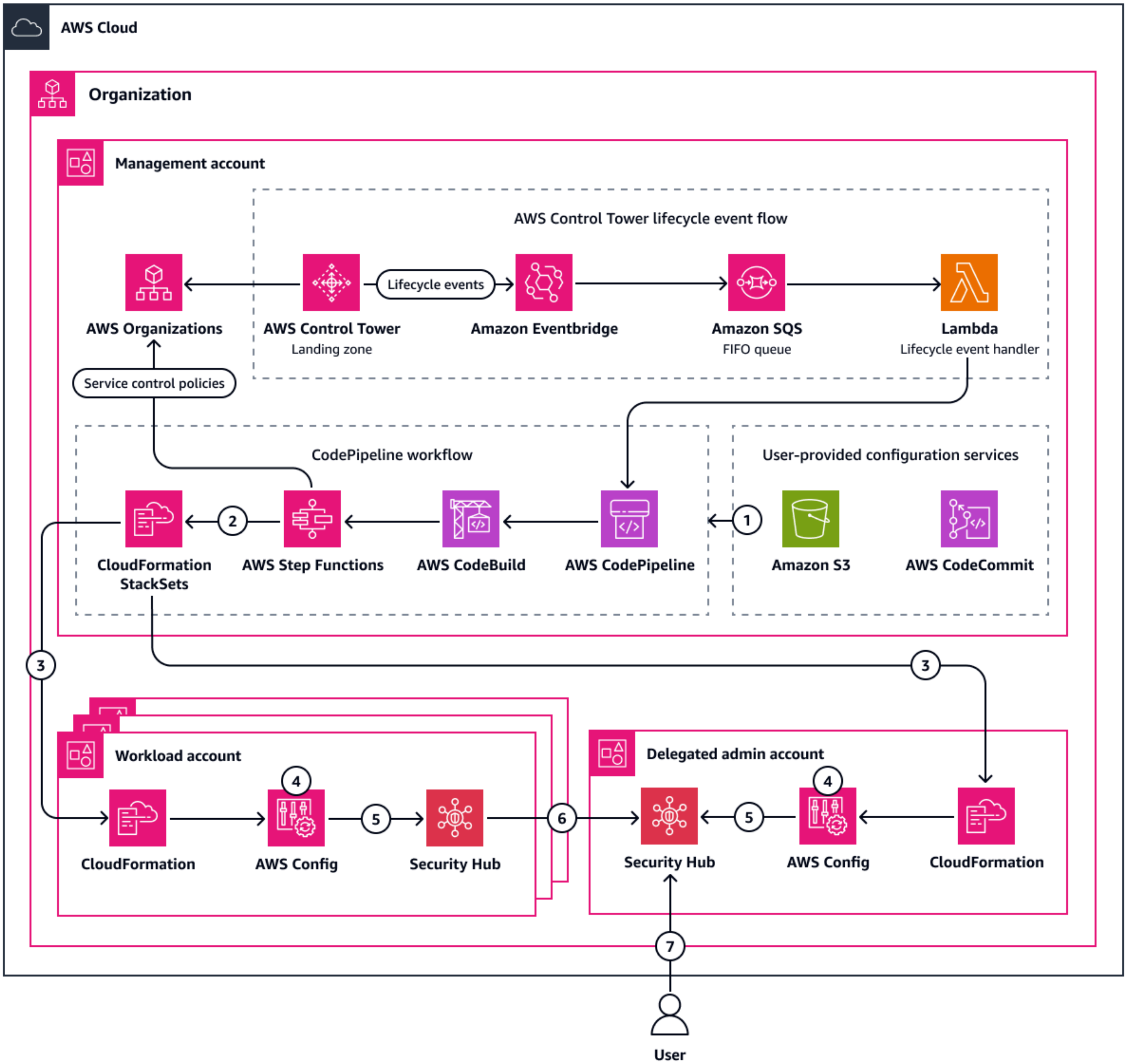


The diagram shows the following steps:

1. You create a CloudFormation stack. This deploys a Lambda function and an AWS Config rule. Both the rule and function are set up with the AWS Identity and Access Management (IAM) permissions required to publish resource evaluations in AWS Config and logs.
2. The AWS Config rule operates in [detective evaluation mode](#) and runs every 24 hours.
3. Security Hub receives all AWS Config findings.
4. You can view the findings in Security Hub or in AWS Config, depending on the account's configuration.

Deploying into an AWS organization

The following diagram shows the assessment of certificate expiration across multiple accounts that are managed through AWS Organizations and AWS Control Tower. You deploy the CloudFormation template through CfCT. The assessment outcomes are centralized in Security Hub in the delegated administrator account. The AWS CodePipeline workflow depicted in the diagram shows the background steps that occur during CfCT deployment.



The diagram shows the following steps:

1. Depending on the configuration for CfCT, in the management account, you push the IaC to an AWS CodeCommit repository or you upload a compressed (ZIP) file of the IaC to an Amazon Simple Storage Service (Amazon S3) bucket.
2. The CfCT pipeline unzips the file, runs cfn-nag (GitHub) checks, and deploys it as a CloudFormation stack set.
3. Depending on the configuration specified in the CfCT manifest file, CloudFormation StackSets deploys stacks into individual accounts or specified OUs. This deploys a Lambda function and an AWS Config rule in the target accounts. Both the rule and function are set up with the IAM permissions required to publish resource evaluations in AWS Config and logs.
4. The AWS Config rule operates in detective evaluation mode and runs every 24 hours.
5. AWS Config forwards all findings to Security Hub.
6. Security Hub findings are aggregated in the delegated administrator account.
7. You can view the findings in Security Hub in the delegated administrator account.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time. An AWS Config rule defines your ideal configuration settings for a resource, and AWS Config can evaluate whether your AWS resources comply with the conditions in your rules.
- AWS Control Tower helps you set up and govern an AWS multi-account environment, following prescriptive best practices. Customizations for AWS Control Tower (CfCT) helps you customize your AWS Control Tower landing zone and stay aligned with AWS best practices. Customizations are implemented with CloudFormation templates and service control policies (SCPs).
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.

- AWS Security Hub provides a comprehensive view of your security state in AWS. It also helps you check your AWS environment against security industry standards and best practices.

Other tools

- Python is a general-purpose computer programming language.

Code repository

The code for this pattern is available in the GitHub Detect Amazon RDS instances with expiring CA certificates repository.

Best practices

We recommend that you adhere to the best practices in the following resources:

- Best Practices for Organizational Units with AWS Organizations (AWS Cloud Operations & Migrations Blog)
- Guidance for Establishing an Initial Foundation using AWS Control Tower on AWS (AWS Solutions Library)
- Guidance for creating and modifying AWS Control Tower resources (AWS Control Tower documentation)
- CfCT deployment considerations (AWS Control Tower documentation)

Epics

Review the solution and code

Task	Description	Skills required
Determine your deployment strategy.	Review the solution and code to determine how you will deploy it into your AWS environment. Determine if you will be deploying into a single account or an AWS organization.	App owner, General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Clone the repository.</td><td>Enter the following command to clone the Detect Amazon RDS instances with expiring CA certificates repository.<pre>git clone https://github.com/aws-samples/config-rds-ca-expiry.git</pre></td><td>App developer, App owner</td></tr>
<tr><td>Validate the Python version.</td><td>1. Navigate into the top-level directory in the cloned repository.<pre>cd config-rds-ca-expiry</pre>2. Open config-rds-ca-expiry.yaml.
3. In the <code>CertExpirationCheckLambdaFunction</code> resource, confirm that the Python version is compatible with your target AWS Regions. By default, this function uses Python 3.12. For more information, see AWS Lambda adds support for Python 3.12. If necessary, update the Python version.
4. Save and close config-rds-ca-expiry.yaml.</td><td>App developer, App owner</td></tr>
</table>

Deploy the solution

Task	Description	Skills required
Deploy the CloudFormation template.	Deploy the CloudFormation template to your AWS environment. Do one of the following: • If you're deploying to a single AWS account, follow the instructions in Creating a stack. • If you're deploying to an organization that isn't managed by AWS Control Tower, follow the instructions in Create a stack set. • If you're deploying to an organization that is managed by AWS Control Tower, see the instructions in the Build your own customizations.	App developer, AWS administrator, General AWS
Verify the deployment.	In the CloudFormation console, verify that the stack or stack set has deployed successfully.	AWS administrator, App owner

Review the findings

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>View the AWS Config rule findings.</td><td>In Security Hub, do the following to view a list of individual findings:

1. Open the Security Hub console.
2. In the navigation pane, choose Findings.
3. In the Add filters box, add the following filters:
• Compliance Status is <code>FAILED</code>
• Title is <code>rds-has-expiring-ca</code>
4. Choose Apply.

In Security Hub, do the following to view a list of total findings grouped by AWS account:

1. Open the Security Hub console.
2. In the navigation pane, choose Insights.
3. Choose Create insight.
4. To select the grouping attribute for the insight:
a. Choose the search box to display the filter options.</td><td>AWS administrator, AWS systems administrator, Cloud administrator</td></tr>
</table>

Task	Description	Skills required
	b. Choose **Group by**. c. Select **AwsAccountId**. d. Choose **Apply**. 5. In the **Add filters** box, add the following filters: • **Title** is `rds-has-expiring-ca` • **Compliance Status** is `FAILED` 6. Choose **Create insight**. 7. Enter an **Insight name**, and then choose **Create insight**. In AWS Config, to view a list of findings, follow the instructions in [Viewing Compliance Information and Evaluation Results](#) in the AWS Config documentation.	

Troubleshooting

Issue	Solution
CloudFormation stack set creation or deletion fails	When AWS Control Tower is deployed, it enforces necessary guardrails and assumes control over AWS Config aggregators and rules. This includes preventing any direct alterations through CloudFormation. To properly deploy or remove this CloudForm

Issue	Solution
	ation template, including all associated resources, you must use CfCT.
CfCT fails to delete the CloudFormation template	If the CloudFormation template persists even after making necessary changes in the manifest file and removing the template files, confirm that the manifest file contains the `enable_stack_set_deletion` parameter and that the value is set to `false`. For more information, see Delete a stack set in the CfCT documentation.

Related resources

- Using SSL/TLS to encrypt a connection to a DB instance or cluster (Amazon RDS documentation)
- AWS Config Custom Rules (AWS Config documentation)

Dynamically generate an IAM policy with IAM Access Analyzer by using Step Functions

Thomas Scott, Koen van Blijderveen, Adil El Kanabi, and Rafal Pawlaszek, Amazon Web Services

Summary

Notice: AWS CodeCommit is no longer available to new customers. Existing customers of AWS CodeCommit can continue to use the service as normal. Learn more.

Least-privilege is the security best practice of granting the minimum permissions required to perform a task. Implementing least-privilege access in an already active Amazon Web Services (AWS) account can be challenging because you don't want to unintentionally block users from performing their job duties by changing their permissions. Before you can implement AWS Identity and Access Management (IAM) policy changes, you need to understand the actions and resources the account users are performing.

This pattern is designed to help you apply the principle of least-privilege access, without blocking or slowing down team productivity. It describes how to use IAM Access Analyzer and AWS Step Functions to dynamically generate an up-to-date IAM policy for your role, based on the actions that are currently being performed in the account. The new policy is designed to permit the current activity but remove any unnecessary, elevated privileges. You can customize the generated policy by defining allow and deny rules, and the solution integrates your custom rules.

This pattern includes options for implementing the solution with AWS Cloud Development Kit (AWS CDK) or HashiCorp CDK for Terraform (CDKTF). You can then associate the new policy to the role by using a continuous integration and continuous delivery (CI/CD) pipeline. If you have a multi-account architecture, you can deploy this solution in any account where you want to generate updated IAM policies for the roles, increasing the security of your entire AWS Cloud environment.

Prerequisites and limitations

Prerequisites

- An active AWS account with a AWS CloudTrail trail enabled.
- IAM permissions for the following:
  - Create and deploy Step Functions workflows. For more information, see Actions, resources, and condition keys for AWS Step Functions (Step Functions documentation).

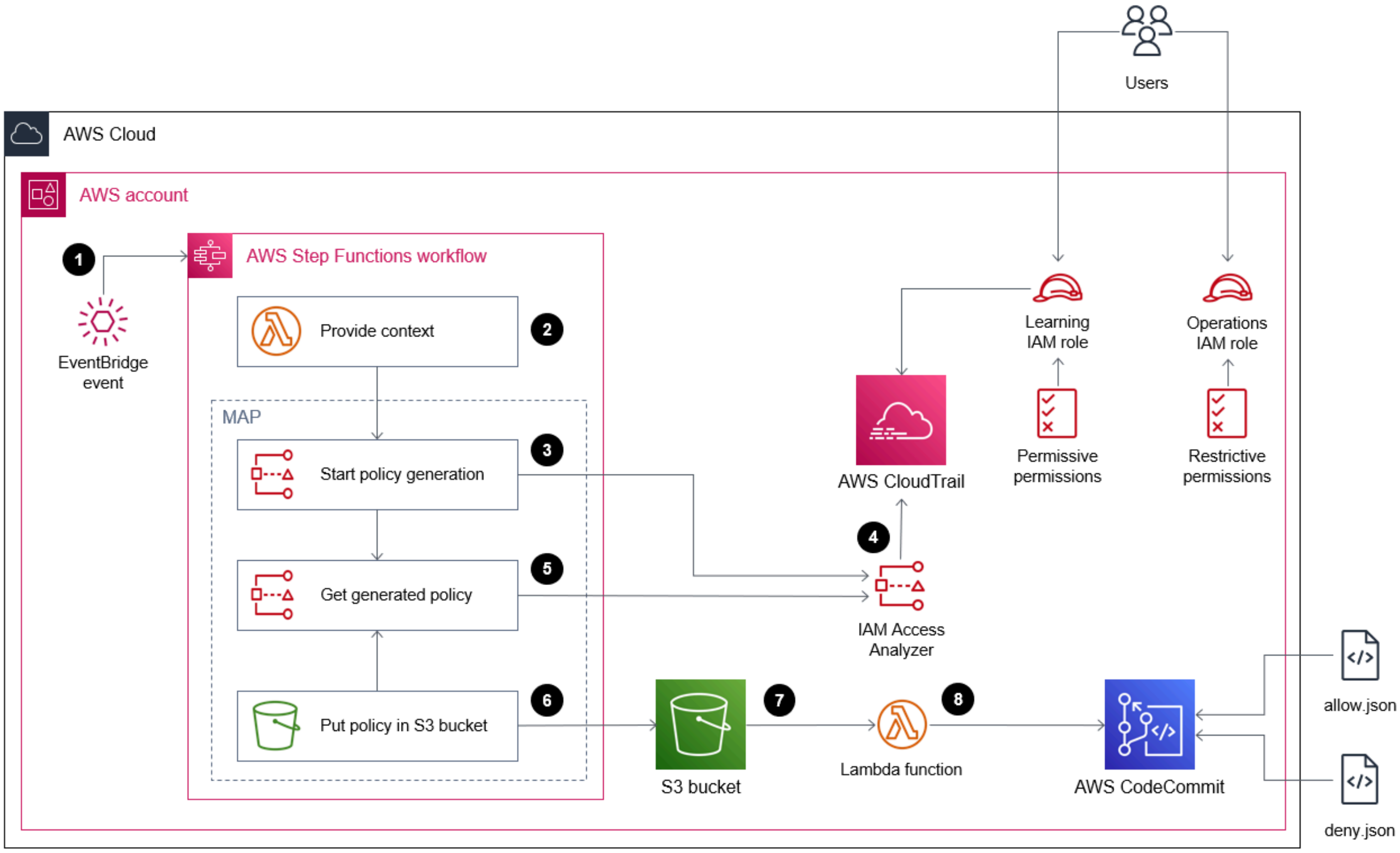
  - Create AWS Lambda functions. For more information, see Execution role and user permissions (Lambda documentation).
  - Create IAM roles. For more information, see Creating a role to delegate permissions to an IAM user (IAM documentation).
- npm installed. For more information, see Downloading and installing Node.js and npm (npm documentation).
- If you are deploying this solution with AWS CDK (Option 1):
  - AWS CDK Toolkit, installed and configured. For more information, see Install the AWS CDK (AWS CDK documentation).
- If you are deploying this solution with CDKTF (Option 2):
  - CDKTF, installed and configured. For more information, see Install CDK for Terraform (CDKTF documentation).
  - Terraform, installed and configured. For more information, see Get Started (Terraform documentation).
- AWS Command Line Interface (AWS CLI) locally installed and configured for your AWS account. For more information, see Installing or updating the latest version of the AWS CLI (AWS CLI documentation).

Limitations

- This pattern does not apply the new IAM policy to the role. At the end of this solution, the new IAM policy is stored in an AWS CodeCommit repository. You can use a CI/CD pipeline to apply policies to the roles in your account.

Architecture

Target architecture



1. A regularly scheduled Amazon EventBridge event rule starts a Step Functions workflow. You define this regeneration schedule as part of setting up this solution.
2. In the Step Functions workflow, a Lambda function generates the date ranges to use when analyzing account activity in the CloudTrail logs.
3. The next workflow step calls the IAM Access Analyzer API to start generating the policy.
4. Using the Amazon Resource Name (ARN) of the role you specify during set up, IAM Access Analyzer analyzes the CloudTrail logs for activity within the specified date rate. Based on the activity, IAM Access Analyzer generates an IAM policy that permits only the actions and services used by the role during the specified date range. When this step is complete, this step generates a job ID.
5. The next workflow step checks for the job ID every 30 seconds. When the job ID is detected, this step uses the job ID to call the IAM Access Analyzer API and retrieve the new IAM policy. IAM Access Analyzer returns the policy as a JSON file.
6. The next workflow step puts the **<IAM role name>/policy.json** file in an Amazon Simple Storage Service (Amazon S3) bucket. You define this S3 bucket as part of setting up this solution.

7. An Amazon S3 event notification starts a Lambda function.
8. The Lambda function retrieves the policy from the S3 bucket, integrates the custom rules you define in the **allow.json** and **deny.json** files, and then pushes the updated policy to CodeCommit. You define the CodeCommit repository, branch, and folder path as part of setting up this solution.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS CDK Toolkit is a command line cloud development kit that helps you interact with your AWS Cloud Development Kit (AWS CDK) app.
- AWS CloudTrail helps you audit the governance, compliance, and operational risk of your AWS account.
- AWS CodeCommit is a version control service that helps you privately store and manage Git repositories, without needing to manage your own source control system.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them. This pattern uses IAM Access Analyzer, a feature of IAM, to analyze your CloudTrail logs to identify actions and services that have been used by an IAM entity (user or role) and then generate an IAM policy that is based on that activity.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications. In this pattern, you use AWS SDK service integrations in Step Functions to call service API actions from your workflow.

Other tools

- CDK for Terraform (CDKTF) helps you define infrastructure as code (IaC) by using common programming languages, such as Python and Typescript.
- Lerna is a build system for managing and publishing multiple JavaScript or TypeScript packages from the same repository.
- Node.js is an event-driven JavaScript runtime environment designed for building scalable network applications.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.

Code repository

The code for this pattern is available in the GitHub Automated IAM Access Analyzer Role Policy Generator repository.

Epics

Prepare for deployment

Task	Description	Skills required
Clone the repo.	The following command clones the Automated IAM Access Analyzer Role Policy Generator (GitHub) repository. `git clone https://github.com/aws-samples/automated-iam-access-analyzer.git`	App developer
Install Lerna.	The following command installs Lerna. `npm i -g lerna`	App developer

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Set up the dependencies.</td><td>The following command installs the dependencies for the repository.
<pre>cd automated-iam-access-analyzer/
npm install && npm run bootstrap</pre></td><td>App developer</td></tr>
<tr><td>Build the code.</td><td>The following command tests, builds, and prepares the zip packages of the Lambda functions.
<pre>npm run test:code
npm run build:code
npm run pack:code</pre></td><td>App developer</td></tr>
<tr><td>Build the constructs.</td><td>The following command builds the infrastructure synthesizing applications, for both AWS CDK and CDKTF.
<pre>npm run build:infra</pre></td><td></td></tr>
<tr><td>Configure any custom permissions.</td><td>In the repo folder of the cloned repository, edit the allow.json and deny.json files to define any custom permissions for the role. If the allow.json and deny.json files contain the same permission, the deny permission is applied.</td><td>AWS administrator, App developer</td></tr>
</table>

Option 1 – Deploy the solution using AWS CDK

Task	Description	Skills required

Deploy the AWS CDK stack.	The following command deploys the infrastructure through AWS CloudFormation. Define the following parameters:
	• `<NAME_OF_ROLE>` – The ARN of the IAM role for which you are creating a new policy.
	• `<TRAIL_ARN>` – The ARN of the CloudTrail trail in which the role activity is stored.
	• `<CRON_EXPRESSION_TO_RUN_SOLUTION>` – The Cron expression that defines the regeneration schedule for the policy. The Step Functions workflow runs on this schedule.
	• `<TRAIL_LOOKBACK>` – The period, in days, to look back in the trail when evaluating the role permissions.
	`cd infra/cdk`
	`cdk deploy —-parameters roleArn=<NAME_OF_ROLE> \`
	`—-parameters trailArn=<TRAIL_ARN> \`
		App developer

Task	Description	Skills required
	`--parameters schedule=<CRON_EXPRESSION_TO_RUN_SOLUTION> \ [ --parameters trailLookBack=<TRAIL_LOOKBACK> ]` **Note** The square brackets denote optional parameters.	
(Optional) Wait for the new policy.	If the trail does not contain a reasonable amount of historical activity for the role, wait until you are confident that there is enough logged activity for IAM Access Analyzer to generate an accurate policy. If the role has been active in the account for a sufficient period of time, this waiting period might not be necessary.	AWS administrator
Manually review the generated policy.	In your CodeCommit repository, review the generated **<ROLE_ARN>.json** file to confirm that the allow and deny permissions are appropriate for the role.	AWS administrator

Option 2 – Deploy the solution using CDKTF

Task	Description	Skills required
Synthesize the Terraform template.	The following command synthesizes the Terraform template. `lerna exec cdktf synth --scope @aiaa/tfm`	App developer
Deploy the Terraform template.	The following command navigates to the directory that contains the CDKTF-defined infrastructure. `cd infra/cdktf` The following command deploys the infrastructure in the target AWS account. Define the following parameters: • `<account_ID>` – The ID of the target account. • `<region>` - The target AWS Region. • `<selected_role_ARN>` – The ARN of the IAM role for which you are creating a new policy. • `<trail_ARN>` – The ARN of the CloudTrail trail in which the role activity is stored.	App developer

Task	Description	Skills required
	• `<schedule_expression>` – The Cron expression that defines the regeneration schedule for the policy. The Step Functions workflow runs on this schedule. • `<trail_look_back>` – The period, in days, to look back in the trail when evaluating the role permissions. `TF_VAR_accountId=<account_ID> \ TF_VAR_region=<region> \ TF_VAR_roleArns=<selected_role_ARN> \ TF_VAR_trailArn=<trail_ARN> \ TF_VAR_schedule=<schedule_expression> \ [ TF_VAR_trailLookBack=<trail_look_back> ] \ cdktf deploy` **Note** The square brackets denote optional parameters.	

Task	Description	Skills required
(Optional) Wait for the new policy.	If the trail does not contain a reasonable amount of historical activity for the role, wait until you are confident that there is enough logged activity for IAM Access Analyzer to generate an accurate policy. If the role has been active in the account for a sufficient period of time, this waiting period might not be necessary.	AWS administrator
Manually review the generated policy.	In your CodeCommit repository, review the generated **<ROLE_ARN>.json** file to confirm that the allow and deny permissions are appropriate for the role.	AWS administrator

Related resources

AWS resources

- IAM Access Analyzer endpoints and quotas
- Configuring the AWS CLI
- Getting started with the AWS CDK
- Least-privilege permissions

Other resources

- CDK for Terraform (Terraform website)

Enable Amazon GuardDuty conditionally by using AWS CloudFormation templates

Ram Kandaswamy, Amazon Web Services

Summary

You can enable Amazon GuardDuty on an Amazon Web Services (AWS) account by using an AWS CloudFormation template. By default, if GuardDuty is already enabled when you try to use CloudFormation to turn it on, the stack deployment fails. However, you can use conditions in your CloudFormation template to check whether GuardDuty is already enabled. CloudFormation supports the use of conditions that compare static values; it does not support using the output of another resource property within the same template. For more information, see Conditions in the CloudFormation documentation.

In this pattern, you use a CloudFormation custom resource backed by an AWS Lambda function to conditionally enable GuardDuty if it is not already enabled. If GuardDuty is enabled, the stack captures the status and records it in the output section of the stack. If GuardDuty is not enabled, the stack enables it.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS Identity and Access Management (IAM) role that has permissions to create, update, and delete CloudFormation stacks
- AWS Command Line Interface (AWS CLI), installed and configured

Limitations

If GuardDuty has been manually disabled for an AWS account or AWS Region, this pattern does not enable GuardDuty for that target account or Region.

Architecture

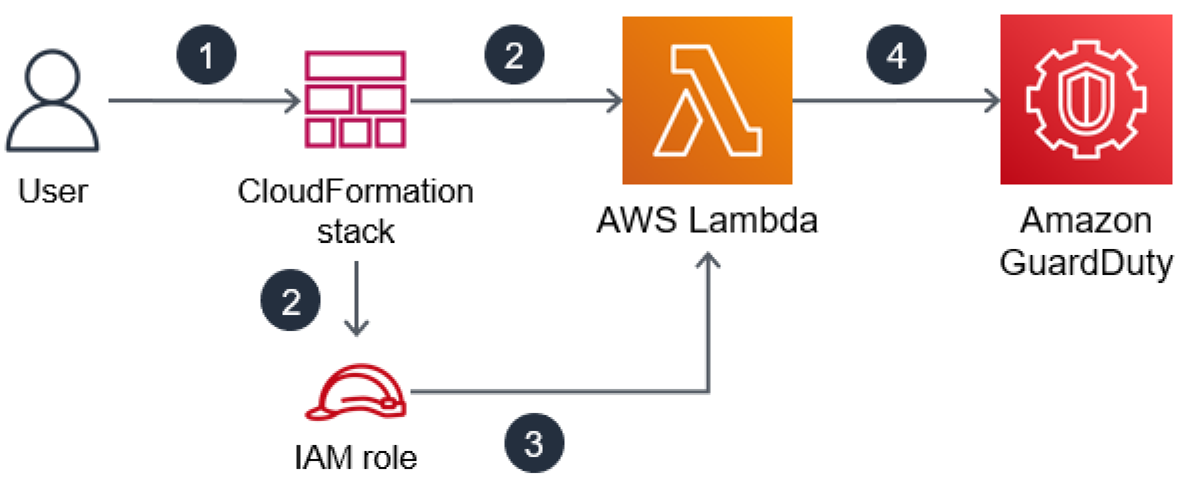
Target technology stack

The pattern uses CloudFormation for infrastructure as code (IaC). You use a CloudFormation custom resource backed by a Lambda function to achieve the dynamic service-enablement capability.

Target architecture

The following high-level architecture diagram shows the process of enabling GuardDuty by deploying a CloudFormation template:

1. You deploy a CloudFormation template to create a CloudFormation stack.
2. The stack creates an IAM role and a Lambda function.
3. The Lambda function assumes the IAM role.
4. If GuardDuty is not already enabled on the target AWS account, the Lambda function enables it.



Automation and scale

You can use the AWS CloudFormation StackSet feature to extend this solution to multiple AWS accounts and AWS Regions. For more information, see Working with AWS CloudFormation StackSets in the CloudFormation documentation.

Tools

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.

- [AWS CloudFormation](#) helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- [Amazon GuardDuty](#) is a continuous security monitoring service that analyzes and processes logs to identify unexpected and potentially unauthorized activity in your AWS environment.
- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Epics

Create the CloudFormation template and deploy the stack

Task	Description	Skills required
Create the CloudFormation template.	1. Copy the code in *CloudFormation template* in the Additional information section. 2. Paste the code in a text editor. 3. Save the file as `sample.yaml` on your workstation.	AWS DevOps
Create the CloudFormation stack.	1. In the AWS CLI, enter the following command. This creates a new CloudFormation stack using the `sample.yaml` file. For more information, see Creating a stack in the CloudFormation documentation.	AWS DevOps

Task	Description	Skills required

	`aws cloudformation create-stack \ --stack-name guardduty-cf-stack \ --template-body file://sample.yaml`
	2. Confirm the following value appears in the AWS CLI, indicating that the stack has been successfully created. The amount of time required to create the stack can vary.
	`"StackStatus": "CREATE_COMPLETE",`	

Validate that GuardDuty is enabled for the AWS account.	1. Sign in to the AWS Management Console and open the [GuardDuty console](#). 2. Verify that the GuardDuty service is enabled.	Cloud administrator, AWS administrator
Configure additional accounts or Regions.	As needed for your use case, use the CloudFormation StackSet feature to extend this solution to multiple AWS accounts and AWS Regions. For more information, see [Working with AWS CloudFormation StackSets](#) in the CloudFormation documentation.	Cloud administrator, AWS administrator

Related resources

References

- AWS CloudFormation documentation
- AWS Lambda resource type reference
- CloudFormation resource type: AWS::IAM::Role
- CloudFormation resource type: AWS::GuardDuty::Detector
- Four ways to retrieve any AWS service property using AWS CloudFormation (blog post)

Tutorials and videos

- Simplify Your Infrastructure Management Using AWS CloudFormation (Tutorial)
- Use Amazon GuardDuty and AWS Security Hub to secure multiple accounts (AWS re:Invent 2020)
- Best practices for authoring AWS CloudFormation (AWS re:Invent 2019)
- Threat Detection on AWS: An Introduction to Amazon GuardDuty (AWS re:Inforce 2019)

Additional information

CloudFormation template

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  rLambdaLogGroup:
    Type: 'AWS::Logs::LogGroup'
    DeletionPolicy: Delete
    Properties:
      RetentionInDays: 7
      LogGroupName: /aws/lambda/resource-checker
  rLambdaCheckerLambdaRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !Sub 'resource-checker-lambda-role-${AWS::Region}'
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
```

```
              Service: lambda.amazonaws.com
            Action: 'sts:AssumeRole'
      Path: /
      Policies:
        - PolicyName: !Sub 'resource-checker-lambda-policy-${AWS::Region}'
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Sid: CreateLogGroup
                Effect: Allow
                Action:
                  - 'logs:CreateLogGroup'
                  - 'logs:CreateLogStream'
                  - 'logs:PutLogEvents'
                  - 'iam:CreateServiceLinkedRole'
                  - 'cloudformation:CreateStack'
                  - 'cloudformation:DeleteStack'
                  - 'cloudformation:Desc*'
                  - 'guardduty:CreateDetector'
                  - 'guardduty:ListDetectors'
                  - 'guardduty:DeleteDetector'
                Resource: '*'
  resourceCheckerLambda:
    Type: 'AWS::Lambda::Function'
    Properties:
      Description: Checks for resource type enabled and possibly name to exist
      FunctionName: resource-checker
      Handler: index.lambda_handler
      Role: !GetAtt
        - rLambdaCheckerLambdaRole
        - Arn
      Runtime: python3.13
      MemorySize: 128
      Timeout: 180
      Code:
        ZipFile: |
          import boto3
          import os
          import json
          from botocore.exceptions import ClientError
          import cfnresponse


          guardduty=boto3.client('guardduty')
```

```
          cfn=boto3.client('cloudformation')


          def lambda_handler(event, context):
              print('Event: ', event)
              if 'RequestType' in event:
                if event['RequestType'] in ["Create","Update"]:
                    enabled=False
                    try:
                      response=guardduty.list_detectors()
                      if "DetectorIds" in response and len(response["DetectorIds"])>0:
                        enabled="AlreadyEnabled"
                      elif "DetectorIds" in response and
 len(response["DetectorIds"])==0:
                        cfn_response=cfn.create_stack(
                          StackName='guardduty-cfn-stack',
                          TemplateBody='{ "AWSTemplateFormatVersion": "2010-09-09",
 "Description": "A sample template",    "Resources": { "IRWorkshopGuardDutyDetector": {
  "Type": "AWS::GuardDuty::Detector",    "Properties": {   "Enable": true  }   } } }'
                          )
                        enabled="True"
                    except Exception as e:
                        print("Exception: ",e)
                    responseData = {}
                    responseData['status'] = enabled
                    cfnresponse.send(event, context, cfnresponse.SUCCESS, responseData,
 "CustomResourcePhysicalID" )
                elif event['RequestType'] == "Delete":
                    cfn_response=cfn.delete_stack(
                            StackName='guardduty-cfn-stack')
                    cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
  CheckResourceExist:
    Type: 'Custom::LambdaCustomResource'
    Properties:
      ServiceToken: !GetAtt
        - resourceCheckerLambda
        - Arn
Outputs:
  status:
    Value: !GetAtt
      - CheckResourceExist
      - status
```

Alternative code option for the Lambda resource

The provided CloudFormation template uses inline code to reference the Lambda resource, for easier reference and guidance. Alternatively, you can place the Lambda code in an Amazon Simple Storage Service (Amazon S3) bucket and reference it in the CloudFormation template. Inline code doesn't support package dependencies or libraries. You can support these by placing the Lambda code in an Amazon S3 bucket and referencing it in the CloudFormation template.

Replace the following lines of code:

```
Code:
         ZipFile: |
```

with the following lines of code:

```
Code:
      S3Bucket: <bucket name>
      S3Key: <python file name>
      S3ObjectVersion: <version>
```

The `S3ObjectVersion` property can be omitted if you are not using versioning in your Amazon S3 bucket. For more information, see [Using versioning in Amazon S3 buckets] in the Amazon S3 documentation.

Enable transparent data encryption in Amazon RDS for SQL Server

Ranga Cherukuri, Amazon Web Services

Summary

This pattern describes how to implement transparent data encryption (TDE) in Amazon Relational Database Service (Amazon RDS) for SQL Server to encrypt data at rest.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An Amazon RDS for SQL Server DB instance

Product versions

Amazon RDS currently supports TDE for the following SQL Server versions and editions:

- SQL Server 2016 Enterprise Edition
- SQL Server 2017 Enterprise Edition
- SQL Server 2019 Standard and Enterprise Editions
- SQL Server 2022 Standard and Enterprise Editions

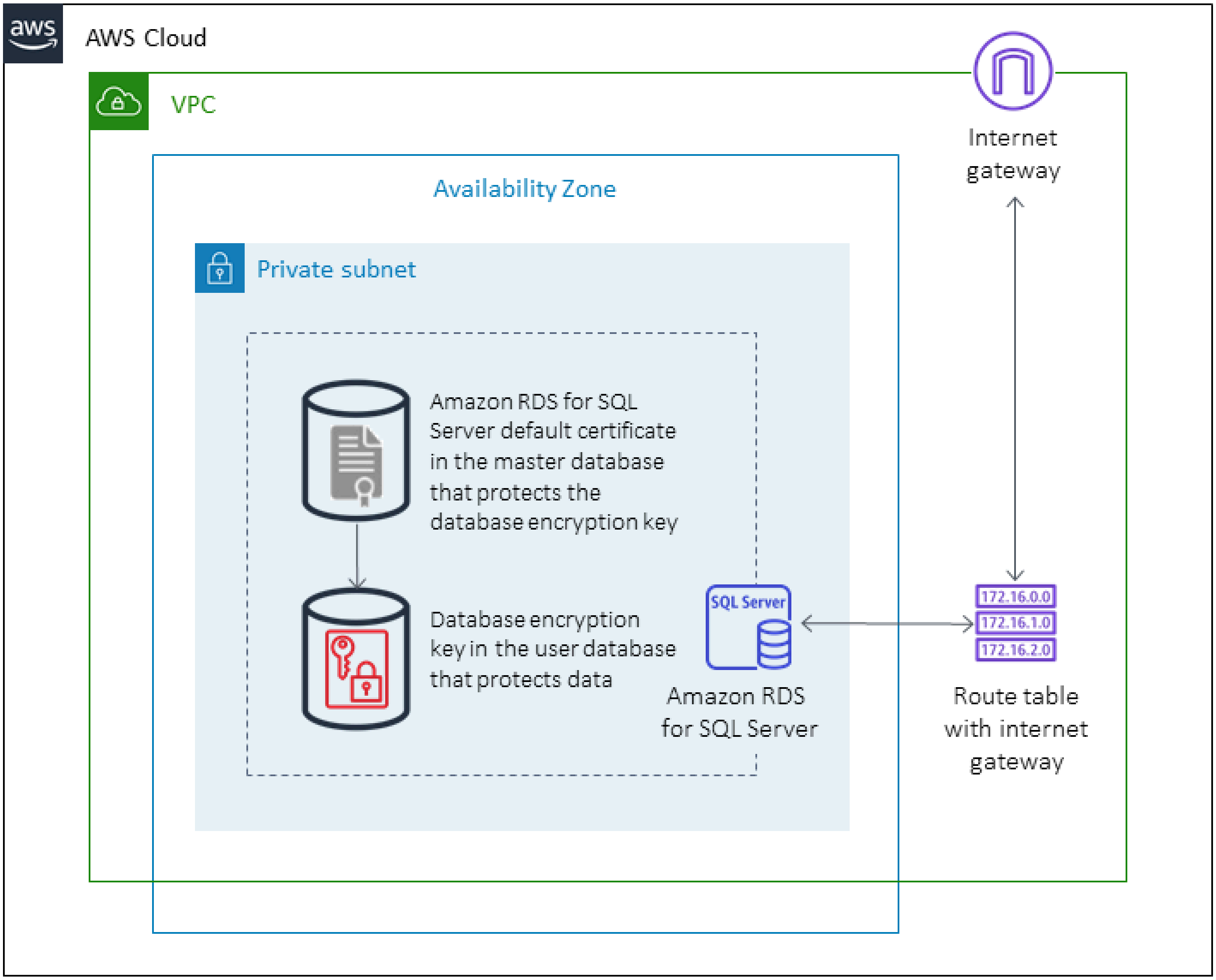
For the latest information about supported versions and editions, see Support for Transparent Data Encryption in SQL Server in the Amazon RDS documentation.

Architecture

Technology stack

- Amazon RDS for SQL Server

Architecture



Tools

- Microsoft SQL Server Management Studio (SSMS) is an integrated environment for managing a SQL Server infrastructure. It provides a user interface and a group of tools with rich script editors that interact with SQL Server.

Epics

Create an option group in the Amazon RDS console

Task	Description	Skills required
Open the Amazon RDS console.	Sign in to the AWS Management Console and open the [Amazon RDS console](https://console.aws.amazon.com/rds/).	Developer, DBA
Create an option group.	In the navigation pane, choose **Option groups**, **Create group**. Choose **sqlserver-ee** as the DB engine, and then select the engine version.	Developer, DBA
Add the TRANSPARENT_DATA_ENCRYPTION option.	Edit the option group you created and add the option called `TRANSPARENT_DATA_ENCRYPTION` .	Developer, DBA

Associate the option group with the DB instance

Task	Description	Skills required
Choose the DB instance.	In the Amazon RDS console, in the navigation pane, choose **Databases**, and then choose the DB instance you want to associate with the option group.	Developer, DBA
Associate the DB instance with the option group.	Choose **Modify**, and then use the **Option group** setting to associate the SQL Server	Developer, DBA

Task	Description	Skills required
	DB instance with the option group you created earlier.	
Apply the changes.	Apply the changes immediately or during the next maintenance window, as desired.	Developer, DBA

Get the certificate name.	Get the default certificate name by using the following query.
	`USE [master]`
	`GO`
	`SELECT name FROM sys.certificates WHERE name LIKE 'RDSTDECertificate%'`
	`GO`
		Developer, DBA

Create the database encryption key

Task	Description	Skills required
Connect to the Amazon RDS for SQL Server DB instance using SSMS.	For instructions, see Using SSMS in the Microsoft documentation.	Developer, DBA
Create the database encryption key by using the default certificate.	Create a database encryption key by using the default certificate name you got earlier. Use the following T-SQL query to create a database encryption key. You can specify the	Developer, DBA

Task	Description	Skills required

	AES_256 algorithm instead of AES_128.
	`USE [Databasename]`
	`GO`
	`CREATE DATABASE ENCRYPTION KEY`
	`WITH ALGORITHM = AES_128`
	`ENCRYPTION BY SERVER CERTIFICATE [certificatename]`
	`GO`	

Enable the encryption on the database.	Use the following T-SQL query to enable database encryption.
	`ALTER DATABASE [Database Name]`
	`SET ENCRYPTION ON`
	`GO`
		Developer, DBA

Check the status of encryption.	Use the following T-SQL query to check the status of encryption. `SELECT DB_NAME(database_id) AS DatabaseName, encryption_state, percent_complete FROM sys.dm_database_encryption_keys`	Developer, DBA

Related resources

- Support for Transparent Data Encryption in SQL Server (Amazon RDS documentation)

- Working with Option Groups (Amazon RDS documentation)
- Modifying an Amazon RDS DB Instance (Amazon RDS documentation)
- Transparent Data Encryption for SQL Server (Microsoft documentation)
- Using SSMS (Microsoft documentation)

Ensure AWS load balancers use secure listener protocols (HTTPS, SSL/TLS)

Chandini Penmetsa and Purushotham G K, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, containers, IP addresses, and AWS Lambda functions. The load balancers use listeners to define the ports and protocols that the load balancer uses to accept traffic from users. Application Load Balancers make routing decisions at the application layer and use the HTTP/HTTPS protocols. Network Load Balancers make routing decisions at the transport layer and use the Transmission Control Protocol (TCP), Transport Layer Security (TLS), User Datagram Protcol (UDP), or TCP_UDP protocols. Classic Load Balancers make routing decisions at either the transport layer, using TCP or Secure Sockets Layer (SSL) protocols, or at the application layer, using HTTP/HTTPS.

Your organization might have a security or compliance requirement that load balancers accept traffic from users only on secure protocols, such as HTTPS or SSL/TLS.

This pattern provides a security control that uses an Amazon EventBridge rule to monitor the `CreateListener` and `ModifyListener` API calls for Application Load Balancers and Network Load Balancers, and the `CreateLoadBalancerListeners` and `CreateLoadBalancer` API calls for Classic Load Balancers. If HTTP, TCP/UDP, or TCP_UDP is used for the load balancer's listener protocol, the control invokes a Lambda function. The Lambda function publishes a message to an Amazon Simple Notification Service (Amazon SNS) topic to send a notification that contains the load balancer details.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An email address where you want to receive the violation notification
- An Amazon Simple Storage Service (Amazon S3) bucket to store the Lambda code .zip file

Limitations

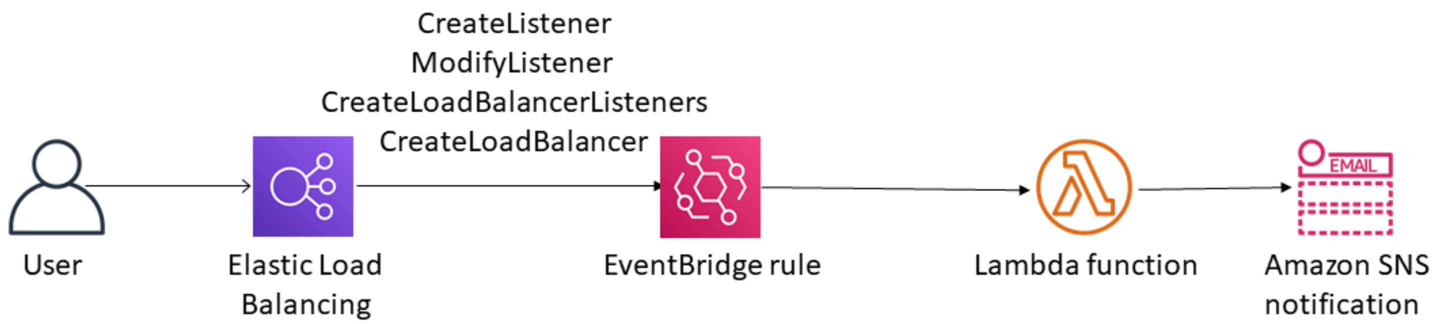
- This security control does not check for existing load balancers unless an update is made to the load balancer listeners.
- This security control is regional and must be deployed in the AWS Regions that you intend to monitor.

Architecture

Target technology stack

- Lambda function
- Amazon SNS topic
- EventBridge rule

Target architecture



Automation and scale

- If you are using AWS Organizations, you can use [AWS Cloudformation StackSets](#) to deploy this template in multiple accounts that you want this to monitor.

Tools

- [AWS CloudFormation](#) – AWS CloudFormation is a service that helps you model and set up AWS resources by using infrastructure as code.
- [Amazon EventBridge](#) – Amazon EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services, routing that data to targets such as Lambda functions.
- [AWS Lambda](#) – Lambda supports running code without provisioning or managing servers.

- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Best practices

Ensure that the SNS topic used isn't publicly accessible. For more information, see the AWS documentation.

Epics

Upload the Lambda code

Task	Description	Skills required
Define the S3 bucket.	On the Amazon S3 console, choose or create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket needs to be in the same Region as the load balancer that is being evaluated.	Cloud Architect
Upload the Lambda code to the S3 bucket.	Upload the Lambda code .zip file that's provided in the "Attachments" section to the defined S3 bucket.	Cloud Architect

Task	Description	Skills required
Deploy the AWS CloudFormation template.	On the AWS CloudFormation console, in the same AWS Region as the S3 bucket, deploy the template that is provided in the "Attachments" section. In the next epic, provide the values for the parameters.	Cloud Architect

CloudFormation parameters

Task	Description	Skills required
Name the S3 bucket.	Enter the name of the S3 bucket that you created in the first epic.	Cloud Architect
Provide the Amazon S3 prefix.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `<directory>/<file-name>.zip`).	Cloud Architect
Provide the SNS topic ARN.	Provide the SNS topic Amazon Resource Name (ARN) if you want to use an existing SNS topic for violation notifications. To create a new SNS topic, keep the value as None (the default value).	Cloud Architect

Task	Description	Skills required
Provide an email address.	Provide an active email address to receive Amazon SNS notifications.	Cloud Architect
Define the logging level.	Define the logging level and frequency for your Lambda function. `Info` designates detailed informational messages on the application's progress. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	Cloud Architect

Deploy the CloudFormation template

Task	Description	Skills required
Download the template.	Download the CloudFormation template that's provided in the *Attachments* section.	Cloud architect
Create the stack.	In the same Region as the S3 bucket, navigate to the CloudFormation service console, and deploy the downloaded template. Refer to the previous epic for parameter details.	Cloud architect
Verify the resources.	After the stack is created completely, navigate to the	Cloud architect

Task	Description	Skills required
	Resources tab, and verify the resources. The template will create the following resources : • EventBridge rule • Lambda function • Lambda execution role • Lambda invoke permission	

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the template successfu lly deploys, if a new SNS topic was created, a subscription email message is sent to the email address provided in the parameters. You must confirm this email subscription to receive violation notifications.	Cloud architect

Troubleshooting

Issue	Solution
Stack creation failed. Error occurred while GetObject. S3 Error Code: PermanentRedirect. S3 Error Message: The bucket is in this region: xx-xxxx-1. Please use this region to retry the request.	Make sure that the S3 bucket Region and the Region where the stack is being deployed are the same.

Issue	Solution
Stack creation failed. The runtime parameter of python3.6 is no longer supported for creating or updating AWS Lambda functions.	Update the downloaded template at line 186 from Python version 3.6 to 3.9.

Related resources

- Creating a stack on the AWS CloudFormation console
- AWS Lambda
- What is a Classic Load Balancer?
- What is an Application Load Balancer?
- What is a Network Load Balancer?
- Best practices for working with AWS Lambda functions
- AWS CloudFormation best practices

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Ensure encryption for Amazon EMR data at rest is enabled at launch

Priyanka Chaudhary, Amazon Web Services

Summary

This pattern provides a security control for monitoring the encryption of Amazon EMR clusters on Amazon Web Services (AWS).

Data encryption helps prevent unauthorized users from reading data on a cluster and associated data storage systems. This includes data that may be intercepted as it travels the network, known as data in transit, and data that is saved to persistent media, known as data at-rest. Data at rest in Amazon Simple Storage Service (Amazon S3) can be encrypted in two ways.

- Server-side encryption with Amazon S3–managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), set up with policies that are suitable for Amazon EMR.

This security control monitors for API calls and initiates an Amazon CloudWatch Events event on RunJobFlow. The trigger invokes AWS Lambda, which runs a Python script. The function retrieves the EMR cluster ID from the event JSON input and determines whether there is a security violation by performing the following checks.

1. Check if an EMR cluster is associated with an Amazon EMR specific security configuration.
2. If an Amazon EMR specific security configuration is associated with the EMR cluster, check if Encryption-at-Rest is turned on.
3. If Encryption-at-Rest is not turned on, send an Amazon Simple Notification Service (Amazon SNS) notification that includes the EMR cluster name, violation details, AWS Region, AWS account, and the Lambda Amazon Resource Name (ARN) that this notification is sourced from.

Prerequisites and limitations

Prerequisites

- An active AWS account

- An S3 bucket for the Lambda code .zip file
- An email address where you want to receive the violation notification
- Amazon EMR logging turned off so that all the API logs can be retrieved

Limitations

- This detective control is regional and must be deployed in the AWS Regions you intend to monitor.

Product versions

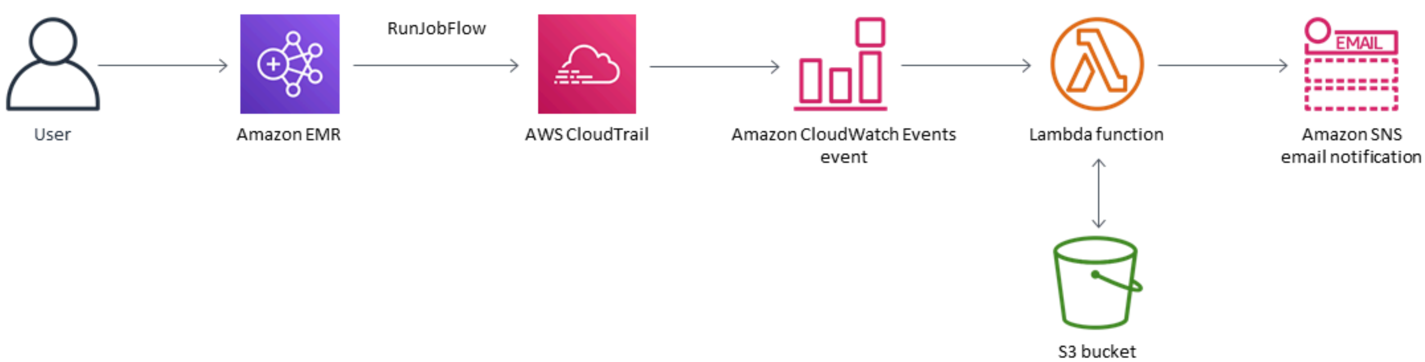
- Amazon EMR release 4.8.0 and above

Architecture

Target technology stack

- Amazon EMR
- Amazon CloudWatch Events event
- Lambda function
- Amazon SNS

Target architecture



Automation and scale

If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy this template in multiple accounts that you want to monitor.

Tools

Tools

- AWS CloudFormation is a service that helps you model and set up AWS resources using infrastructure as code.
- Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- Amazon EMR is a managed cluster platform that simplifies running big data frameworks.
- AWS Lambda supports running code without provisioning or managing servers.
- Amazon S3 is a highly scalable object storage service that can be used for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

- The EMREncryptionAtRest.zip and EMREncryptionAtRest.yml files for this project available as an attachment.

Epics

Define the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	On the Amazon S3 console, choose or create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts.	Cloud Architect

Task	Description	Skills required
	Your S3 bucket needs to be in the same Region as the Amazon EMR cluster that is being evaluated.	

Upload the Lambda code to the S3 bucket

Task	Description	Skills required
Upload the Lambda code to the S3 bucket.	Upload the Lambda code .zip file that's provided in the "Attachments" section to the defined S3 bucket.	Cloud Architect

Deploy the AWS CloudFormation template

Task	Description	Skills required
Deploy the AWS CloudForm ation template.	On the AWS CloudFormation console, in the same Region as your S3 bucket, deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters. For more information about deploying AWS CloudForm ation templates, see the "Related resources" section.	Cloud Architect

Complete the parameters in the AWS CloudFormation template

Task	Description	Skills required
Name the S3 bucket.	Enter the name of the S3 bucket that you created in the first epic.	Cloud Architect
Provide the Amazon S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, <directory>/<file-name>.zip).	Cloud Architect
Provide an email address.	Provide an active email address to receive Amazon SNS notifications.	Cloud Architect
Define the logging level.	Define the logging level and frequency for your Lambda function. "Info" designate s detailed informational messages on the application's progress. "Error" designate s error events that could still allow the application to continue running. "Warning" designates potentially harmful situations.	Cloud Architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the template successfu lly deploys, it sends a subscription email message to	Cloud Architect

Task	Description	Skills required
	the email address provided. You must confirm this email subscription to receive violation notifications.	

Related resources

- Creating a stack on the AWS CloudFormation console
- AWS Lambda
- Amazon EMR encryption options

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Ensure that an IAM profile is associated with an EC2 instance

Mansi Suratwala, Amazon Web Services

Summary

This pattern provides an AWS CloudFormation security control template that sets up automatic notification when an AWS Identity and Access Management (IAM) profile violation occurs for an Amazon Elastic Compute Cloud (Amazon EC2) instance.

An instance profile is a container for an IAM role that you can use to pass role information to an EC2 instance when the instance starts.

Amazon CloudWatch Events initiates this check when AWS CloudTrail logs Amazon EC2 API calls based on the `RunInstances`, `AssociateIamInstanceProfile`, and `ReplaceIamInstanceProfileAssociation` actions. The trigger calls an AWS Lambda function, which uses an Amazon CloudWatch Events event to check for an IAM profile.

If an IAM profile does not exist, the Lambda function initiates an Amazon Simple Notification Service (Amazon SNS) email notification that includes the Amazon Web Services (AWS) account ID and the AWS Region.

If an IAM profile does exist, the Lambda function checks for any wildcard entries in the policy documents. If the wildcards entries exist, initiates an Amazon SNS violation notification, which helps you to implement enhanced security. The notification contains the name of the IAM profile, the event, the EC2 instance ID, the name of the managed policy, the violation, the account ID, and the Region.

Prerequisites and limitations

Prerequisites

- An active account
- An Amazon Simple Storage Service (Amazon S3) bucket for the Lambda code .zip file

Limitations

- The AWS CloudFormation template must be deployed for the `RunInstances`, `AssociateIamInstanceProfile`, and `ReplaceIamInstanceProfileAssociation` actions only.

- The security control does not monitor the detachment of IAM profiles.
- The security control does not check for modification of IAM policies that are attached to the EC2 instance IAM profile.
- The security control does not account for unsupported resource-level permissions that require the use of `"Resource":*`.

Architecture

Target technology stack

- Amazon EC2
- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda
- Amazon S3
- Amazon SNS

Target architecture

Automation and scale

You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to launch the template only one time for each account or Region.

Tools

Tools

- Amazon EC2 – Amazon EC2 provides scalable computing capacity (virtual servers) in the AWS Cloud.
- AWS CloudTrail – AWS CloudTrail helps you enable governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, a role, or an AWS service are recorded as events in CloudTrail.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS Lambda – AWS Lambda is a compute service that you can use to run code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon S3 provides highly scalable object storage that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- Amazon SNS – Amazon SNS enables applications and devices to send and receive notifications from the cloud.

Code

- A .zip file of the project is available as an attachment.

Epics

Define the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	To host the Lambda code .zip file, choose or create an S3 bucket with a unique name that does not contain leading slashes. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. Your S3 bucket needs to be in the same	Cloud Architect

Task	Description	Skills required
	Region as the EC2 instance that is being evaluated.	

Upload the Lambda code to the S3 bucket

Task	Description	Skills required
Upload the Lambda code to the S3 bucket.	Upload the Lambda code that's provided in the *Attachments* section to the S3 bucket. The S3 bucket must be in the same Region as the EC2 instance being evaluated.	Cloud Architect

Deploy the AWS CloudFormation template

Task	Description	Skills required
Deploy the AWS CloudForm ation template.	Deploy the AWS CloudForm ation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters.	Cloud Architect

Complete the parameters in the AWS CloudFormation template

Task	Description	Skills required
Name the S3 bucket.	Enter the name of the S3 bucket that you created in the first epic.	Cloud Architect

Task	Description	Skills required
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `<directory>/<file-name>.zip`).	Cloud Architect
Provide an email address.	Provide an active email address to receive Amazon SNS notifications.	Cloud Architect
Define the logging level.	Define the logging level and frequency for your Lambda function. `Info` designates detailed informational messages on the application's progress. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	Cloud Architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the template successfully deploys, it sends a subscription email message to the email address provided. You must confirm this email subscription to receive violation notifications.	Cloud Architect

Related resources

- Creating an S3 bucket
- Uploading files to an S3 bucket
- Using instance profiles
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Ensure an Amazon Redshift cluster is encrypted upon creation

Mansi Suratwala, Amazon Web Services

Summary

This pattern provides an AWS CloudFormation template that provides you with automatic notification when a new Amazon Redshift cluster is created without encryption.

The AWS CloudFormation template creates an Amazon CloudWatch Events event and an AWS Lambda function. The event watches for any Amazon Redshift cluster being created or being restored from a snapshot through AWS CloudTrail. If the cluster is created without AWS Key Management Service (AWS KMS) or cloud hardware security model (HSM) encryption in the AWS account, CloudWatch initiates a Lambda function that sends you an Amazon Simple Notification Service (Amazon SNS) notification informing you of the violation.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- A virtual private cloud (VPC) with a cluster subnet group, and an associated security group.

Limitations

- The AWS CloudFormation template can be deployed for the `CreateCluster` and `RestoreFromClusterSnapshot` actions only.

Architecture

Target technology stack

- Amazon Redshift
- AWS CloudTrail
- Amazon CloudWatch
- AWS Lambda

- Amazon Simple Storage Service (Amazon S3)
- Amazon SNS

Target architecture

Automation and scale

You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run it only one time in each Region or account.

Tools

Tools

- Amazon Redshift – Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. Amazon Redshift is integrated with your data lake, which enables you to use your data to acquire new insights for your business and customers.
- AWS CloudTrail – AWS CloudTrail is an AWS service that helps you implement governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.
- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS Lambda – AWS Lambda supports running code without provisioning or managing servers. AWS Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon S3 is a highly scalable object storage service that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.

- Amazon SNS – Amazon SNS is a web service that coordinates and manages the delivery or sending of messages to between publishers and clients, including web servers and email addresses.

Code

- A .zip file of the project is available as an attachment.

Epics

Define the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	On the Amazon S3 console, choose or create an S3 bucket. This S3 bucket will host the Lambda code .zip file. Your S3 bucket needs to be in the same Region as the Amazon Redshift cluster being evaluated. The S3 bucket's name cannot contain leading slashes.	Cloud Architect

Upload the Lambda code to the S3 bucket

Task	Description	Skills required
Upload the Lambda code to the S3 bucket.	Upload the Lambda code provided in the *Attachments* section to the S3 bucket. The S3 bucket must be in the same Region as the Amazon Redshift cluster being evaluated.	Cloud Architect

Deploy the AWS CloudFormation template

Task	Description	Skills required
Deploy the AWS CloudFormation template.	Deploy the AWS CloudFormation template that's provided as an attachment to this pattern. In the next epic, provide the values for the parameters.	Cloud Architect

Complete the parameters in the AWS CloudFormation template

Task	Description	Skills required
Name the S3 bucket.	Enter the name of the S3 bucket that you created in the first epic.	Cloud Architect
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `<directory>/<file-name>.zip`).	Cloud Architect
Provide an email address.	Provide an active email address to receive Amazon SNS notifications.	Cloud Architect
Define the logging level.	Define the logging level and frequency for your Lambda function. `Info` designates detailed informational messages on the application's progress. `Error` designates error events that could	Cloud Architect

Task	Description	Skills required
	still allow the application to continue running. `Warning` designates potentially harmful situations.	

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the template successfully deploys, it sends a subscription email to the email address provided. You must confirm this email subscription to receive violation notifications.	Cloud Architect

Related resources

- Creating an S3 bucket
- Uploading files to an S3 bucket
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail
- Creating an Amazon Redshift cluster

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Monitor and remediate scheduled deletion of AWS KMS keys

Mikesh Khanal and Ramya Pulipaka, Amazon Web Services

Summary

On the Amazon Web Services (AWS) Cloud, deleting an AWS Key Management Services (AWS KMS) key can result in data loss. Deletion removes the key material and all metadata associated with the AWS KMS key, and it is irreversible. After an AWS KMS key is deleted, you can no longer decrypt the data that were encrypted under that AWS KMS key, so that data cannot be recovered.

This pattern sets up monitoring, with notifications when an application or a user schedules an AWS KMS key for deletion. If you receive a notification, you might want to cancel deletion of the AWS KMS key and reconsider your decision to delete it. The pattern uses the AWS Systems Manager automation runbook [AWSConfigRemediation-CancelKeyDeletion] to facilitate canceling the deletion of an AWS KMS key.

Note

The pattern's CloudFormation template must be deployed in all AWS Regions where you want to monitor deletion of AWS KMS keys.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Understanding of the following AWS services:
  - Amazon EventBridge
  - AWS KMS
  - Amazon Simple Notification Service (Amazon SNS)
  - AWS Systems Manager

Limitations

- Any customization of the solution requires knowledge of AWS CloudFormation templates and the AWS services used in this pattern.

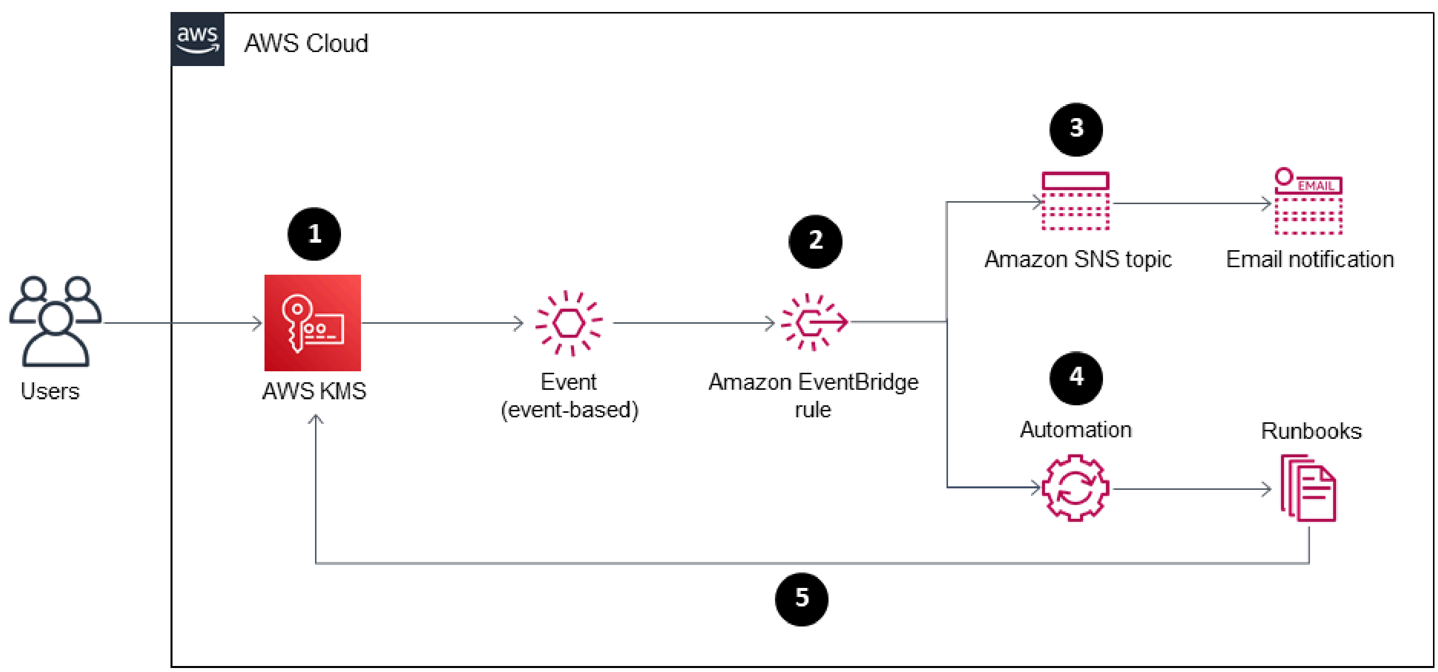
- Currently, this solution uses the default event bus, and it can be customized according to the requirements. For more information about the custom event bus, see the AWS documentation.

Architecture

Target technology stack

- Amazon EventBridge
- AWS KMS
- Amazon SNS
- AWS Systems Manager
- Automation using the following:
  - AWS Command Line Interface (AWS CLI) or AWS SDK
  - AWS CloudFormation stack

Target architecture



1. Deletion of an AWS KMS key is scheduled.
2. The scheduled-deletion event is evaluated by an EventBridge rule.
3. The EventBridge rule engages the Amazon SNS topic.

4. The EventBridge rule initiates the Systems Manager automation and runbooks.
5. The runbooks cancel the deletion.

Automation and scale

The CloudFormation stack deploys all the necessary resources and services for this solution to work. The pattern can be run independently in a single account or run using AWS CloudFormation StackSets for multiple independent accounts or an organization.

```
aws cloudformation create-stack --stack-name  <stack-name>\
    --template-body file://<Full-Path-of-file> \
    --parameters ParameterKey=,ParameterValue= \
    --capabilities CAPABILITY_NAMED_IAM
```

Tools

Tools

- AWS CloudFormation – AWS CloudFormation is a service that helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run on AWS. You can use a CloudFormation template to create stacks in an AWS account in an AWS Region. The template describes all the AWS resources that you want, and CloudFormation provisions and configures those resources for you.
- AWS CLI – The AWS Command Line Interface (AWS CLI) is an open source tool that you can use to interact with AWS services using commands in your command line shell.
- Amazon EventBridge – Amazon EventBridge is a serverless event bus service connecting your applications with data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications and AWS services, and it routes that data to targets such as AWS Lambda. EventBridge simplifies the process of building event-driven architectures.
- AWS KMS – AWS Key Management Service (AWS KMS) is a managed service for creating and controlling AWS KMS keys, the encryption keys used to encrypt your data.
- AWS SDKs – AWS tools include SDKs so that you can develop and manage applications on AWS in the programming language of your choice.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and

consumers). Publishers communicate asynchronously with subscribers by sending messages to a topic, which is a logical access point and communication channel.

- [AWS Systems Manager](#) – AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can automate operational tasks across your AWS resources. Systems Manager helps you maintain security and compliance by scanning your managed instances and reporting on (or taking corrective action on) any policy violations it detects.

Code

- The `alerting_ct_logs.yaml` CloudFormation template for the project is attached.

Epics

Prepare the AWS account

Task	Description	Skills required
Install and configure AWS CLI.	Install AWS CLI version 2. Then configure the security credentials settings for an identity, the default output format, and the default AWS Region that AWS CLI uses to interact with AWS. The identity must have the required permissions to perform the tasks.	Developer, Security engineer

Deploy the AWS CloudFormation template

Task	Description	Skills required
Download the CloudForm ation template.	Download the attachment to a local path on your computer	Developer, Security engineer

Task	Description	Skills required
	and extract the `alerting_ct_logs.yaml` template file.	
Deploy the template.	In the terminal window where the AWS account profile has been configured, run the following command. ```aws cloudformation create-stack --stack-name <stack_name> \ --capabilities <Value> \ --template-body file://<Full_Path> \ --parameters ParameterKey=DestinationEmailAddress,ParameterValue=<Value> \ ParameterKey=SNSTopicName,ParameterValue=<Value> \ ParameterKey=EnableRemediation ,ParameterValue=<Value> \ ParameterKey=AutomationAssumeRole,ParameterValue=<Value>``` In the next step, enter values for the template parameters.	Developer, Security engineer

Task	Description	Skills required
Complete the template parameters.	Enter the required values for the parameters. • `DestinationEmailAddress` – The email address to receive an alert when an AWS KMS key is scheduled for deletion. • `SNSTopicName` – The name of the Amazon SNS topic. • `EnableRemediation` – Cancellation of the scheduled key deletion using a Systems Manager runbook. Allowed values are `true` and `false`. • `AutomationAssumeRole` – The Amazon Resource Name (ARN) of the role that allows Systems Manager automation to perform the actions on your behalf. For more information, see the *Required IAM Permissions* section in the AWSConfigRemediation-CancelKeyDeletion documentation. • `Capabilities` – For AWS CloudFormation to create the stack, you must explicitly acknowledge that your	Developer, Security engineer

Task	Description	Skills required
	stack template contains certain capabilities.	

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	Check your email inbox and choose **Confirm subscription** in the email message that you receive from Amazon SNS. A web browser window will open and display a subscript ion confirmation and your subscription ID.	Developer, Security engineer

Related resources

References

- Creating a rule for an AWS service
- Creating an Amazon CloudWatch alarm to detect usage of an AWS KMS key that is pending deletion

Tutorials and videos

- How to get started with Amazon EventBridge
- Deep dive on Amazon EventBridge (AWS Online Tech Talks)

AWS workshop

- Working with EventBridge rules

Additional information

The following code provides examples for extending the solution to monitor and notify you of any changes in any AWS service. The examples include predefined patterns and custom patterns. For more information, see Events and event patterns in EventBridge.

```
EventPattern:
        source:
        - aws.kms
        detail-type:
        - AWS API Call via CloudTrail
        detail:
          eventSource:
          - kms.amazonaws.com
          eventName:
          - ScheduleKeyDeletion
```

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Identify public Amazon S3 buckets in AWS Organizations by using Security Hub

Mourad Cherfaoui, Arun Chandapillai, and Parag Nagwekar, Amazon Web Services

Summary

This pattern shows you how to build a mechanism for identifying public Amazon Simple Storage Service (Amazon S3) buckets in your AWS Organizations accounts. The mechanism works by using controls from the AWS Foundational Security Best Practices (FSBP) standard in AWS Security Hub to monitor Amazon S3 buckets. You can use Amazon EventBridge to process Security Hub findings, and then post these findings to an Amazon Simple Notification Service (Amazon SNS) topic. Stakeholders in your organization can subscribe to the topic and get immediate email notifications about the findings.

New Amazon S3 buckets and their objects don't allow public access by default. You can use this pattern in scenarios where you must modify default Amazon S3 configurations based on your organization's requirements. For example, this could be a scenario where you have an Amazon S3 bucket that hosts a public-facing website or files that everyone on the internet must be able to read from your Amazon S3 bucket.

Security Hub is often deployed as a central service to consolidate all security findings, including those related to security standards and compliance requirements. There are other AWS services that you can use to detect public Amazon S3 buckets, but this pattern uses an existing Security Hub deployment with minimal configuration.

Prerequisites and limitations

Prerequisites

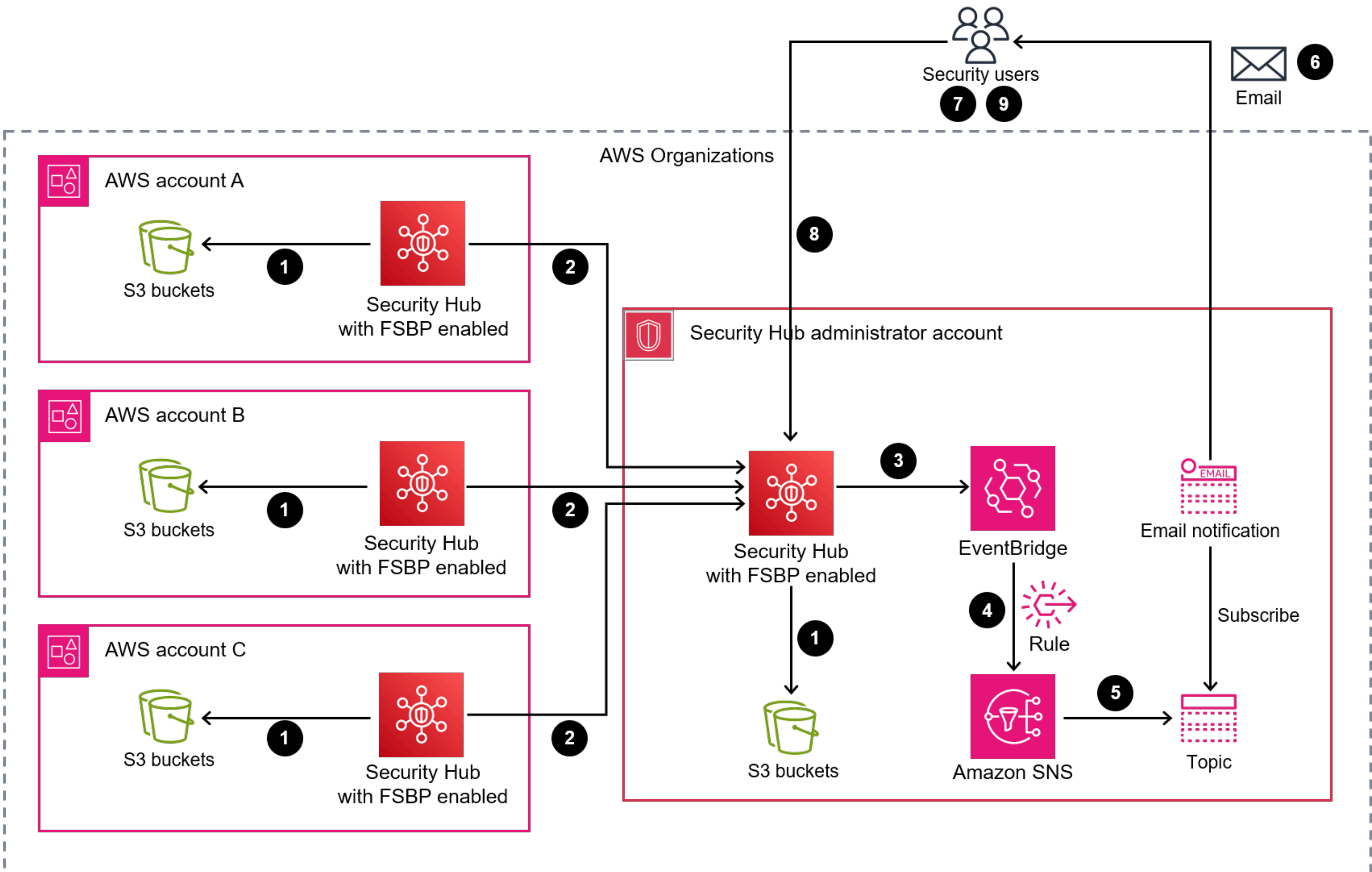
- An AWS multi-account setup with a dedicated Security Hub administrator account
- Security Hub and AWS Config, enabled in the AWS Region that you want to monitor

> **Note**
>
> You must enable cross-Region aggregation in Security Hub if you want to monitor multiple Regions from a single aggregation Region.

- User permissions for accessing and updating the Security Hub administrator account, read access to all the Amazon S3 buckets in the organization, and permissions for turning off public access (if required)

Architecture

The following diagram shows an architecture for using Security Hub to identify public Amazon S3 buckets.



The diagram show the following workflow:

1. Security Hub monitors the configuration of Amazon S3 buckets in all AWS Organizations accounts (including the administrator account) by using the S3.2 and S3.3 controls from the FSBP security standard, and detects a finding if a bucket is configured as public.
2. The Security Hub administrator account accesses the findings (including those for S3.2 and S3.3) from all member accounts.

3. Security Hub automatically sends all new findings and all updates to existing findings to EventBridge as **Security Hub Findings - Imported** events. This includes events for findings from both the administrator and member accounts.
4. An EventBridge rule filters on findings from S3.2 and S3.3 that have a `ComplianceStatus` of `FAILED`, a workflow status of `NEW`, and a `RecordState` of `ACTIVE`.
5. Rules use the event patterns to identify events and send them to an Amazon SNS topic once matched.
6. An Amazon SNS topic sends the events to its subscribers (through email, for example).
7. Security analysts designated to receive the email notifications review the Amazon S3 bucket in question.
8. If the bucket is approved for public access, the security analyst sets the workflow status of the corresponding finding in Security Hub to `SUPPRESSED`. Otherwise, the analyst sets the status to `NOTIFIED`. This eliminates future notifications for the Amazon S3 bucket and reduces notification noise.
9. If the workflow status is set to `NOTIFIED`, the security analyst reviews the finding with the bucket owner to determine if public access is justified and complies with privacy and data protection requirements. The investigation results in either removing public access to the bucket or approving public access. In the latter case, the security analyst sets the workflow status to `SUPPRESSED`.

Note

The architecture diagram applies to both single Region and cross-Region aggregation deployments. In accounts A, B, and C in the diagram, Security Hub can belong to the same Region as the administrator account or belong to different Regions if cross-Region aggregation is enabled.

Tools

- [Amazon EventBridge](#) is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. EventBridge delivers a stream of real-time data from your own applications, software as a service (SaaS) applications, and AWS services. EventBridge routes that data to targets such as Amazon SNS topics and AWS Lambda functions if the data matches user-defined rules.

- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Security Hub provides a comprehensive view of your security state in AWS. Security Hub also helps you check your AWS environment against security industry standards and best practices. Security Hub collects security data from across AWS accounts, services, and supported third-party partner products, and then helps to analyze security trends and identify the highest priority security issues.

Epics

Configure Security Hub accounts

Task	Description	Skills required
Enable Security Hub in AWS Organizations accounts.	To enable Security Hub in the organization accounts where you want to monitor Amazon S3 buckets, see the guideline s from Designating a Security Hub administrator account (console) and Managing member accounts that belong to an organization in the Security Hub documentation.	AWS administrator
(Optional) Enable cross-Reg ion aggregation.	If you want to monitor Amazon S3 buckets in multiple Regions from a single Region, set up cross-Region aggregation.	AWS administrator

Task	Description	Skills required
Enable the S3.2 and S3.3 controls for the FSBP security standard.	You must enable S3.2 and S3.3 controls for the FSBP security standard. 1. To enable S3.2 controls, follow the instructions from [S3.2] S3 buckets should prohibit public read access in the Security Hub documentation. 2. To enable S3.3 controls, follow the instructions from [3] S3 buckets should prohibit public write access in the Security Hub documentation.	AWS administrator

Set up the environment

Task	Description	Skills required
Configure the Amazon SNS topic and email subscription.	1. Sign in to the AWS Management Console and open the Amazon SNS console. 2. In the navigation pane, choose **Topics**, and then choose **Create topic**. 3. For **Type**, choose **Standard**. 4. For **Name**, enter a name for your topic (for example, **public-s3-buckets**). 5. Choose **Create topic**.	AWS administrator

Task	Description	Skills required
	6. On the **Subscriptions** tab for your topic, choose **Create subscription**. 7. For **Protocol**, select **Email**. 8. For **Endpoint**, enter the email address that will receive the notificat ions. You can use the email address of an AWS administrator, IT professio nal, or Infosec professional. 9. Choose **Create subscript ion**. To create additional email subscriptions, repeat steps 6–8 as needed.	

Task	Description	Skills required
Configure the EventBridge rule.	1. Open the [EventBridge console](). 2. In the **Get started** section, select **EventBridge Rule**, and then choose **Create rule**. 3. On the **Define rule detail** page, for **Name**, enter a name for your rule (for example, **public-s3-buckets**). Choose **Next**. 4. In the **Event pattern** section, choose **Edit pattern**. 5. Copy the following code, paste it into the **Event pattern** code editor, and then choose **Next**. `{ "source": ["aws.securityhub"], "detail-type": ["Security Hub Findings - Imported"], "detail": { "findings": { "Compliance": { "Status": ["FAILED"] }, "RecordState": ["ACTIVE"], "Workflow": { "Status": ["NEW"]`	AWS administrator

Task	Description	Skills required

	``
	      },
	      "ProductF
	ields": {
	        "ControlId":
	 ["S3.2", "S3.3"]
	      }
	    }
	  }
	}
	``
	6. On the **Select target(s)** page, for **Select a target**, select **SNS topic** as the target, and then select the topic that you created earlier.
	7. Choose **Next**, choose **Next** again, and then choose **Create rule**.	

Troubleshooting

Issue	Solution
I have an Amazon S3 bucket with public access enabled, but I'm not getting email notifications for it.	This could be because the bucket was created in another Region and cross-Region aggregation is not enabled in the Security Hub administrator account. To resolve this issue, enable cross-Region aggregation or implement this pattern's solution in the Region where your Amazon S3 bucket currently resides.

Related resources

- What is AWS Security Hub? (Security Hub documentation)
- AWS Foundational Security Best Practices (FSBP) standard (Security Hub documentation)
- AWS Security Hub multi-account enable scripts (AWS Labs)
- Security best practices for Amazon S3 (Amazon S3 documentation)

Additional information

Workflow for monitoring public Amazon S3 buckets

The following workflow illustrates how you can monitor the public Amazon S3 buckets in your organization. The workflow assumes that you completed the steps in the *Configure the Amazon SNS topic and email subscription* story of this pattern.

1. You receive an email notification when an Amazon S3 bucket is configured with public access.
   - If the bucket is approved for public access, set the workflow status of the corresponding finding to `SUPPRESSED` in the Security Hub administrator account. This prevents Security Hub from issuing further notifications for this bucket and can eliminate duplicate alerts.
   - If the bucket isn't approved for public access, set the workflow status of the corresponding finding in the Security Hub administrator account to `NOTIFIED`. This prevents Security Hub from issuing further notifications for this bucket from Security Hub and can eliminate noise.
2. If the bucket might contain sensitive data, turn off public access immediately until the review is completed. If you turn off public access, then Security Hub changes the workflow status to `RESOLVED`. Then, email notifications for the bucket stop.
3. Find the user who configured the bucket as public (for example, by using AWS CloudTrail) and start a review. The review results in either removing public access to the bucket or approving public access. If public access is approved, then set the workflow status of the corresponding finding to `SUPPRESSED`.

Ingest and analyze AWS security logs in Microsoft Sentinel

Ivan Girardi and Sebastian Wenzel, Amazon Web Services

Summary

This pattern describes how to automate the ingestion of AWS security logs, such as AWS CloudTrail logs, Amazon CloudWatch Logs data, Amazon VPC Flow Logs data, and Amazon GuardDuty findings, into Microsoft Sentinel. If your organization uses Microsoft Sentinel as a security information and event management (SIEM) system, this helps you centrally monitor and analyze logs in order to detect security-related events. As soon as the logs are available, they are automatically delivered to an Amazon Simple Storage Service (Amazon S3) bucket in less than 5 minutes. This can help you quickly detect security events in your AWS environment.

Microsoft Sentinel ingests CloudTrail logs in a tabular format that includes the original timestamp for when the event was recorded. The structure of the ingested logs enables query capabilities by using Kusto Query Language in Microsoft Sentinel.

The pattern deploys a monitoring and alerting solution that detects ingestion failures in less than 1 minute. It also includes a notification system that the external SIEM can monitor. You use AWS CloudFormation to deploy the required resources in the logging account.

Target audience

This pattern is recommended for users who have experience with AWS Control Tower, AWS Organizations, CloudFormation, AWS Identity and Access Management (IAM), and AWS Key Management Service (AWS KMS).

Prerequisites and limitations

Prerequisites

The following are the prerequisites for deploying this solution:

- Active AWS accounts that are managed as an organization in AWS Organizations and are part of an AWS Control Tower landing zone. The organization should include a dedicated account for logging. For instructions, see Creating and configuring an organization in the AWS Organizations documentation.
- A CloudTrail trail that logs events for the entire organization and stores logs in an Amazon S3 bucket in the logging account. For instructions, see Creating a trail for an organization.

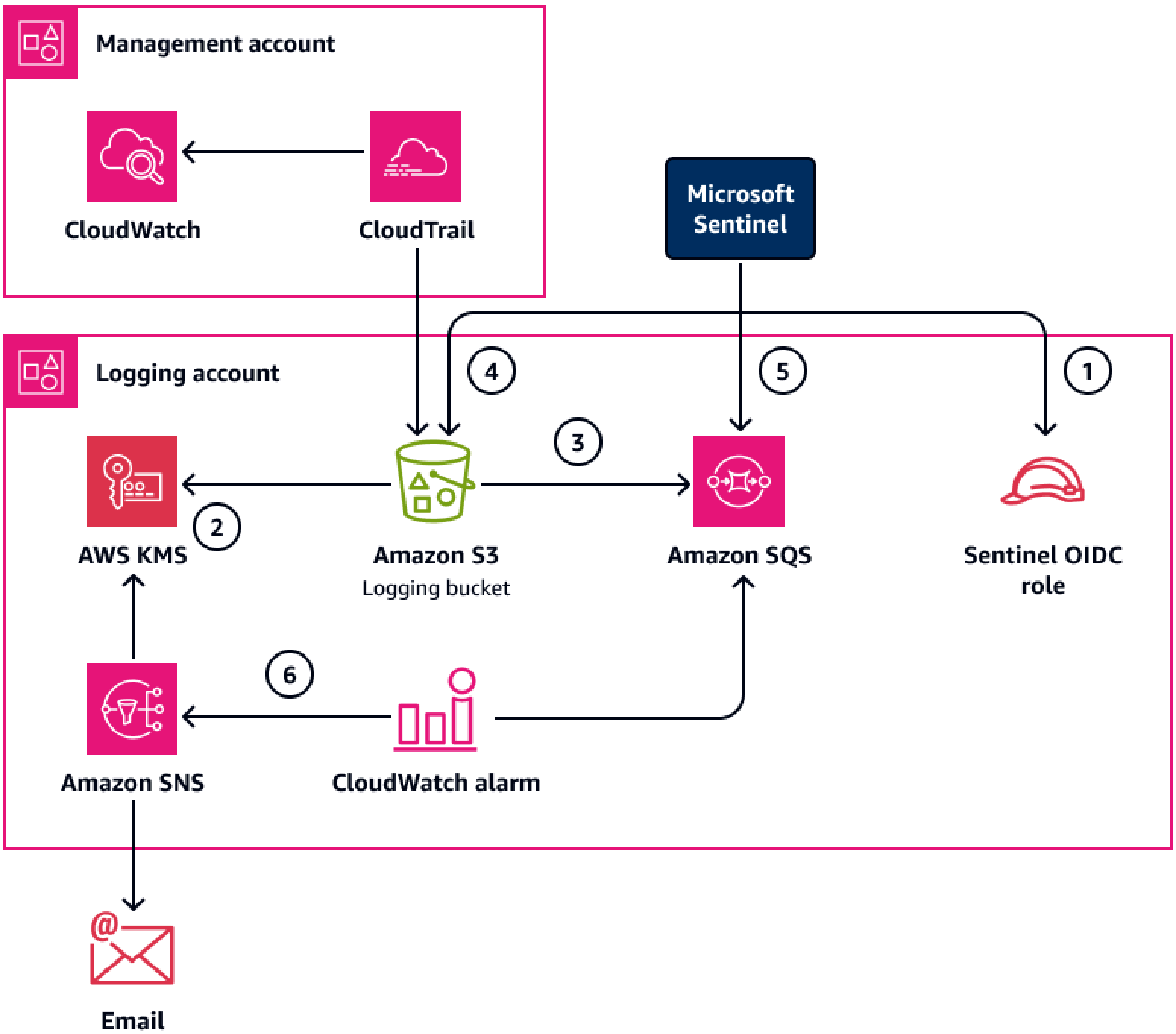
- In the logging account, permissions to assume an existing IAM role that has the following permissions:
  - Deploy the resources defined in the provided CloudFormation template.
  - Deploy the provided CloudFormation template.
  - Modify the AWS KMS key policy if the logs are encrypted with a customer managed key.
- AWS Command Line Interface (AWS CLI), installed and configured.
- A Microsoft Azure account with a subscription to use Microsoft Sentinel.
- Enable and set up Microsoft Sentinel. For instructions, see Enable Microsoft Sentinel and initial features and content in the Microsoft Sentinel documentation.
- Meet the prerequisites for setting up the Microsoft Sentinel S3 connector.

Limitations

- This solution forwards the security logs from an Amazon S3 bucket in the logging account to Microsoft Sentinel. Instructions for how to send the logs to Amazon S3 are not explicitly provided.
- This pattern provides instructions for deployment in an AWS Control Tower landing zone. However, use of AWS Control Tower is not required.
- This solution is compatible with an environment where the Amazon S3 logging bucket is restricted with service control policies (SCPs), such as Disallow Changes to Bucket Policy for AWS Control Tower Created Amazon S3 Buckets in Log Archive.
- This pattern provides instructions for forwarding CloudTrail logs, but you can adapt this solution to send other logs that Microsoft Sentinel supports, such as logs from CloudWatch Logs, Amazon VPC Flow Logs, and GuardDuty.
- The instructions use the AWS CLI to deploy the CloudFormation template, but you could also use the AWS Management Console. For instructions, see Using the AWS CloudFormation console. If you use the console to deploy the stack, deploy the stack in the same AWS Region as the logging bucket.
- This solution deploys an Amazon Simple Queue Service (Amazon SQS) queue to deliver Amazon S3 notifications. The queue contains messages with the paths of objects uploaded in the Amazon S3 bucket, not actual data. The queue uses SSE-SQS encryption to help protect the content of the messages. If you want to encrypt the SQS queue with SSE-KMS, you can use a customer managed KMS key. For more information, see Encryption at rest in Amazon SQS.

Architecture

This section provides a high-level overview of the architecture that the sample code establishes. The following diagram shows the resources deployed in the logging account in order to ingest logs from an existing Amazon S3 bucket into Microsoft Sentinel.



The architecture diagram shows the following resource interactions:

1. In the logging account, Microsoft Sentinel assumes an IAM role through OpenID Connect (OIDC) to access logs in a specific Amazon S3 bucket and Amazon SQS queue.

2. Amazon Simple Notification Service (Amazon SNS) and Amazon S3 use AWS KMS for encryption.
3. Amazon S3 sends notification messages to the Amazon SQS queue whenever it receives new logs.
4. Microsoft Sentinel checks Amazon SQS for new messages. The Amazon SQS queue uses SSE-SQS encryption. The message retention period is set to 14 days.
5. Microsoft Sentinel pulls messages from the Amazon SQS queue. The messages contain the path of the uploaded Amazon S3 objects. Microsoft Sentinel ingests those objects from the Amazon S3 bucket into the Microsoft Azure account.
6. A CloudWatch alarm monitors the Amazon SQS queue. If messages are not received and deleted from the Amazon SQS queue within 5 minutes, then it initiates an Amazon SNS notification that sends an email.

AWS Control Tower helps you set up the foundational organization unit (OU) structure and centralizes CloudTrail logs in the logging account. It also implements mandatory SCPs to protect the logging bucket.

We have provided the target architecture in an AWS Control Tower landing zone, but this is not strictly required. In this diagram, the resources in the management account reflect an AWS Control Tower deployment and a CloudTrail trail that logs events for the entire organization.

This pattern focuses on the deployment of resources in the logging account. If the logs stored in Amazon S3 in your AWS Control Tower landing zone are encrypted with a customer managed KMS key, then you must update the key policy to allow Microsoft Sentinel to decrypt the logs. In an AWS Control Tower landing zone, you manage the key policy from the management account, which is where the key was created.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- Amazon CloudWatch helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- AWS Control Tower helps you set up and govern an AWS multi-account environment, following best practices.

- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon Simple Queue Service (Amazon SQS) provides a secure, durable, and available hosted queue that helps you integrate and decouple distributed software systems and components.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- Microsoft Sentinel is a cloud-native SIEM system that provides security orchestration, automation, and response (SOAR).

Code repository

The code for this pattern is available in the GitHub Ingest and analyze AWS security logs in Microsoft Sentinel repository.

Best practices

- Follow the principle of least-privilege (IAM documentation).
- Follow the Best practices for AWS Control Tower administrators (AWS Control Tower documentation).
- Follow the AWS CloudFormation best practices (CloudFormation documentation).
- Use code analysis tools, such as cfn_nag, to scan the generated CloudFormation templates. The cfn_nag tool identifies potential security issues in CloudFormation templates by searching for patterns.

Epics

Connect Microsoft Sentinel to Amazon S3

Task	Description	Skills required
Prepare the Microsoft Sentinel S3 connector.	1. In Microsoft Sentinel, choose **Data connectors**. 2. From the data connector s gallery, choose **Amazon Web Services S3**. **Note** If you don't see the connector, install the [Amazon Web Services solution](#) from the Content Hub in Microsoft Sentinel. 3. In the details pane for the connector, choose **Open connector page**. 4. In the **Configuration** section, copy the **External ID**. You need this ID later.	DevOps engineer, General AWS

Deploy the CloudFormation stack

Task	Description	Skills required
Clone the repository.	In a bash shell, enter the following command. This clones the [Ingest and](#)	DevOps engineer, General AWS

Task	Description	Skills required
	analyze AWS Security Logs in Microsoft Sentinel repository. `git clone https://github.com/aws-samples/ingest-and-analyze-aws-security-logs-in-microsoft-sentinel.git`	
Assume the IAM role in the logging account.	In the logging account, assume the IAM role that has permissions to deploy the CloudFormation stack. For more information about assuming an IAM role in the AWS CLI, see Use an IAM role in the AWS CLI.	DevOps engineer, General AWS

Task	Description	Skills required
Deploy the stack.	To deploy the CloudFormation stack enter the following command, where: • `<Bucket name>` is the name of the logging Amazon S3 bucket. • `<Sentinel external ID>` is the external ID of the Amazon S3 connector in Microsoft Sentinel. • `<Email address>` is a valid email address where you want to receive notifications. • `<Customer managed key ARN>` is the Amazon Resource Name (ARN) of the customer managed KMS key. Provide this parameter only if the logs are encrypted with a customer managed KMS key. • `<Suffix>` is an optional parameter to avoid resource name conflicts. • `<ARN for the OIDC provider>` is the ARN of the OIDC provider if it already exists. If you do not provide this parameter, CloudFormation creates the OIDC provider.	DevOps engineer, General AWS

Task	Description	Skills required
	Important If the AWS organization is monitored with [Microsoft Code Defender](), the OIDC provider for Microsoft has been already deployed. You must provide this parameter and the ARN of the existing provider. `aws cloudformation deploy --stack-name cloudtrail-sentinel-integration \ --no-fail-on-empty-changeset \ --template-file template.yml \ --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM CAPABILITY_AUTO_EXPAND \ --parameter-overrides \ ControlTowerS3BucketName="<Bucket name>" \ AzureWorkspaceID="<Sentinel external ID>" \`	

Task	Description	Skills required

	``
	    EmailAddress="<Ema
	il address>" \
	    KMSKeyArn="<Custom
	er managed key ARN>" \
	    Suffix="<Suffix to
	 avoid name conflicts>"
	 \
	    OIDCProviderArn="<
	ARN for the OIDC
	 provider>"
	``	

Copy outputs.	From the output of the CloudFormation stack, copy the values for `SentinelRoleArn` and `SentinelSQS` . You use these values later to complete the configuration in Microsoft Sentinel.	DevOps engineer, General AWS

Task	Description	Skills required

Modify the key policy.	If you aren't using a customer managed KMS key to encrypt the logs in the Amazon S3 bucket, you can skip this step.
	
	If the logs are encrypted with a customer managed KMS key, modify the key policy to grant Microsoft Sentinel permission to decrypt the logs. The following is an example key policy. This example policy allows cross-account access if the KMS key is in another AWS account.
	
	`{`
	`    "Version": "2012-10-17",`
	`    "Id": "key-policy",`
	`    "Statement": [`
	`        ...`
	`        {`
	`            "Sid": "Grant access to decrypt",`
	`            "Effect": "Allow",`
	`            "Principal": {`
	`                "AWS": "<SentinelRoleArn>"`
	`            },`
	`            "Action": "kms:Decrypt",`
	`            "Resource": "<KeyArn>"`
	`        }`
	`    ]`
		DevOps engineer, General AWS

Task	Description	Skills required
	`}`	

Configure the connector in Microsoft Sentinel

Task	Description	Skills required
Complete the configuration in Microsoft Sentinel.	1. In Microsoft Sentinel, choose **Data connectors**. 2. From the data connectors gallery, choose **Amazon Web Services S3**. 3. In the details pane for the connector, choose **Open connector page**. 4. In the **Configuration** section, do the following: a. In **Role to add**, enter the `SentinelRoleArn` value that you copied. b. In **SQS URL**, enter the `SentinelSQS` value that you copied. c. In the **Destination table** list, choose `AWSCloudTrail` . 5. Choose **Add connection**.	DevOps engineer
Send Amazon S3 event notifications to Amazon SQS.	Follow the instructions in [Enabling and configuring event notifications using the Amazon S3 console](#) to configure the Amazon S3 logging bucket to send event	DevOps engineer, General AWS

Task	Description	Skills required
	notifications to the Amazon SQS queue. If CloudTrail has been configured for the whole organization, logs in the this bucket have the prefix `<OrgID>/AWSLogs/<OrgID>/` , where `<OrgID>` is the organization ID. For more information, see [Viewing details about your organization]().	
Confirm that the logs are ingested.	1. Wait until the logs are ingested in Microsoft Sentinel. This can take several minutes. 2. In Microsoft Sentinel, open the Amazon S3 Data Connector page, and then do the following: • Confirm that the Amazon S3 Data Connector status is `Connected` . • Check the data volume in the **Data received** graph. For more information about inspecting the data connector activity, see [Data connector]() in the Microsoft documentation.	DevOps engineer

Validate the solution

Task	Description	Skills required
Compare CloudWatch and Sentinel logs.	In the default configura tion of AWS Control Tower, CloudTrail logs are sent to Amazon CloudWatch and stored in the AWS Control Tower management account. For more information, see [Logging and monitoring in AWS Control Tower](). Use the following steps to confirm that that logs are automatic ally ingested into Microsoft Sentinel: 1. Open the [CloudWatch console](). 2. In the navigation pane, choose **Logs**, and then choose **Logs Insights**. 3. For **Select log group(s)**, select the log group where the CloudTrail logs are stored, such as `aws-controltower/CloudTrail Logs` . 4. In the query editor box, enter `fields eventID`. 5. Choose **Run query**. 6. Choose **Export results**, and then choose **Copy table to clipboard (CSV)**.	DevOps engineer, General AWS

Task	Description	Skills required

	7. Paste the results to a text editor.
	8. Change the format of the output so that it can be used in a Microsoft Sentinel query. The following is an example that uses the Kusto Query Language:
	`AWSCloudTrail`
	`| where AwsEventId in`
	`(`
	`'aa08b5fe-3bfb-391a-a14e-5fcebe14dab2',`
	`'9decd805-269c-451c-b75b-762f5dce59f9'`
	`)`
	9. In Microsoft Sentinel, open the Amazon S3 Data Connector page. Next to the **Data received** graph, choose **Go to log analytics**.
	10. In the query editor box, enter the query, and then choose **Run**.
	11. In Microsoft Sentinel and CloudWatch, verify that the number of entries is the same. Adapt the time range if needed.	

Related resources

AWS documentation and resources

- AWS CLI Command Reference (AWS CLI documentation)
- Optionally configure AWS KMS keys (AWS Control Tower documentation)
- Encryption at rest in Amazon SQS (Amazon SQS documentation)
- How do I keep mailing list recipients from unsubscribing everyone on the list from my Amazon SNS topic emails? (AWS Knowledge Center)

Microsoft documentation

- Connect Microsoft Sentinel to Amazon Web Services to ingest AWS service log data
- Kusto Query Language in Microsoft Sentinel

Manage AWS Organizations policies as code by using AWS CodePipeline and Amazon Bedrock

Andre Cavalcante and Mariana Pessoa de Queiroz, Amazon Web Services

Summary

You can use *authorization policies* in AWS Organizations to centrally configure and manage access for principals and resources in your member accounts. Service control policies (SCPs) define the maximum available permissions for the AWS Identity and Access Management (IAM) roles and users in your organization. Resource control policies (RCPs) define the maximum available permissions available for resources in your organization.

This pattern helps you to manage SCPs and RCPs as infrastructure as code (IaC) that you deploy through a continuous integration and continuous deployment (CI/CD) pipeline. By using AWS CloudFormation or Hashicorp Terraform to manage these policies, you can reduce the burden associated with building and maintaining multiple authorization policies.

This pattern includes the following features:

- You create, delete, and update the authorization policies by using *manifest files* (`scp-management.json` and `rcp-management.json`).
- You work with guardrails instead of policies. You define your guardrails and their targets in the manifest files.
- The pipeline, which uses AWS CodeBuild and AWS CodePipeline, merges and optimizes the guardrails in the manifest files. For each statement in the manifest file, the pipeline combines the guardrails into a single SCP or RCP and then applies it to the defined targets.
- AWS Organizations applies the policies to your targets. A *target* can be an AWS account, an organizational unit (OU), an environment (which is a group of accounts or OUs that you define in the `environments.json` file), or a group of accounts that share an AWS tag.
- Amazon Bedrock reads the pipeline logs and summarizes all policy changes.
- The pipeline requires a manual approval. The approver can review the executive summary that Amazon Bedrock prepared, which helps them understand the changes.

Prerequisites and limitations

Prerequisites

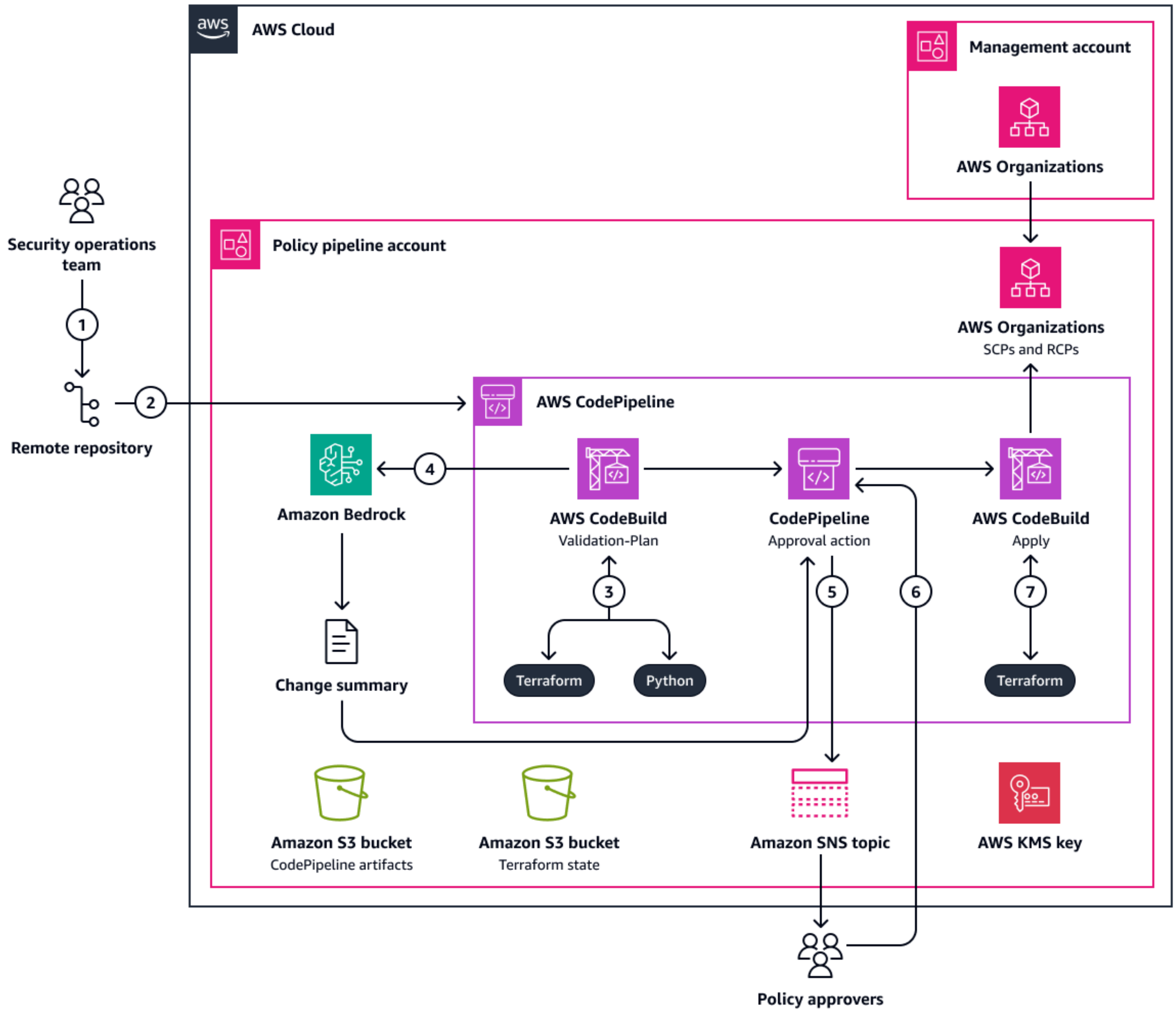
- Multiple AWS accounts that are managed as an organization in AWS Organizations. For more information, see Creating an organization.
- The SCP and RCP features are enabled in AWS Organizations. For more information, see Enabling a policy type.
- Terraform version 1.9.8 or later is installed.
- If you are not deploying this solution through a Terraform pipeline, then the Terraform state file must be stored in an Amazon Simple Storage Service (Amazon S3) bucket in the AWS account where you are deploying the policy management pipeline.
- Python version 3.13.3 or later is installed.

Limitations

- You cannot use this pattern to manage SCPs or RCPs that were created outside of this CI/CD pipeline. However, you can recreate existing policies through the pipeline. For more information, see *Migrating existing policies to the pipeline* in the Additional information section of this pattern.
- The number of accounts, OUs, and policies in each account are subject to the quotas and service limits for AWS Organizations.
- This pattern cannot be used to configure management policies in AWS Organizations, such as backup policies, tag policies, chat applications policies, or declarative policies.

Architecture

The following diagram shows the workflow of the policy management pipeline and its associated resources.



The diagram shows the following workflow:

1. A user commits changes to the `scp-management.json` or `rcp-management.json` manifest files in the main branch of the remote repository.
2. The change to the `main` branch initiates the pipeline in AWS CodePipeline.
3. CodePipeline starts the `Validate-Plan` CodeBuild project. This project uses a Python script in the remote repository to validate policies and the policy manifest files. This CodeBuild project does the following:
   a. Checks that the SCP and RCP manifest files contain unique statement IDs (`Sid`).

b. Uses the `scp-policy-processor/main.py` and `rcp-policy-processor/main.py` Python scripts to concatenate guardrails in the guardrails folder into a single RCP or SCP policy. It combines guardrails that have the same `Resource`, `Action`, and `Condition`.

c. Uses AWS Identity and Access Management Access Analyzer to validate the final, optimized policy. If any there are any findings, the pipeline stops.

d. Creates `scps.json` and `rcps.json` files, which Terraform uses to create resources.

e. Runs the `terraform plan` command, which creates a Terraform execution plan.

4. (Optional) The `Validate-Plan` CodeBuild project uses the `bedrock-prompt/prompt.py` script to send a prompt to Amazon Bedrock. You define the prompt in the `bedrock-prompt/prompt.txt` file. Amazon Bedrock uses Anthropic Claude Sonnet 3.5 to generate a summary of the proposed changes by analyzing the Terraform and Python logs.
5. CodePipeline uses an Amazon Simple Notification Service (Amazon SNS) topic in order to notify approvers that changes must be reviewed. If Amazon Bedrock generated a change summary, the notification includes this summary.
6. A policy approver approves the action in CodePipeline. If Amazon Bedrock generated a change summary, the approver can review the summary in CodePipeline prior to approving.
7. CodePipeline starts the `Apply` CodeBuild project. This project uses Terraform to apply the RCP and SCP changes in AWS Organizations.

The IaC template associated with this architecture also deploys the following resources that support the policy management pipeline:

- An Amazon S3 bucket for storing the CodePipeline artifacts and scripts, such as `scp-policy-processor/main.py` and `bedrock-prompt/prompt.py`
- An AWS Key Management Service (AWS KMS) key that encrypts the resources created by this solution

Tools

AWS services

- Amazon Bedrock is a fully managed AI service that makes many high-performing foundation models available for use through a unified API.

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Other tools

- HashiCorp Terraform is an IaC tool that helps you use code to provision and manage cloud infrastructure and resources.

Code repository

The code for this pattern is available in the organizations-policy-pipeline GitHub repository. The following are the key files that are contained in the `sample-repository` folder:

- In the `environments` folder, `environments.json` contains a list of environments. *Environments* are a group of targets, and they can contain AWS account IDs or organizational units (OUs).
- In the `rcp-management` folder:
  - The `guardrails` folder contains the individual guardrails for your RCPs
  - The `policies` folder contains the individual RCPs
  - The `rcp-management.json` manifest file helps you manage RCP guardrails, full RCPs, and their associated targets.
- In the `scp-management` folder:
  - The `guardrails` folder contains the individual guardrails for your SCPs
  - The `policies` folder contains the individual SCPs
  - The `scp-management.json` manifest file helps you manage SCP guardrails, full SCPs, and their associated targets.

- The `utils` folder contains scripts that can help you migrate your current SCPs and RCPs so that you can manage them through the pipeline. For more information, see the Additional information section of this pattern.

Best practices

- Before you set up the pipeline, we recommend that you verify that you have not reached the limits of your AWS Organizations quotas.
- We recommend that you use the AWS Organizations management account only for tasks that must be performed in that account. For more information, see Best practices for the management account.

Epics

Set up the target account

Task	Description	Skills required
Create a repository.	Create a repository from which your security operation s team will manage the policies. Use one of the third-party repository providers that AWS CodeConnections supports.	DevOps engineer
Delegate policy administr ation.	Delegate administration of AWS Organizations policies to the member account where you are deploying the pipeline. For instructions, see Create a resource-based delegation policy with AWS Organizations. For a sample policy, see *Sample resource-based delegation policy* in	AWS administrator

Task	Description	Skills required
	the Additional information section of this pattern.	
(Optional) Enable the foundation model.	If you want to generate summaries of the policy changes, enable access to the Anthropic Claude 3.5 Sonnet foundation model in Amazon Bedrock in the AWS account where you are deploying the pipeline. For instructions, see Add or remove access to Amazon Bedrock foundation models.	General AWS

Deploy the resources for the pipeline

Task	Description	Skills required
Clone the repository.	Enter the following command to clone the organizations-policy-pipeline repository from GitHub: `git clone https://github.com/aws-samples/organizations-policy-pipeline.git`	DevOps engineer
Define your deployment method.	1. In the cloned repository, open the `variables.tf` file. 2. For `project_name`, enter the prefix that you want	DevOps engineer

Task	Description	Skills required
	to apply to names of the deployed resources. 3. For `provider_type` , enter the provider of the remote repository. Valid values are provided in the file. 4. For `full_repository_name` , enter the name of the remote repository. 5. For `branch_name` , enter the name of the Git branch that you will use to deploy policies. A push or merge in this branch starts the pipeline. Typically, this is the main branch. 6. For `terraform_version` , enter the version of Terraform that you are using. 7. For `enable_bedrock` , enter `true` if you want Amazon Bedrock to summarize the changes. Enter `false` if you do not want to generate a summary of the changes. 8. For `tags`, enter the key-value pairs that you want to assign as tags to the deployed resources.	

Task	Description	Skills required
	9. Save and close the `variables.tf` file.	

Deploy the pipeline.	1. Enter the following command to create a plan and review the changes:
	`terraform plan`
	2. Enter the following command to apply the plan and create the pipeline infrastructure:
	`terraform apply`
		DevOps engineer, Terraform

Connect the remote repository.	In the previous step, Terraform created an CodeConnections connection to the third-party repository. In the [AWS Developer Tools console](), change the status of the connection from `PENDING` to `AVAILABLE`. For instructions, see [Update a pending connection]().	AWS DevOps

Task	Description	Skills required
Subscribe to the Amazon SNS topic.	Terraform created an Amazon SNS topic. Subscribe an endpoint to the topic and confirm the subscription so that the approvers receive notifications about pending approval actions in the pipeline. For instructions, see [Creating a subscription to an Amazon SNS topic]().	General AWS

Define your guardrails and policies

Task	Description	Skills required
Populate the remote repository.	From the cloned repository, copy the contents of the `sample-repository` folder to your remote repository. This includes the `environments` , `rcp-management` , `scp-management` , and `utils` folders.	DevOps engineer
Define your environments.	1. In the `environments` folder, open the `environments.json` file. This is the file where you define the target AWS accounts and OUs for your RCPs and SCPs. 2. Delete the example environments.	DevOps engineer

Task	Description	Skills required
	3. Add your target environments in the following format: `[ { "ID": "<environment-name>", "Target": [ "<ou-name>:<ou-id>", "<account-name>:<account-id>" ] } ]` Where: • `<environment-name>` is the name you assign to the group of OUs and AWS accounts. You can use this name in the manifest file to define where you want to apply your policies. • `<ou-name>` is the name of the target OU. • `<ou-id>` is the ID of the target OU. • `<account-name>` is the name of the target AWS account.	

Task	Description	Skills required
	• `<account-id>` is the ID of the target AWS account. For examples, see the source code repository. 4. Save and close the `environments.json` file.	

Task	Description	Skills required
Define your guardrails.	1. Navigate to the `rcp-management/guardrails` folder in your remote repository. This is the folder where you define the guardrails for your RCP manifest file. Each guardrail must in an individual file. Guardrail files can contain one or more statements. **Note** You can use the same guardrail in multiple statements in the manifest files for SCPs and RCPs. If you modify the guardrail, any policies that include this guardrail are affected. 2. Delete any example guardrails that were copied from the source code repository. 3. Create a new .json file and give it a descriptive name. 4. Open the .json file you created.	DevOps engineer

Task	Description	Skills required
	5. Define the guardrail in the following format: ```[{ "Sid": "<guardrail-name>", "Effect": "<effect-value>", "Action": ["<action-name>"], "Resource": "<resource-arn>", "Condition": { "<condition-operator>": { "<condition-key>": ["<condition-value>"] } } }]``` Where: • `<guardrail-name>` is a unique name for the guardrail. This name cannot be used for any other guardrails. • `<effect-value>` must be `Allow` or `Deny`.	

Task	Description	Skills required
	For more information, see Effect. • `<action-name>` must be a valid name of an action that the service supports. For more information, see Action. • `<resource-arn>` is the Amazon Resource Name (ARN) of the resource that the guardrail applies to. You can also use wildcard characters, such as `*` or `?`. For more information, see Resource. • `<condition-operator>` is a valid condition operator. For more information, see Condition operators. • `<condition-key>` is a valid global condition context key or a service-specific context key. For more information, see Condition. • `<condition-value>` is the specific value used in a condition to evaluate whether a guardrail applies. For more information, see Condition.	

Task	Description	Skills required
	For example RCP guardrails, see the [source code repository](). 6. Save and close the .json file. 7. Repeat these steps to create as many RCP guardrails as needed. 8. Repeat these steps in the `scp-management/guardrails` folder to create as many guardrails as you need for your SCPs. For example SCP guardrails, see the [source code repository]().	

Task	Description	Skills required
Define your policies.	1. Navigate to the `rcp-management/policies` folder in your remote repository. This is the folder where you define full policies for your RCP manifest file. Each policy must be an individual file. **Note** If you modify a policy in this folder, the policy changes affect any accounts or OUs that this policy is applied to, as defined in the manifest file. 2. Delete any example policies that were copied from the source code repository. 3. Create a new .json file and give it a descriptive name. 4. Open the .json file you created. 5. Define the RCP. For example RCPs, see the source code repository or see Resource control policy examples in the AWS	DevOps engineer

Task	Description	Skills required
	Organizations documentation. 6. Save and close the .json file. 7. Repeat these steps to create as many RCPs as needed. 8. Repeat these steps in the `scp-management/policies` folder to create as many SCPs as needed. For example SCPs, see the [source code repository](#) or see [Service control policy examples](#) in the AWS Organizations documentation.	

Use the manifest file to manage the policies

Task	Description	Skills required
Configure the manifest files.	1. In the `rcp-management` folder, open the `rcp-management.json` file. This is the file where you define which RCP guardrails and full RCPs apply to your target environments. For an example of this file, see the [source code repository](#).	DevOps engineer

Task	Description	Skills required
	2. Delete the example statement. 3. Add a new statement in the following format: `[ { "SID": "<statement-name>", "Target": { "Type": "<target-type>", "ID": "<target-name>" }, "Guardrails": [ "<guardrail-name>" ], "Policy": "<policy-name>", "Comments": "<comment-text>" } ]` Where: • `<statement-name>` is a unique name for the statement. • `<target-type>` is the type of target where you want to apply the policy. Valid values are `Account`, `OU`, `Environment` , or `Tag`.	

Task	Description	Skills required
	• `<target-name>` is the identifier of the target where you want to apply the policy. Enter one of the following: ◦ For an AWS account, enter the identifier as `<account-name>:<account-id>` . ◦ For an OU, enter the identifier as `<OU-name>:<ou-id>` . ◦ For an environment, enter the unique name that you defined in the `environments.json` file. ◦ For a tag, enter the key-value pair as `<tag-key>:<tag-value>` . • `<guardrail-name>` is the unique name of the RCP guardrail that you defined in the `rcp-management/guardrails` folder. You can add multiple guardrails in this element. You can leave this field empty if you	

Task	Description	Skills required
	do not want to apply a guardrail. • `<policy-name>` is the unique name of the RCP that you defined in the `rcp-management/policies` folder. You can add only one policy in this element. You can leave this field empty if you do not want to apply a policy. • `<comment-text>` is a description that you can enter for documentation purposes. This field is not used during pipeline processing. You can leave this field empty if you do not want to add a comment. 4. Repeat these steps to add as many statements as necessary to configure RCPs for your organization. 5. Save and close the `rcp-management.json` file. 6. In the `scp-management` folder, repeat these steps in the `scp-management.json` file. This is the file where you define which SCP guardrails and full SCPs apply to your	

Task	Description	Skills required
	target environments. For an example of this file, see the source code repository.	
Start the pipeline.	Commit and push the changes to the branch of the remote repository that you defined in the `variables.tf` file. Typically, this is the `main` branch. The CI/CD pipeline automatically starts. If there are any pipeline errors, see the Troubleshooting section of this pattern.	DevOps engineer
Approve the changes.	When the `Validate-Plan` CodeBuild project is complete, the policy approvers receive a notification through the Amazon SNS topic that you previously configured. Do the following: 1. Open the notification message. 2. If available, review the summary of policy changes. 3. Follow the instructions in Approve or reject an approval action in CodePipeline.	General AWS, Policy approver

Task	Description	Skills required
Validate the deployment.	1. Sign in to the [AWS Organizations console](https://console.aws.amazon.com/organizations/) in the account that is the delegated administrator for AWS Organizations. 2. On the **Service control policies** page, confirm that the SCPs that you created are listed. 3. Choose an SCP that is managed through the pipeline and confirm that it applies to the intended targets. 4. On the **Resource control policies** page, confirm that the RCPs that you created are listed. 5. Choose an RCP that is managed through the pipeline and confirm that it applies to the intended targets.	General AWS

Troubleshooting

Issue	Solution
Manifest file errors in the `Validate-Plan` phase of the pipeline	A "Pipeline errors in the Validation & Plan phase for manifest files" message appears in the pipeline output if there are any errors in the `scp-management.json` or `rcp-management.json` files. Possible errors

Issue	Solution
	include an incorrect environment name, duplicated SIDs, or invalid fields or values. Do the following: 1. Follow the instructions in [View build details in AWS CodeBuild](). 2. In the build log, find the validation error. The error provides more information about what caused the build to fail. 3. Update the corresponding .json file. 4. Commit and push the updated file to the remote repository. The pipeline restarts. 5. Monitor the status to confirm that the validation error is resolved.

Issue	Solution
IAM Access Analyzer findings in the `Validate-Plan` phase of the pipeline	A "Findings in IAM Access Analyzer during Validation & Plan phase" message appears in the pipeline output if there are any errors in the guardrail or policy definitions. This pattern uses IAM Access Analyzer to validate the final policy. Do the following: 1. Follow the instructions in View build details in AWS CodeBuild. 2. In the build log, find the IAM Access Analyzer validation error. The error provides more information about what caused the build to fail. For more information about the finding types, see IAM policy validation check reference. 3. Update the corresponding .json file for the guardrail or policy. 4. Commit and push the updated file to the remote repository. The pipeline restarts. 5. Monitor the status to confirm that the validation error is resolved.

Related resources

- JSON policy element reference (IAM documentation)
- Resource control policies (AWS Organizations documentation)
- Service control policies (AWS Organizations documentation)
- Add or remove access to Amazon Bedrock foundation models (Amazon Bedrock documentation)
- Approve or reject an approval action in CodePipeline (CodePipeline documentation)

Additional information

Sample resource-based delegation policy

The following is a sample resource-based delegation policy for AWS Organizations. It allows the delegated administer account to manage SCPs and RCPs for the organization. In the following sample policy, replace <MEMBER_ACCOUNT_ID> with the ID of the account where you are deploying the policy management pipeline.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DelegationToAudit",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<MEMBER_ACCOUNT_ID>:root"
      },
      "Action": [
        "organizations:ListTargetsForPolicy",
        "organizations:CreatePolicy",
        "organizations:DeletePolicy",
        "organizations:AttachPolicy",
        "organizations:DetachPolicy",
        "organizations:DisablePolicyType",
        "organizations:EnablePolicyType",
        "organizations:UpdatePolicy",
        "organizations:DescribeEffectivePolicy",
        "organizations:DescribePolicy",
        "organizations:DescribeResourcePolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

Migrating existing policies to the pipeline

If you have existing SCPs or RCPs that you want to migrate and manage through this pipeline, you can use the Python scripts in the `sample-repository/utils` folder of the code repository. These scripts include:

- `check-if-scp-exists-in-env.py` – This script checks whether a specified policy applies to any targets in a specific environment, which you define in the `environments.json` file. Enter the following command to run this script:

```
python3 check-if-scp-exists-in-env.py \
   --policy-type <POLICY_TYPE> \
   --policy-name <POLICY_NAME> \
   --env-id <ENV_ID>
```

  Replace the following in this command:

  - `<POLICY_TYPE>` is `scp` or `rcp`
  - `<POLICY_NAME>` is the name of the SCP or RCP
  - `<ENV_ID>` is the ID of the environment that you defined in the `environments.json` file

- `create-environments.py` – This script creates an environments.json file based on the current SCPs and RCPs in your environment. It excludes policies deployed through AWS Control Tower. Enter the following command to run this script, where `<POLICY_TYPE>` is `scp` or `rcp`:

```
python create-environments.py --policy-type <POLICY_TYPE>
```

- `verify-policies-capacity.py` – This script checks each environment that you define to determine how much capacity remains for each AWS Organizations policy-related quota. You define the environments to check in the in `environments.json` file. Enter the following command to run this script, where `<POLICY_TYPE>` is `scp` or `rcp`:

```
python verify-policies-capacity.py --policy-type <POLICY_TYPE>
```

Manage AWS IAM Identity Center permission sets as code by using AWS CodePipeline

Andre Cavalcante and Claison Amorim, Amazon Web Services

Summary

AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to all of your AWS accounts and applications. You can create and manage user identities in IAM Identity Center, or you can connect an existing identity source, such as a Microsoft Active Directory domain or an external identity provider (IdP). IAM Identity Center provides a unified administration experience to define, customize, and assign fine-grained access to your AWS environment by using permission sets. Permission sets apply to the federated users and groups from your IAM Identity Center identity store or your external IdP.

This pattern helps you to manage IAM Identity Center permission sets as code in your multi-account environment that is managed as an organization in AWS Organizations. With this pattern, you can achieve the following:

- Create, delete, and update permission sets
- Create, update, or delete permission set assignments to target AWS accounts, organizational units (OUs), or your organization root.

To manage IAM Identity Center permissions and assignments as code, this solution deploys a continuous integration and continuous delivery (CI/CD) pipeline that uses AWS CodeBuild and AWS CodePipeline. You manage the permission sets and assignments in JSON templates that you store in a remote repository. When Amazon EventBridge rules detect a change to the repository or detect modifications to the accounts in the target OU, it starts an AWS Lambda function. The Lambda function initiates the CI/CD pipeline that updates the permission sets and assignments in IAM Identity Center.

Prerequisites and limitations

Prerequisites

- A multi-account environment managed as an organization in AWS Organizations. For more information, see Creating an organization.

- IAM Identity Center, enabled and configured with an identity source. For more information, see Getting Started in the IAM Identity Center documentation.
- A member account that is registered as the delegated administrator for the following AWS services:
  - IAM Identity Center – For instructions, see Register a member account in the IAM Identity Center documentation.
  - AWS Organizations – For instructions, see Delegated administrator for AWS Organizations. This account must have permissions to list and describe accounts and OUs.

> **Note**
>
> You must use the same account as the delegated administrator for both services.

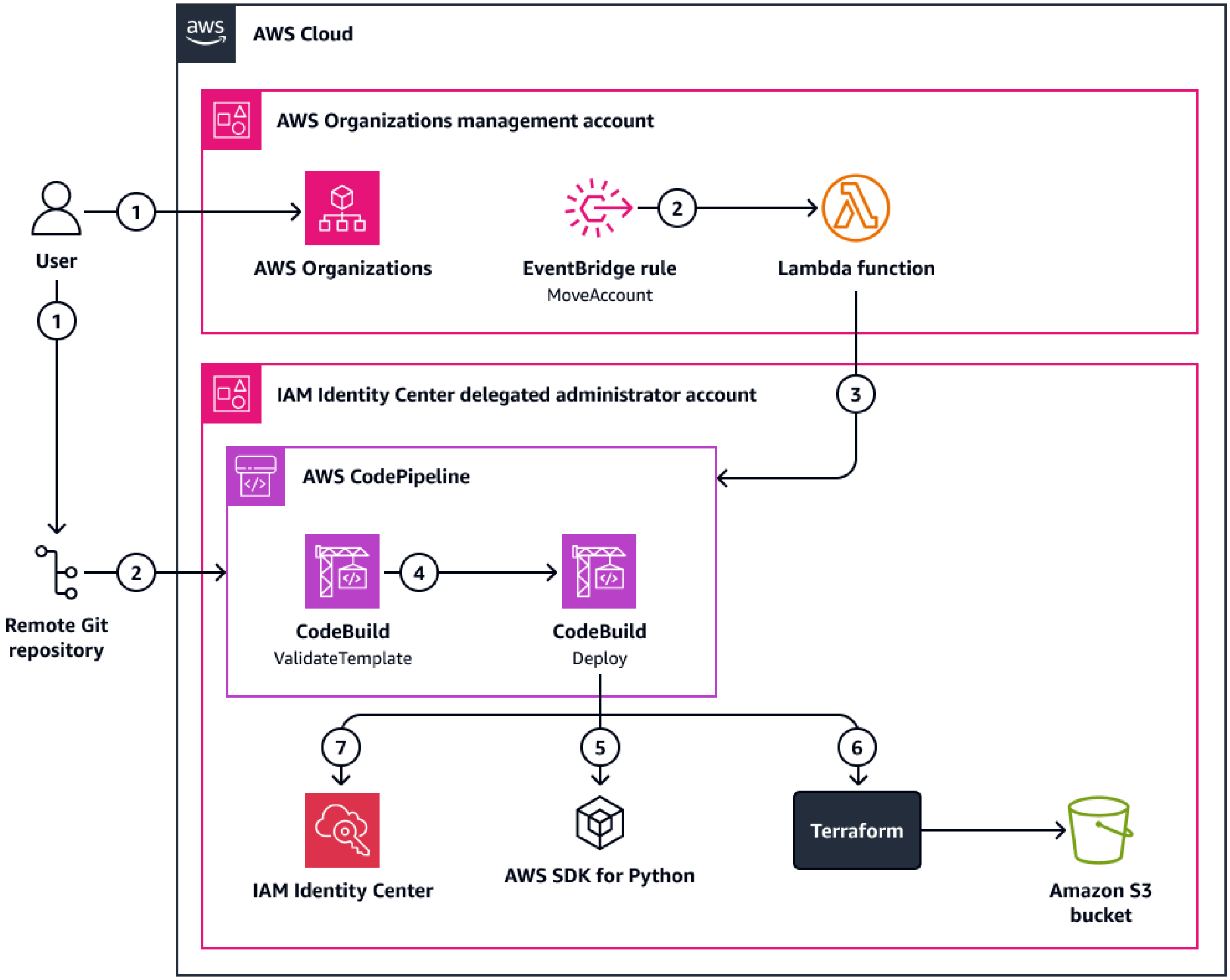
- Permissions to deploy AWS CloudFormation stacks in the IAM Identity Center delegated administrator account and in the organization's management account. For more information, see Controlling access in the CloudFormation documentation.
- An Amazon Simple Storage Service (Amazon S3) bucket in the IAM Identity Center delegated administrator account. You upload the artifact code into this bucket. For instructions, see Creating a bucket in the Amazon S3 documentation.
- The account ID of the organization's management account. For instructions, see Finding your AWS account ID.
- A repository in your source code host, such as GitHub.

Limitations

- This pattern cannot be used to manage or assign permission sets for single-account environments or for accounts that are not managed as an organization in AWS Organizations.
- Permission set names, assignment IDs, and IAM Identity Center principal types and IDs cannot be modified after deployment.
- This pattern helps you create and manage custom permissions. You cannot use this pattern to manage or assign predefined permissions.
- This pattern cannot be used to manage a permission set for the organization's management account.

Architecture

Target architecture



The diagram shows the following workflow:

1. A user makes one of the following changes:
   - Commits one or more changes to the remote repository, such as GitHub
   - Modifies the accounts in the OU in AWS Organizations
2. If the user committed a change to the remote repository to the main branch, the pipeline starts.

   If the user modified the accounts in the OU, then the `MoveAccount` EventBridge rule detects the change and starts a Lambda function in the organization's management account.

3. The initiated Lambda function starts the CI/CD pipeline in CodePipeline.
4. CodePipeline starts the `TemplateValidation` CodeBuild project. The `TemplateValidation` CodeBuild project uses a Python script in the remote repository to validate the permission set templates. CodeBuild validates the following:
   - The permission set names are unique.
   - The assignment statement IDs (`Sid`) are unique.
   - Policy definitions in the `CustomPolicy` parameter and valid. (This validation uses AWS Identity and Access Management Access Analyzer.)
   - The Amazon Resource Names (ARNs) of the managed policies are valid.
5. The `PermissionSet` action group in the `Deploy` CodeBuild project uses AWS SDK for Python (Boto3) to delete, create, or update the permission sets in IAM Identity Center. Only permission sets with the `SSOPipeline:true` tag are affected. All permission sets that are managed through this pipeline have this tag.
6. The `Assignments` action group in the `Deploy` CodeBuild project uses Terraform to delete, create, or update the assignments in IAM Identity Center. The Terraform backend state files are stored in an Amazon S3 bucket in the same account.
7. CodeBuild updates the permissions sets and assignments in IAM Identity Center.

Automation and scale

Because all new accounts in a multi-account environment are moved to a specific organizational unit in AWS Organizations, this solution automatically runs and grants the required permission sets to all accounts that you specify in the assignment templates. No additional automations or scaling actions are necessary.

In large environments, the number of API requests to IAM Identity Center might cause this solution to run more slowly. Terraform and Boto3 automatically manage throttling to minimize any performance degradation.

Tools

AWS services

- [AWS CloudFormation](#) helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and AWS Regions.

- AWS CodeBuild is a fully managed build service that helps you compile source code, run unit tests, and produce artifacts that are ready to deploy.
- AWS CodePipeline helps you quickly model and configure the different stages of a software release and automate the steps required to release software changes continuously.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to all of your AWS accounts and cloud applications.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

Code repository

The code for this pattern is available in the aws-iam-identity-center-pipeline repository. The templates folder in the repository includes sample templates for both permission sets and assignments. It also includes AWS CloudFormation templates for deploying the CI/CD pipeline and AWS resources in the target accounts.

Best practices

- Before you start modifying the permission set and assignment templates, we recommend that you plan permission sets for your organization. Consider what the permissions should be, which accounts or OUs the permission set should apply to, and which IAM Identity Center principals (users or groups) should be affected by the permission set. Permission set names, association IDs, and IAM Identity Center principal types and IDs cannot be modified after deployment.
- Adhere to the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices in the AWS Identity and Access Management (IAM) documentation.

Epics

Plan permission sets and assignments

Task	Description	Skills required
Clone the repository.	In a bash shell, enter the following command. This clones the [aws-iam-identity-center-pipeline](https://github.com/aws-samples/aws-iam-identity-center-pipeline) repository from GitHub. `git clone https://github.com/aws-samples/aws-iam-identity-center-pipeline.git`	DevOps engineer
Define the permission sets.	1. In the cloned repository, navigate to the `templates/permissionsets` folder, and then open one of the available templates. 2. In the `Name` parameter, enter a name for the permission set. This value must be unique, and it cannot be changed after deployment. 3. In the `Description` parameter, briefly describe the permission set, such as its use case. 4. In the `SessionDuration` parameter, specify the length of time that a user can be signed in	DevOps engineer

Task	Description	Skills required
	to an AWS account. Use ISO-8601 duration format (Wikipedia), such as PT4H for 4 hours. If no value is defined, the default in IAM Identity Center is 1 hour. 5. In the RelayState parameter, specify the URL that provides quick access to the console that is most appropriate for the user's role. 6. Customize the policies in the permission set. All of the following parameter s are optional and can be modified after deploymen t. You must use at least one of the parameters in order to define the policies in the permission set: • In the ManagedPo licies parameter, enter the ARNs of any AWS managed policies that you want to assign. • In the CustomerM anagedPolicies parameter, enter the names of any customer managed policies that you want to assign. Do not use the ARN.	

Task	Description	Skills required

	• In the `PermissionBoundary` parameter, do the following to assign a [permissions boundary](#):
	  • If you're using an AWS managed policy as a permissions boundary, in `PolicyType`, enter `AWS`, and in `Policy`, enter the ARN of the policy.
	  • If you're using a customer managed policy as a permissions boundary, in `PolicyType`, enter `Customer`, and in `Policy`, enter the name of the policy. Do not use the ARN.
	• In the `CustomPolicy` parameter, define any custom, JSON-formatted policies that you want to assign. For more information about the JSON policy structure, see [Overview of JSON policies](#).
	7. Save and close the permission set template. We recommend that you save the file with a name	

Task	Description	Skills required
	that matches the name of the permission set. 8. Repeat this process to create as many permissions sets as needed for your organization, and delete any sample templates that aren't required.	

Task	Description	Skills required
Define the assignments.	1. In the cloned repository, navigate to the `templates/assignments` folder, and then open `iam-identitycenter-assigments.json` . This file describes how you want to assign the permission sets to AWS accounts or OUs. 2. In the `SID` parameter, enter an identifier for the assignment. This value must be unique, and it cannot be modified after deployment. 3. In the `Target` parameter, define the accounts or organizations where you want to apply the permission set. Valid values are account IDs, OUs, or `root`. `root` assigns the permission set to all member accounts in the organization, excluding the management account. Enter the values in double quotation marks, and separate multiple values with commas. Account IDs and OUs should follow the pattern: `{{account_name}}:{{account_`	DevOps engineer

Task	Description	Skills required
	`id}}` or `{{ou_name}}:{{ou_id}}`. If you want to assign permissions recursively to nested OUs, use the OU pattern with a wildcard at the end. Example: `{{ou_name}}:{{ou_id}}:*` 4. In the `PrincipalType` parameter, enter the type of IAM Identity Center principal that will be affected by the permission set. Valid values are `USER` or `GROUP`. This value cannot be modified after deployment. 5. In the `PrincipalID` parameter, enter the name of the user or group in the IAM Identity Center identity store that will be affected by the permission set. This value cannot be modified after deployment. 6. In the `PermissionSetName` parameter, enter the name of the permission set that you want to assign. 7. Repeat steps 2–6 to create as many assignments as needed in this file. Typically, there is one	

Task	Description	Skills required
	assignment for each permission set. Delete any sample assignments that aren't required. 8. Save and close the `iam-identitycenter-assigments.json` file.	

Deploy the permission sets and assignments

Task	Description	Skills required
Deploy resources in the IAM Identity Center delegated administrator account.	1. In the IAM Identity Center delegated administrator account, open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/). 2. Deploy the `iam-identitycenter-pipeline.yaml` template. Give the stack a clear and descriptive name, and update the parameters as instructed. For instructions, see [Creating a stack](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html) in the CloudFormation documentation.	DevOps engineer
Deploy resources in the AWS Organizations management account.	1. Sign in to the organization's management account. 2. Open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/). 3. In the navigation bar, choose the name of the currently displayed AWS	DevOps engineer

Task	Description	Skills required
	Region. Then choose the `us-east-1` Region. This Region is required so that the `MoveAccount` EventBridge rule can detect AWS CloudTrail events associated with organization changes. 4. Deploy the `iam-identitycenter-organization` template. Give the stack a clear and descriptive name, and update the parameters as instructed. For instructions, see [Creating a stack](#) in the CloudFormation documentation.	
Finish the remote repository setup.	Change the status of the AWS CodeConnections connection from `PENDING` to `AVAILABLE`. This connection was created when you deployed the CloudFormation stack. For instructions, see [Update a pending connection](#) in the CodeConnections documentation.	DevOps engineer

Task	Description	Skills required
Upload files to the remote repository.	Upload all of files you have downloaded from the `aws-samples` repository and edited in previous steps to the remote repository. Changes to the `main` branch start the pipeline, which creates or updates the permission sets and assignments.	DevOps engineer

Updating the permission sets and assignments

Task	Description	Skills required
Update the permission sets and assignments.	When the `MoveAccount` Amazon EventBridge rule detects modifications to the accounts in the organization, the CI/CD pipeline automatically starts and updates the permission sets. For example, if you add an account to an OU specified in the assignments JSON file, then the CI/CD pipeline will apply the permission set to the new account. If you want to modify the deployed permission sets and assignments, update the JSON files and then commit them to the remote repository.	DevOps engineer

Task	Description	Skills required
	Note the following when using the CI/CD pipeline to manage previously deployed permission sets and associati ons: • If you change the name of a permission set, the CI/CD pipeline deletes the original permission set and creates a new one. • This pipeline manages only permission sets that have the `SSOPipeline:true` tag. • You can have multiple permission set and assignment templates in the same folder in the repository. • If you delete a template, the pipeline deletes the assignment or permission set. • If you delete an entire assignment JSON block, the pipeline deletes the assignment from IAM Identity Center. • You can't delete a permissio n set that is assigned to an AWS account. First, you must unassign the permission set.	

Troubleshooting

Issue	Solution
Access denied errors	Confirm that you have the permissions required to deploy the CloudFormation templates and the resources defined within them. For more information, see Controlling access in the CloudFormation documentation.
Pipeline errors in the validation phase	This error appears if there are any errors in the permission set or assignment templates. 1. In CodeBuild, view the build details. 2. In the build log, find the validation error that provides more information about what caused the build to fail. 3. Update the permission set or assignment templates, and then commit them to the repository. 4. The CI/CD pipeline restarts the CodeBuild project. Monitor the status to confirm that the validation error is resolved.

Related resources

- Permission sets (IAM Identity Center documentation)

Manage credentials using AWS Secrets Manager

Durga Prasad Cheepuri, Amazon Web Services

Summary

This pattern walks you through using AWS Secrets Manager to dynamically fetch database credentials for a Java Spring application.

In the past, when you created a custom application that retrieves information from a database, you typically had to embed the credentials (the secret) for accessing the database directly in the application. When it was time to rotate the credentials, you had to invest time to update the application to use the new credentials, and then distribute the updated application. If you had multiple applications that shared credentials and you missed updating one of them, the application would fail. Because of this risk, many users chose not to regularly rotate their credentials, which effectively substituted one risk for another.

Secrets Manager enables you to replace hard-coded credentials in your code (including passwords) with an API call to retrieve the secret programmatically. This helps ensure that the secret can't be compromised by someone who is examining your code, because the secret simply isn't there. You can also configure Secrets Manager to automatically rotate the secret according to a schedule that you specify. This enables you to replace long-term secrets with short-term ones, which helps significantly reduce the risk of compromise. For more information, see the AWS Secrets Manager documentation.

Prerequisites and limitations

Prerequisites

- An AWS account with access to Secrets Manager
- A Java Spring application

Architecture

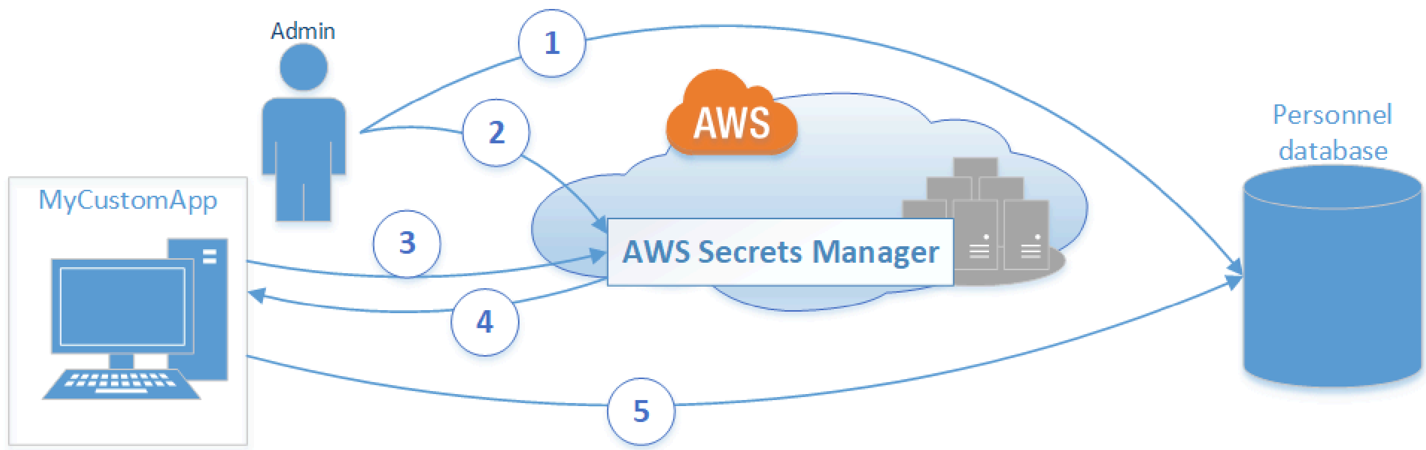
Source technology stack

- A Java Spring application with code that accesses a database, with DB credentials managed from the application.properties file.

Target technology stack

- A Java Spring application with code that accesses a database, with DB credentials managed in Secrets Manager. The application.properties file holds the secrets to Secrets Manager.

Secrets Manager integration with an application



Tools

- **Secrets Manager** – AWS Secrets Manager is an AWS service that makes it easier for you to manage secrets. Secrets can be database credentials, passwords, third-party API keys, and even arbitrary text. You can store and control access to these secrets centrally by using the Secrets Manager console, the Secrets Manager command-line interface (CLI), or the Secrets Manager API and SDKs.

Epics

Store secret in Secrets Manager

Task	Description	Skills required
Store the DB credentials as a secret in Secrets Manager.	Store Amazon Relational Database Service (Amazon RDS) or other DB credentials as a secret in Secrets Manager by following the steps in	Sys Admin

Task	Description	Skills required
	Creating a secret in the Secrets Manager documenta tion.	
Set permissions for the Spring application to access Secrets Manager.	Set the appropriate permissio ns based on how the Java Spring application uses Secrets Manager. To control access to the secret, create a policy based on the information provided in the Secrets Manager documenta tion, in the sections Using identity-based policies (IAM Policies) and ABAC for Secrets Manager and Using resource-based policies for Secrets Manager. Follow the steps in the section Retrieving the secret value in the Secrets Manager documentation.	Sys Admin

Update the Spring application

Task	Description	Skills required
Add JAR dependencies to use Secrets Manager.	See the *Additional informati on* section for details.	Java developer
Add the details of the secret to the Spring application.	Update the application.proper ties file with the secret name, endpoints, and AWS Region. For an example, see the *Additional information* section.	Java developer

Task	Description	Skills required
Update the DB credentials retrieval code in Java.	In the application, update the Java code that fetches the DB credentials to fetch those details from Secrets Manager. For example code, see the *Additional information* section.	Java developer

Related resources

- AWS Secrets Manager documentation
- Using identity-based policies (IAM Policies) and ABAC for Secrets Manager
- Using resource-based policies for Secrets Manager
- Sample code

Additional information

Adding JAR dependencies for using Secrets Manager

Maven:

```
<groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-secretsmanager</artifactId>
    <version>1.11. 355 </version>
```

Gradle:

```
compile group: 'com.amazonaws', name: 'aws-java-sdk-secretsmanager', version:
 '1.11.355'
```

Updating the application.properties file with the details of the secret

```
spring.aws.secretsmanager.secretName=postgres-local
spring.aws.secretsmanager.endpoint=secretsmanager.us-east-1.amazonaws.com
```

```
spring.aws.secretsmanager.region=us-east-1
```

Updating the DB credentials retrieval code in Java

```
String  secretName  =  env.getProperty("spring.aws.secretsmanager.secretName");
String  endpoints  =  env.getProperty("spring.aws.secretsmanager.endpoint");
String  AWS Region  =  env.getProperty("spring.aws.secretsmanager.region");
AwsClientBuilder.EndpointConfiguration  config  =  new
  AwsClientBuilder.EndpointConfiguration(endpoints, AWS Region);
AWSSecretsManagerClientBuilder  clientBuilder  =
  AWSSecretsManagerClientBuilder.standard();
clientBuilder.setEndpointConfiguration(config);
AWSSecretsManager  client  =  clientBuilder.build();

ObjectMapper  objectMapper  =  new  ObjectMapper();

JsonNode  secretsJson  =  null;

ByteBuffer  binarySecretData;

GetSecretValueRequest  getSecretValueRequest  =  new
  GetSecretValueRequest().withSecretId(secretName);

GetSecretValueResult  getSecretValueResponse  =  null;

try  {
     getSecretValueResponse  =  client.getSecretValue(getSecretValueRequest);
    }

catch  (ResourceNotFoundException  e)  {
     log.error("The requested secret "  +  secretName  +  " was not found");
    }

catch  (InvalidRequestException  e)  {
     log.error("The request was invalid due to: "  +  e.getMessage());
     }

catch  (InvalidParameterException  e)  {
     log.error("The request had invalid params: "  +  e.getMessage());
     }
if  (getSecretValueResponse  ==  null)  {
     return  null;
```

```
    }  // Decrypted secret using the associated KMS key // Depending on whether the
 secret was a string or binary, one of these fields will be populated


String secret = getSecretValueResponse.getSecretString();

if (secret != null) {
    try {
                secretsJson  =  objectMapper.readTree(secret);
          }

    catch  (IOException  e)  {
                log.error("Exception while retrieving secret values: "  +
  e.getMessage());
          }
}

else  {
    log.error("The Secret String returned is null");

    return null;

    }
    String  host  =  secretsJson.get("host").textValue();
    String  port  =  secretsJson.get("port").textValue();
    String  dbname  =  secretsJson.get("dbname").textValue();
    String  username  =  secretsJson.get("username").textValue();
    String  password  =  secretsJson.get("password").textValue();
}
```

Monitor ElastiCache clusters for security groups

Susanne Kangnoh and Archit Mathur, Amazon Web Services

Summary

Amazon ElastiCache is an Amazon Web Services (AWS) service that provides a high-performance, scalable, and cost-effective caching solution for distributing an in-memory data store or cache environment in the cloud. It retrieves data from high-throughput and low-latency, in-memory data stores. This functionality makes it a popular choice for real-time use cases such as caching, session stores, gaming, geo-spatial services, real-time analytics, and queuing. ElastiCache offers Redis and Memcached data stores, both of which provide sub-millisecond response times.

A *security group* acts as a virtual firewall for your ElastiCache instances by controlling inbound and outbound traffic. Security groups act at the instance level, not at the subnet level. For each security group, you add one set of rules that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic. You can specify allow rules but not deny rules.

This pattern provides a security control that monitors for API calls and generates an event in Amazon CloudWatch Events on the **CreateReplicationGroup**, **CreateCacheCluster**, **ModifyCacheCluster**, and **ModifyReplicationGroup** operations. This event calls an AWS Lambda function, which runs a Python script. The function gets the replication group ID from the event JSON input, and performs the following checks to determine whether there's a security violation:

- Checks if the security group of the cluster matches the security group that's configured in the Lambda function.
- If the security group of the cluster doesn't match, the function sends a violation message to an email address you provide, by using an Amazon Simple Notification Service (Amazon SNS) notification.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An Amazon Simple Storage Service (Amazon S3) bucket to upload the provided Lambda code.
- An email address where you would like to receive violation notifications.

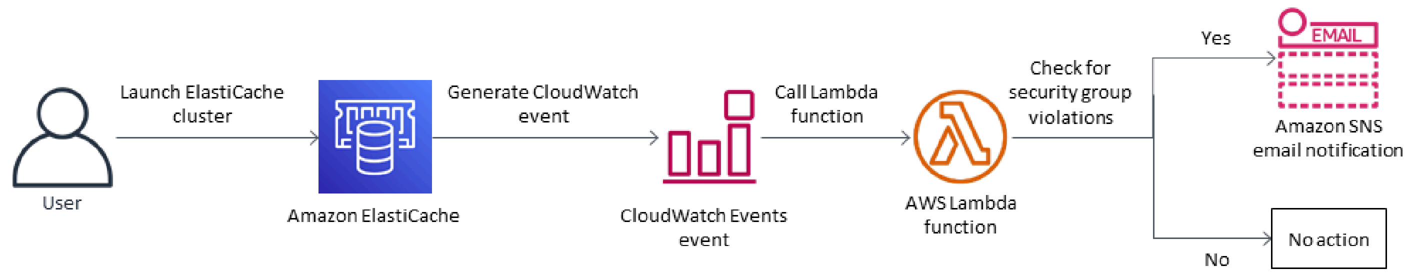
- ElastiCache logging enabled, for access to all the API logs.

Limitations

- This detective control is regional and must be deployed in each AWS Region that you want to monitor.
- The control supports replication groups that are running in a virtual private cloud (VPC).

Architecture

Workflow architecture



Automation and scale

- If you are using AWS Organizations, you can use [AWS CloudFormation StackSets] to deploy this template into multiple accounts that you want to monitor.

Tools

AWS services

- [Amazon ElastiCache] makes it easy to set up, manage, and scale distributed in-memory cache environments in the AWS Cloud. It provides a high performance, resizable, and cost-effective in-memory cache, while removing complexity associated with deploying and managing a distributed cache environment. ElastiCache works with both the Redis and Memcached engines.
- [AWS CloudFormation] helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.

- [Amazon CloudWatch Events] delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- [AWS Lambda] is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- [Amazon Simple Notification Service (Amazon SNS)] coordinates and manages the sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes an attachment with two files:

- `ElastiCacheAllowedSecurityGroup.zip` is a compressed file that includes the security control (Lambda code).
- `ElastiCacheAllowedSecurityGroup.yml` is a CloudFormation template that deploys the security control.

See the *Epics* section for information about how to use these files.

Epics

Deploy the security control

Task	Description	Skills required
Upload the code to an S3 bucket.	Create a new S3 bucket or use an existing S3 bucket to upload the attached `ElastiCacheAllowedSecurityGroup.zip` file (Lambda code). This bucket must be in the same AWS	Cloud architect

Task	Description	Skills required
	Region as the resources that you want to evaluate.	
Deploy the CloudFormation template.	Open the CloudFormation console in the same AWS Region as the S3 bucket, and deploy the `ElasticCacheAllowedSecurityControl.yml` file that's provided in the attachment. In the next epic, provide values for the template parameters.	Cloud architect

Complete the parameters in the CloudFormation template

Task	Description	Skills required
Provide the S3 bucket name.	Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the resource that will be evaluated.	Cloud architect
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `ElasticCacheAllowedSecurityGroup.zip` or `controls/ElasticCa`	Cloud architect

Task	Description	Skills required
	cheAllowedSecurity Group.zip).	
Provide an email address.	Provide an active email address where you want to receive violation notifications.	Cloud architect
Specify a logging level.	Specify the logging level and verbosity. `Info` designates detailed informational messages on the application's progress and should be used only for debugging. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the email subscription.	When the CloudFormation template deploys successfully, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.	Cloud architect

Related resources

- [Creating a stack on the AWS CloudFormation console] (AWS CloudFormation documentation)
- [Amazon VPCs and ElastiCache security] (Amazon ElastiCache documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Monitor Amazon EMR clusters for in-transit encryption at launch

Susanne Kangnoh, Amazon Web Services

Summary

This pattern provides a security control that monitors Amazon EMR clusters at launch and sends an alert if in-transit encryption hasn't been enabled.

Amazon EMR is a web service that makes it easy for you to run big data frameworks, such as Apache Hadoop, to process and analyze data. Amazon EMR enables you to process vast amounts of data in a cost-effective way by running mapping and reducing steps in parallel.

Data encryption prevents unauthorized users from accessing or reading data at rest or data in transit. *Data at rest* refers to data that is stored in media such as a local file system on each node, Hadoop Distributed File System (HDFS), or the EMR File System (EMRFS) through Amazon Simple Storage Service (Amazon S3). *Data in transit* refers to data that travels the network and is in flight between jobs. In-transit encryption supports open-source encryption features for Apache Spark, Apache TEZ, Apache Hadoop, Apache HBase, and Presto. You enable encryption by creating a security configuration from the AWS Command Line Interface (AWS CLI), the console, or AWS SDKs, and specifying the data encryption settings. You can provide the encryption artifacts for in-transit encryption in these two ways:

- By uploading a compressed file of certificates to Amazon S3.
- By referencing a custom Java class that provides encryption artifacts.

The security control that's included with this pattern monitors API calls and generates an Amazon CloudWatch Events event on the **RunJobFlow** action. The event calls an AWS Lambda function, which runs a Python script. The function gets the EMR cluster ID from the event JSON input, and performs the following checks to determine whether there's a security violation:

- Checks if the EMR cluster has an Amazon EMR-specific security configuration.
- If the cluster does have a security configuration, checks to see if encryption in transit is enabled.
- If the cluster doesn't have a security configuration, sends an alert to an email address that you provide, by using Amazon Simple Notification Service (Amazon SNS). The notification specifies

the EMR cluster name, violation details, AWS Region and account information, and the AWS Lambda ARN (Amazon Resource Name) that the notification is sourced from.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An S3 bucket to upload the Lambda code that's provided with this pattern.
- An email address where you would like to receive violation notifications.
- Amazon EMR logging enabled, for access to all the API logs.

Limitations

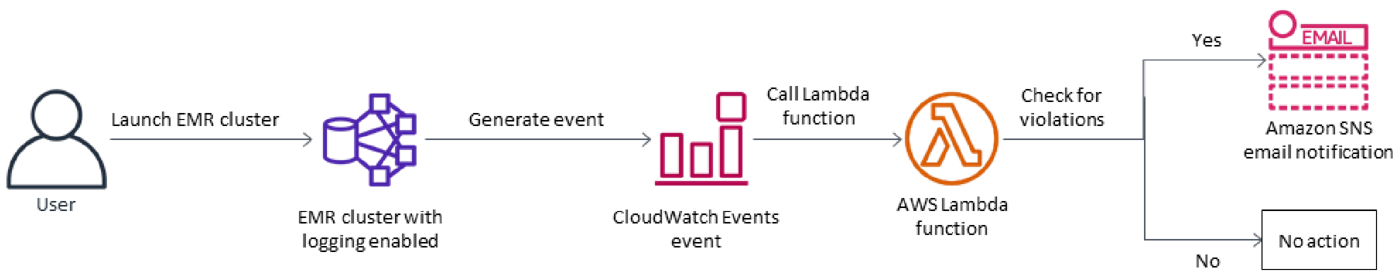
- This detective control is regional and must be deployed in each AWS Region that you want to monitor.

Product versions

- Amazon EMR release 4.8.0 or later.

Architecture

Workflow architecture



Automation and scale

- If you are using AWS Organizations, you can use AWS Cloudformation StackSets to deploy the template in multiple accounts that you want to monitor.

Tools

AWS services

- Amazon EMR – Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. By using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads. Additionally, you can use Amazon EMR to transform and move large amounts of data into and out of other AWS data stores and databases, such as Amazon S3 and Amazon DynamoDB.
- AWS Cloudformation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- AWS Cloudwatch Events – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.
- AWS SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes an attachment with two files:

- `EMRInTransitEncryption.zip` is a compressed file that includes the security control (Lambda code).
- `EMRInTransitEncryption.yml` is a CloudFormation template that deploys the security control.

See the *Epics* section for information about how to use these files.

Epics

Deploy the security control

Task	Description	Skills required
Upload the code to an S3 bucket.	Create a new S3 bucket or use an existing S3 bucket to upload the attached `EMRInTransitEncryption.zip` file (Lambda code). This bucket must be in the same AWS Region as the CloudFormation template and the resources that you want to evaluate.	Cloud architect
Deploy the CloudFormation template.	Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the `EMRInTransitEncryption.yml` file that's provided in the attachment. In the next epic, provide values for the template parameters.	Cloud architect,

Complete the parameters in the CloudFormation template

Task	Description	Skills required
Provide the S3 bucket name.	Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the resource that will be evaluated.	Cloud architect
Provide the S3 key.	Specify the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `EMRInTransitEncryption.zip` or `controls/EMRInTransitEncryption.zip`).	Cloud architect
Provide an email address.	Specify an active email address where you want to receive violation notifications.	Cloud architect
Specify a logging level.	Specify the logging level and verbosity for the Lambda logs. `Info` designates detailed informational messages on the application's progress and should be used only for debugging. `Error` designates error events that could still allow	Cloud architect

Task	Description	Skills required
	the application to continue running. `Warning` designates potentially harmful situation s.	

Confirm the subscription

Task	Description	Skills required
Confirm the email subscript ion.	When the CloudFormation template deploys successfu lly, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.	Cloud architect

Related resources

- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Encryption options (Amazon EMR documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Monitor Amazon ElastiCache clusters for at-rest encryption

Susanne Kangnoh, Amazon Web Services

Summary

Amazon ElastiCache is an Amazon Web Services (AWS) service that provides a high-performance, scalable, and cost-effective caching solution for distributing an in-memory data store or cache environment in the cloud. It retrieves data from high-throughput and low-latency, in-memory data stores. This functionality makes it a popular choice for real-time use cases such as caching, session stores, gaming, geo-spatial services, real-time analytics, and queuing. ElastiCache offers Redis and Memcached data stores, both of which provide sub-millisecond response times.

Data encryption helps prevent unauthorized users from reading sensitive data available on your Redis clusters and their associated cache storage systems. This includes data saved to persistent media, known as *data at rest*, and data that can be intercepted as it travels through the network between clients and cache servers, known as *data in transit*.

You can enable at-rest encryption for ElastiCache for Redis when you create a replication group, by setting the **AtRestEncryptionEnabled** parameter to **true**. When this parameter is enabled, it encrypts the disk during sync, backup, and swap operations, and encrypts backups stored in Amazon Simple Storage Service (Amazon S3). You cannot enable at-rest encryption on an existing replication group. When you create a replication group, you can enable encryption at rest in these two ways:

- By choosing the **Default** option, which uses service-managed encryption at rest.
- By using a customer managed key and providing the key ID or Amazon Resource Name (ARN) from AWS Key Management Service (AWS KMS).

This pattern provides a security control that monitors for API calls and generates an Amazon CloudWatch Events event on the **CreateReplicationGroup** operation. This event calls an AWS Lambda function, which runs a Python script. The function gets the replication group ID from the event JSON input, and performs the following checks to determine whether there's a security violation:

- Checks if the **AtRestEncryptionEnabled** key exists.
- If **AtRestEncryptionEnabled** exists, checks the value to see if it is **true**.

- If the **AtRestEncryptionEnabled** value is set to **false**, sets a variable that tracks violations and sends a violation message to an email address you provide, by using an Amazon Simple Notification Service (Amazon SNS) notification.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An S3 bucket to upload the provided Lambda code.
- An email address where you would like to receive violation notifications.
- ElastiCache logging enabled, for access to all the API logs.

Limitations

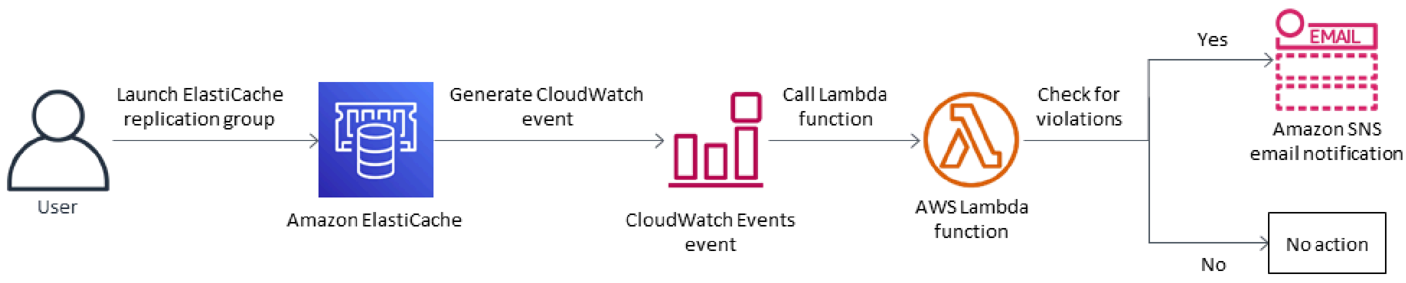
- This detective control is regional and must be deployed in each AWS Region that you want to monitor.
- The control supports replication groups that are running in a virtual private cloud (VPC).
- The control supports replication groups that are running the following node types:
  - R5, R4, R3
  - M5, M4, M3
  - T3, T2

Product versions

- ElastiCache for Redis version 3.2.6 or later

Architecture

Workflow architecture



Automation and scale

- If you are using AWS Organizations, you can use [AWS Cloudformation StackSets](#) to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- [Amazon ElastiCache](#) – Amazon ElastiCache makes it easy to set up, manage, and scale distributed in-memory cache environments in the AWS Cloud. It provides a high performance, resizable, and cost-effective in-memory cache, while removing complexity associated with deploying and managing a distributed cache environment. ElastiCache works with both the Redis and Memcached engines.
- [AWS CloudFormation](#) – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- [AWS Cloudwatch Events](#) – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources. CloudWatch Events becomes aware of operational changes as they occur and takes corrective action as necessary, by sending messages to respond to the environment, activating functions, making changes, and capturing state information.
- [AWS Lambda](#) – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.

- [Amazon SNS](#) – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes an attachment with two files:

- `ElasticCache-EncryptionAtRest.zip` is a compressed file that includes the security control (Lambda code).
- `elasticache_encryption_at_rest.yml` is a CloudFormation template that deploys the security control.

See the *Epics* section for information about how to use these files.

Epics

Deploy the security control

Task	Description	Skills required
Upload the code to an S3 bucket.	Create a new S3 bucket or use an existing S3 bucket to upload the attached `ElastiCache-EncryptionAtRest.zip` file (Lambda code). This bucket must be in the same AWS Region as the resources that you want to evaluate.	Cloud architect
Deploy the CloudFormation template.	Open the Cloudformation console in the same AWS Region as the S3 bucket, and deploy the `elasticache_encryption_at_r`	Cloud architect

Task	Description	Skills required
	`est.yml` file that's provided in the attachment. In the next epic, provide values for the template parameters.	

Complete the parameters in the CloudFormation template

Task	Description	Skills required
Provide the S3 bucket name.	Enter the name of the S3 bucket that you created or selected in the first epic. This S3 bucket contains the .zip file for the Lambda code and must be in the same AWS Region as the CloudFormation template and the resource that will be evaluated.	Cloud architect
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example, `ElasticCache-EncryptionAtRest.zip` or `controls/ElasticCache-EncryptionAtRest.zip`).	Cloud architect
Provide an email address.	Provide an active email address where you want to receive violation notifications.	Cloud architect

Task	Description	Skills required
Specify a logging level.	Specify the logging level and verbosity. `Info` designates detailed informational messages on the application's progress and should be used only for debugging. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the email subscription.	When the CloudFormation template deploys successfully, it sends a subscription email message to the email address you provided. To receive notifications, you must confirm this email subscription.	Cloud architect

Related resources

- [Creating a stack on the AWS CloudFormation console](#) (AWS CloudFormation documentation)
- [At-Rest Encryption in ElastiCache for Redis](#) (Amazon ElastiCache documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Monitor EC2 instance key pairs using AWS Config

Wassim Benhallam, Sergio Bilbao Lopez, and Vikrant Telkar, Amazon Web Services

Summary

When launching an Amazon Elastic Compute Cloud (Amazon EC2) instance on the Amazon Web Services (AWS) Cloud, a best practice is to create or use an existing key pair to connect to the instance. The key pair, which consists of a public key stored in the instance and a private key provided to the user, allows secure access through Secure Shell (SSH) to the instance and avoids the use of passwords. However, users can sometimes inadvertently launch instances without attaching a key pair. Because key pairs can be assigned only during the launch of an instance, it's important to quickly identify and flag as noncompliant any instances launched without key pairs. This is particularly useful when working in accounts or environments that mandate the use of key pairs for instance access.

This pattern describes how to create a custom rule in AWS Config to monitor EC2 instance key pairs. When instances are identified as noncompliant, an alert is sent using Amazon Simple Notification Service (Amazon SNS) notifications initiated through an Amazon EventBridge event.

Prerequisites and limitations

Prerequisites

- An active AWS account
- AWS Config enabled for the AWS Region you want to monitor and configured to record all AWS resources

Limitations

- This solution is Region-specific. All resources should be created in the same AWS Region.

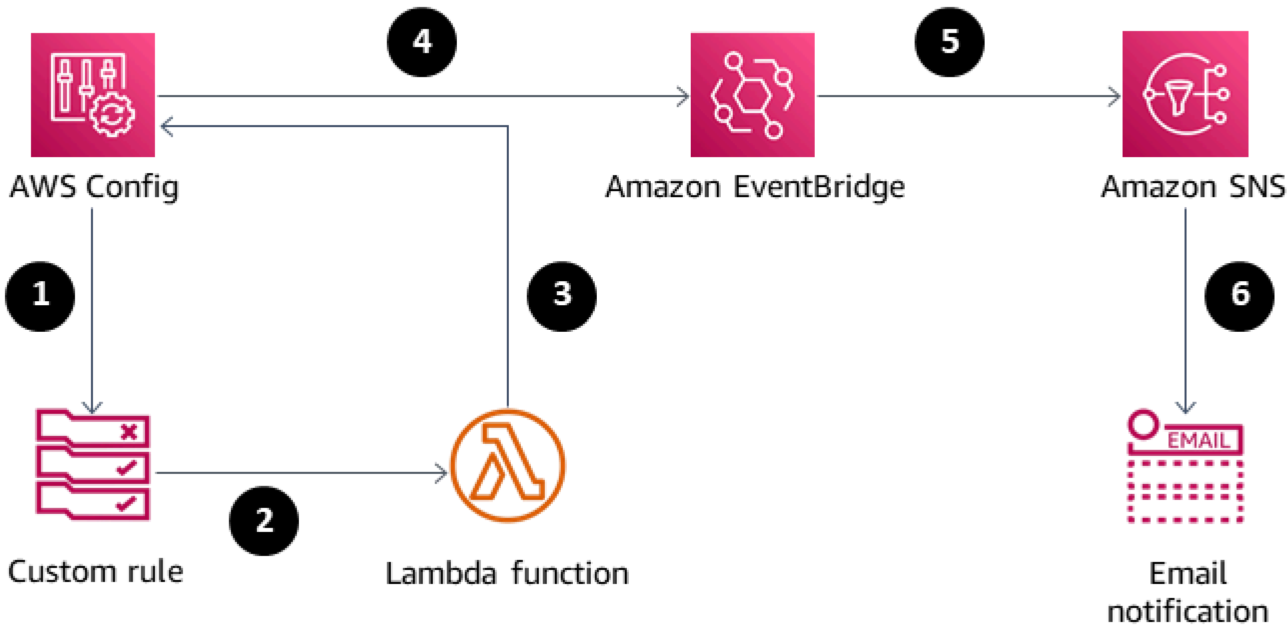
Architecture

Target technology stack

- AWS Config

- Amazon EventBridge
- AWS Lambda
- Amazon SNS

Target architecture



1. AWS Config initiates the rule.
2. The rule invokes the Lambda function to evaluate compliance of EC2 instances.
3. The Lambda function sends the updated compliance state to AWS Config.
4. AWS Config sends an event to EventBridge.
5. EventBridge publishes compliance change notifications to an SNS topic.
6. Amazon SNS sends an alert in email.

Automation and scale

The solution can monitor any number of EC2 instances within a Region.

Tools

Tools

- [AWS Config](#) – AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.
- [Amazon EventBridge](#) – Amazon EventBridge is a serverless event bus service for connecting your applications with data from a variety of sources.
- [AWS Lambda](#) – AWS Lambda is a serverless compute service that supports running code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes.
- [Amazon SNS](#) – Amazon Simple Notification Service (Amazon SNS) is a fully managed messaging service for both application-to-application (A2A) and application-to-person (A2P) communication.

Code

The code for the Lambda function is attached.

Epics

Create a Lambda function to evaluate Amazon EC2 compliance

Task	Description	Skills required
Create an AWS Identity and Access Management (IAM) role for Lambda.	On the AWS Management Console, choose **IAM**, and then create the role, using Lambda as the trusted entity and adding the `AmazonEventBridgeFullAccess` and `AWSConfigRulesExecutionRole` permissions. For more information, see the [AWS documentation](#).	DevOps
Create and deploy the Lambda function.	1. On the Lambda console, create a function, using	DevOps

Task	Description	Skills required
	Author from scratch, with **Python 3.6** as the runtime and the previously created IAM role. Note the Amazon Resource Name (ARN). 2. On the **Code** tab, choose `lambda_function.py` , and paste the code that is attached to this pattern. 3. To save your changes, choose **Deploy**.	

Create a custom AWS Config rule

Task	Description	Skills required
Add a custom AWS Config rule.	On the AWS Config console, add a custom rule, using the following settings: • ARN – The ARN of the previously created Lambda function • Trigger type – Configura tion changes • Scope of changes – Resources • Resource type – Amazon EC2 instance For more information, see the [AWS documentation]().	DevOps

Configure email notifications when a compliance change event is detected

Task	Description	Skills required
Create the SNS topic and subscription.	On the Amazon SNS console, create a topic using **Standard** as the type, and then create a subscription using **Email** as the protocol. When you receive the confirmation email message, choose the link to confirm the subscription. For more information, see the [AWS documentation].	DevOps
Create an EventBridge rule to initiate Amazon SNS notificat ions.	On the EventBridge console, create a rule, using the following settings: • Service name – AWS Config • Event type – Config Rules Compliance Change • Message type – Specific message types, Complianc eChangeNotification • Specific rule name – The name of your previously created AWS Config rule • Target – SNS topic, your previously created topic For more information, see the [AWS documentation].	DevOps

Verify the rule and notifications

Task	Description	Skills required
Create EC2 instances.	Create two EC2 instances of any type and attach a key pair, and create one EC2 instance without a key pair.	DevOps
Verify the rule.	1. On the AWS Config console, on the **Rules** page, select your rule. 2. To see compliant and noncompliant EC2 instances, change **Resources in scope** to **All**. Verify that two instances are listed as compliant and that one instance is listed as noncompliant. 3. Wait to receive Amazon SNS email notification regarding the compliance state of the EC2 instances.	DevOps

Related resources

- Creating a role to delegate permissions to an AWS service
- Creating a custom rule in AWS Config
- Creating an Amazon SNS topic
- Subscribing to an Amazon SNS topic
- Create a rule in Amazon EventBridge

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Monitor IAM root user activity

Mostefa Brougui, Amazon Web Services

Summary

Every Amazon Web Services (AWS) account has a root user. As a security best practice for AWS Identity and Access Management (IAM), we recommend that you use the root user to complete the tasks that only the root user can perform. For the complete list, see Tasks that require root user credentials in the *AWS Account Management Reference Guide*. Because the root user has full access to all of your AWS resources and billing information, we recommend that you don't use this account and monitor it for any activity, which might indicate that the root user credentials have been compromised.

Using this pattern, you set up an event-driven architecture that monitors the IAM root user. This pattern sets up a hub-and-spoke solution that monitors multiple AWS accounts, the *spoke* accounts, and centralizes management and reporting in a single account, the *hub* account.

When the IAM root user credentials are used, Amazon CloudWatch and AWS CloudTrail record the activity in the log and trail, respectively. In the spoke account, an Amazon EventBridge rule sends the event to the central event bus in the hub account. In the hub account, an EventBridge rule sends the event to an AWS Lambda function. The function uses an Amazon Simple Notification Service (Amazon SNS) topic that notifies you of the root user activity.

In this pattern, you use an AWS CloudFormation template to deploy the monitoring and event-handling services in the spoke accounts. You use a HashiCorp Terraform template to deploy the event-management and notification services in the hub account.

Prerequisites and limitations

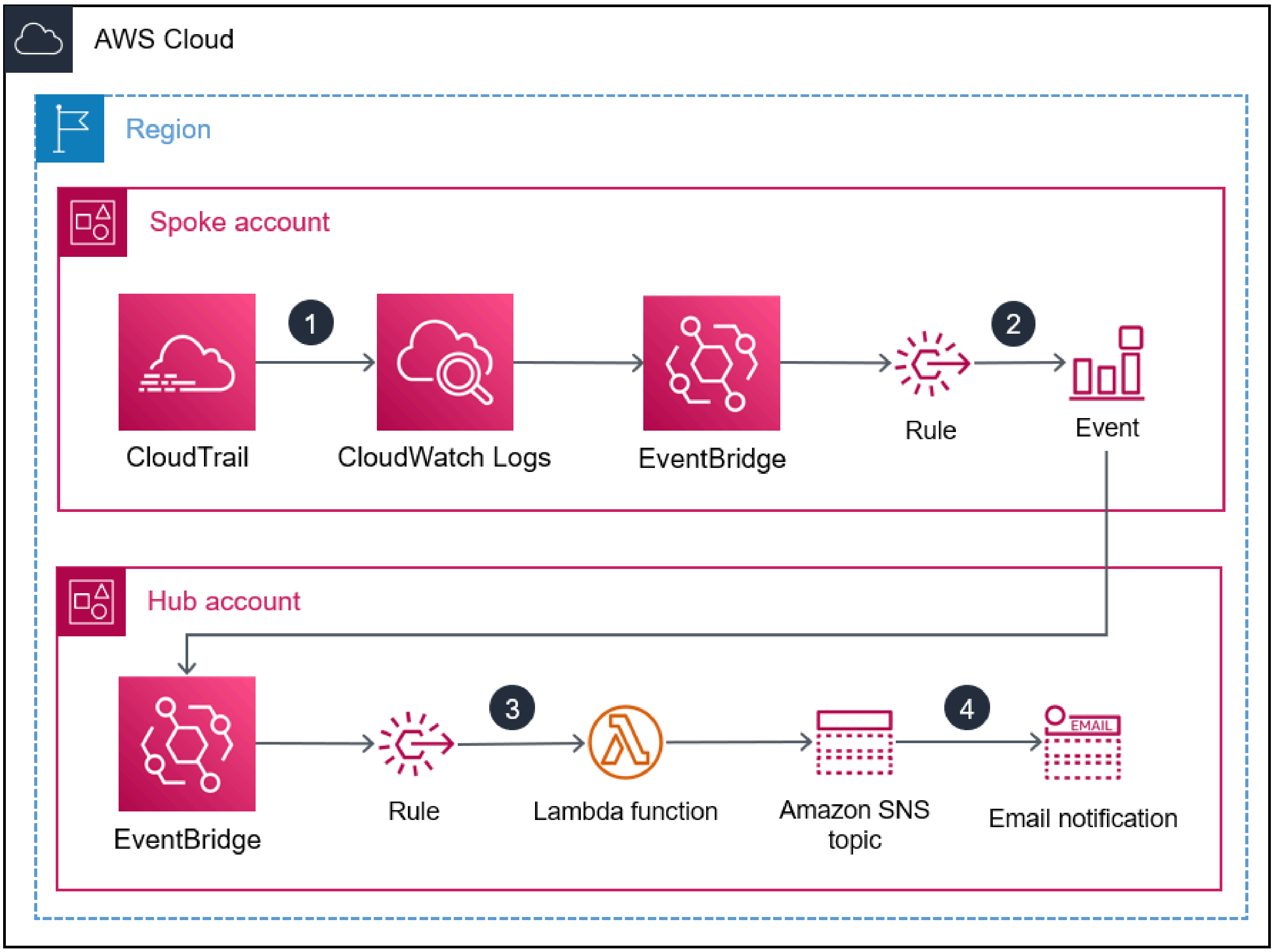
Prerequisites

1. Permissions to deploy AWS resources in your AWS environment.
2. Permissions to deploy CloudFormation stack sets. For more information, see Prerequisites for stack set operations (CloudFormation documentation).
3. Terraform installed and ready to use. For more information, see Get Started – AWS (Terraform documentation).
4. An existing trail in each spoke account. For more information, see Getting started with AWS CloudTrail (CloudTrail documentation).

5. The trail is configured to send events to CloudWatch Logs. For more information, see [Sending events to CloudWatch Logs](#) (CloudTrail documentation).
6. Your hub and spoke accounts must be managed by AWS Organizations.

Architecture

The following diagram illustrates the building blocks of the implementation.



1. When the IAM root user credentials are used, CloudWatch and CloudTrail record the activity in the log and trail, respectively.
2. In the spoke account, an EventBridge rule sends the event to the central [event bus](#) in the hub account.
3. In the hub account, an EventBridge rule sends the event to a Lambda function.

4. The Lambda function uses an Amazon SNS topic that notifies you of the root user activity.

Tools

AWS services

- AWS CloudFormation helps you set up AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle across AWS accounts and Regions.
- AWS CloudTrail helps you audit the governance, compliance, and operational risk of your AWS account.
- Amazon CloudWatch Logs helps you centralize the logs from all your systems, applications, and AWS services so you can monitor them and archive them securely.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.

Other tools and services

- Terraform is a CLI application for provisioning and managing cloud infrastructure and resources by using code, in the form of configuration files.

Code repository

The source code and templates for this pattern are available in a GitHub repository. This pattern provides two templates:

- A Terraform template containing the resources you deploy in the hub account
- A CloudFormation template you deploy as a stack set instance in the spoke accounts

The repository has the following overall structure.

```
.
 |__README.md
 |__spoke-stackset.yaml
 |__hub.tf
 |__root-activity-monitor-module
     |__main.tf  # contains Terraform code to deploy resources in the Hub account
     |__iam      # contains IAM policies JSON files
         |__ lambda-assume-policy.json          # contains trust policy of the IAM role
 used by the Lambda function
         |__ lambda-policy.json                 # contains the IAM policy attached to
 the IAM role used by the Lambda function
     |__outputs  # contains Lambda function zip code
```

The *Epics* section provides step-by-step instructions for deploying the templates.

Epics

Deploy resources to the hub account

Task	Description	Skills required
Clone the sample code repository.	1. Open the [AWS IAM Root User Activity Monitor](#) repository. 2. On the Code tab, above the file list, choose **Code**, and then copy the HTTPS URL. 3. In a command-line interface, change your working directory to the location where you want to store the sample files. 4. Enter the following command: `git clone <repoURL>`	General AWS

Task	Description	Skills required
Update the Terraform template.	1. Retrieve your organization ID. For instructions, see [Viewing the details of an organization from the management account](#) (AWS Organizations documentation). 2. In the cloned repository, open `hub.tf`. 3. Update the following with the appropriate values for your environment: • `OrganizationId` – Add your organization ID. • `SNSTopicName` – Add a name for the Amazon SNS topic. • `SNSSubscriptions` – Add the email to which Amazon SNS notifications should be sent. • `Region` – Add the AWS Region code where you are deploying the resources. For example, `eu-west-1`. • `Tags` – Add your tags. For more information, see [Tagging AWS resources](#) (AWS General Reference).	General AWS

Task	Description	Skills required
	4. Save and close the `hub.tf` file.	

Deploy the resources to the AWS hub account.	1. In the Terraform command-line interface, navigate to the root folder of the cloned repositor y, and then enter the following command.
	`terraform init && terraform plan`
	2. Review the output and confirm you want to create the resources described.
	3. Enter the following command.
	`terraform apply`
	4. When prompted, confirm the deployment by entering `yes`.
		General AWS

Deploy resources to your spoke accounts

Task	Description	Skills required
Deploy the CloudFormation template.	1. Sign in to the AWS Management Console, and open the [CloudFormation console](https://console.aws.amazon.com/cloudformation/). 2. From the navigation pane, choose **StackSets**.	General AWS

Task	Description	Skills required
	3. At the top of the **StackSets** page, choose **Create StackSet**. 4. Under **Permissions**, choose **Service-managed permissions**. CloudFormation automatically configures the permissions required to deploy to the target accounts managed by AWS Organizations. 5. Under **Prerequisite - Prepare template**, choose **Template is ready**. 6. Under **Specify Template**, choose **Upload a template file**. 7. Choose **Choose file**, and then in the cloned repository, select `spoke-stackset.yaml` . 8. Choose **Next**. 9. On the **Specify StackSet details** page, enter a name for the stack set. 10. Under **Parameters**, enter the account ID of the hub account, and then choose **Next.** 11. On the **Configure StackSet options** page, under **Tags**, add your tags.	

Task	Description	Skills required
	12. Under **Execution configuration**, choose **Inactive**, and then choose **Next.** 13. On the **Set deployment options** page, specify the organizational units and Regions where you want to deploy the stack set, then choose **Next**. 14. On the Review page, select **I acknowledge that AWS CloudFormation might create IAM resources**, and then choose **Submit**. CloudFormation starts deploying your stack set. For more information and instructions, see Create a stack set (CloudFormation documentation).	

(Optional) Test the notifications

Task	Description	Skills required
Use the root user credentials.	1. Sign into a spoke account or the hub account by using the root user credentials. 2. Confirm that the email account you specified	General AWS

Task	Description	Skills required
	receives the Amazon SNS notification.	

Related resources

- Security best practices (IAM documentation)
- Working with StackSets (CloudFormation documentation)
- Get Started (Terraform documentation)

Additional information

Amazon GuardDuty is a continuous security monitoring service that analyzes and processes logs to identify unexpected and potentially unauthorized activity in your AWS environment. As an alternative to this solution, if you have enabled GuardDuty, it can alert you when the root user credentials have been used. The GuardDuty finding is `Policy:IAMUser/RootCredentialUsage`, and the default severity is **Low**. For more information, see Managing Amazon GuardDuty findings.

Send a notification when an IAM user is created

Mansi Suratwala and Sergiy Shevchenko, Amazon Web Services

Summary

On Amazon Web Services (AWS), you can use this pattern to deploy an AWS CloudFormation template to receive notifications automatically when AWS Identity and Access Management (IAM) users are created.

Using IAM, you can manage access to AWS services and resources securely. You can create and manage AWS users and groups, and use permissions to allow and deny those users and groups access to AWS resources.

The CloudFormation template creates an Amazon CloudWatch Events event and an AWS Lambda function. The event uses AWS CloudTrail to monitor for any IAM user being created in the AWS account. If a user is created, the CloudWatch Events event initiates a Lambda function, which sends you an Amazon Simple Notification Service (Amazon SNS) notification informing you of the new user creation event.

Prerequisites and limitations

Prerequisites

- An active AWS account
- An AWS CloudTrail trail created and deployed

Limitations

- The AWS CloudFormation template must be deployed for `CreateUser` only.

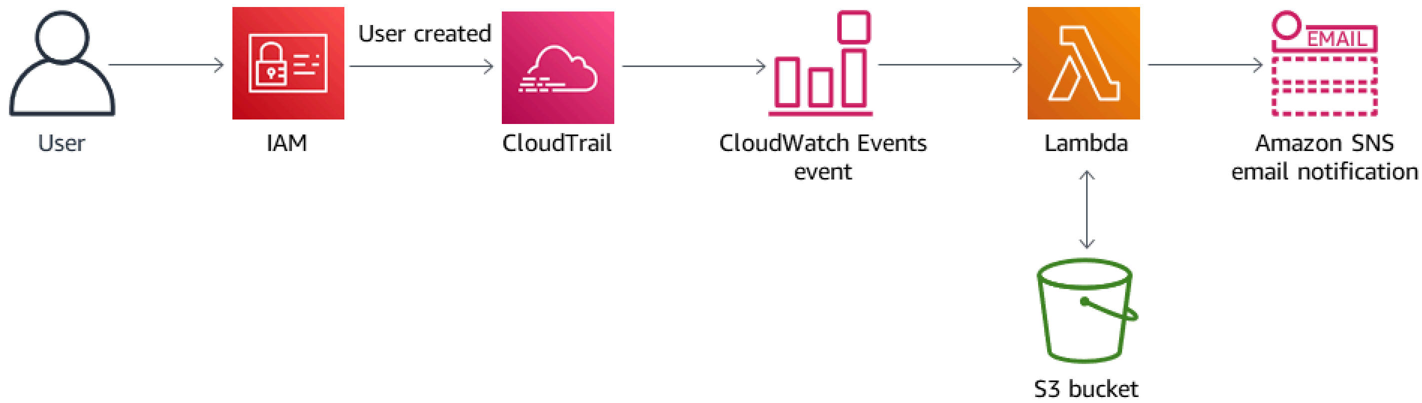
Architecture

Target technology stack

- IAM
- AWS CloudTrail
- Amazon CloudWatch Events

- AWS Lambda
- Amazon Simple Storage Service (Amazon S3)
- Amazon SNS

Target architecture



Automation and scale

You can use the AWS CloudFormation template multiple times for different AWS Regions and accounts. You need to run it only once in each Region or account. To automate deployment to multiple accounts, use AWS CloudFormation StackSets. The CloudFormation template will be able to deploy all the required resources in each account.

Tools

Tools

- IAM – AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- AWS CloudFormation – AWS CloudFormation helps you model and set up your Amazon Web Services resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want, and CloudFormation takes care of provisioning and configuring those resources for you.
- AWS CloudTrail – AWS CloudTrail helps you manage governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, a role, or an AWS service are

recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.

- Amazon CloudWatch Events – Amazon CloudWatch Events delivers a near-real-time stream of system events that describe changes in AWS resources.
- AWS Lambda – AWS Lambda is a compute service that supports running code without provisioning or managing servers. Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is storage for the internet. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery using Lambda, HTTP, email, mobile push notifications, and mobile text messages (SMS).

Code

A .zip file of the project is available as an attachment.

Epics

Create the S3 bucket for the Lambda script

Task	Description	Skills required
Define the S3 bucket.	Open the Amazon S3 console, and choose or create an S3 bucket. This S3 bucket will host the Lambda code .zip file. The S3 bucket name cannot contain leading slashes.	Cloud architect

Upload the Lambda code to the S3 bucket

Task	Description	Skills required
Upload the Lambda code.	Upload the Lambda code .zip file provided in the *Attachments* section to the S3 bucket that you defined.	Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Deploy the CloudFormation template.	On the CloudFormation console, deploy the CloudFormation `createIAM user.yaml` template that's provided as an attachment to this pattern. In the next epic, provide values for the template parameters.	Cloud architect

Complete the parameters in the CloudFormation template

Task	Description	Skills required
Provide the S3 bucket name.	Enter the name of the S3 bucket that you created or chose in the first epic.	Cloud architect
Provide the S3 key.	Provide the location of the Lambda code .zip file in your S3 bucket, without leading slashes (for example,	Cloud architect

Task	Description	Skills required
	`<directory>/<file-name>.zip`).	
Provide an email address.	Provide an active email address to receive Amazon SNS notifications.	Cloud architect
Define the logging level.	Define the logging level and frequency for your Lambda function. `Info` designates detailed informational messages on the application's progress. `Error` designates error events that could still allow the application to continue running. `Warning` designates potentially harmful situations.	Cloud architect

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the template successfully deploys, it sends a subscription email message to the email address provided. To receive notifications, you must confirm this email subscription.	Cloud architect

Related resources

- Creating a trail

- [Creating an S3 bucket]
- [Uploading files to an S3 bucket]
- [Deploying a CloudFormation template]
- [Creating an IAM user]
- [Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail]

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Prevent internet access at the account level by using a service control policy

Sergiy Shevchenko, Sean O'Sullivan, and Victor Mazeo Whitaker, Amazon Web Services

Summary

Organizations frequently want to limit internet access for account resources that should remain private. In these accounts, the resources in virtual private clouds (VPCs) should not access the internet by any means. Many organizations choose a centralized inspection architecture. For the east-west (VPC-to-VPC) traffic in a centralized inspection architecture, you need to make sure that the spoke accounts and their resources do not have access to the internet. For north-south (internet egress and on-premises) traffic, you want to allow internet access only through the inspection VPC.

This pattern uses a service control policy (SCP) to help prevent internet access. You can apply this SCP at the account or organizational unit (OU) level. The SCP limits internet connectivity by preventing the following:

- Creating or attaching an IPv4 or IPv6 internet gateway that allows direct internet access to the VPC
- Creating or accepting a VPC peering connection that might allow indirect internet access through another VPC
- Creating or updating an AWS Global Accelerator configuration that might allow direct internet access to VPC resources

Prerequisites and limitations

Prerequisites

- One or multiple AWS accounts managed as an organization in AWS Organizations.
- All features are enabled in AWS Organizations.
- SCPs are enabled in the organization.
- Permissions to:
  - Access the organization's management account.
  - Create SCPs. For more information about the minimum permissions, see Creating an SCP.

- Attach the SCP to the target accounts or organizational units (OUs). For more information about the minimum permissions, see Attaching and detaching service control policies.

Limitations

- SCPs don't affect users or roles in the management account. They affect only the member accounts in your organization.
- SCPs affect only AWS Identity and Access Management (IAM) users and roles that are managed by accounts that are part of the organization. For more information, see SCP effects on permissions.

Tools

AWS services

- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage. In this pattern, you use service control policies (SCPs) in AWS Organizations.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Best practices

After establishing this SCP in your organization, make sure to update it frequently to address any new AWS services or features that might affect internet access.

Epics

Create and attach the SCP

Task	Description	Skills required
Create the SCP.	1. Sign in to the AWS Organizations console. You must sign in to the	AWS administrator

Task	Description	Skills required

	organization's managemen t account.
	2. In the left pane, choose **Policies**.
	3. On the policies page, choose **Service control policies**.
	4. On the **Service control policies** page, choose **Create policy**.
	5. On the **Create new service control policy** page, enter a **Policy name** and an optional **Policy descripti on**.
	6. (Optional) Add [AWS tags](#) to your policy.
	7. In the JSON editor, delete the placeholder policy.
	8. Paste the following policy into the JSON editor.
	<pre>{
	  "Version":
	 "2012-10-17",
	  "Statement": [
	    {
	      "Action": [
	        "ec2:Atta
	chInternetGateway",
	        "ec2:Crea
	teInternetGateway",
	
	        "ec2:Crea
	teVpcPeeringConnec
	tion",</pre>	

Task	Description	Skills required

	<pre> "ec2:Acce
	ptVpcPeeringConnec
	tion",
	        "ec2:Crea
	teEgressOnlyIntern
	etGateway"
	      ],
	      "Resource":
	 "*",
	      "Effect": "Deny"
	    },
	    {
	      "Action": [
	        "globalac
	celerator:Create*",
	        "globalac
	celerator:Update*"
	      ],
	      "Resource":
	 "*",
	      "Effect": "Deny"
	    }
	  ]
	}</pre>9. Choose **Create policy**.	

Task	Description	Skills required
Attach the SCP.	1. On the **Service control policies** page, choose the policy you created. 2. On the **Targets** tab, choose **Attach**. 3. Select the OU or account that you want to attach the policy to. You might have to expand the OUs to find the OU or account that you want. 4. Choose **Attach policy**.	AWS administrator

Related resources

- AWS Organizations documentation
- Service control policies (SCPs)
- Centralized inspection architecture with AWS Gateway Load Balancer and AWS Transit Gateway (AWS blog post)

Export a report of AWS IAM Identity Center identities and their assignments by using PowerShell

Jorge Pava, Frank Allotta, Manideep Reddy Gillela, and Chad Miles, Amazon Web Services

Summary

When you use AWS IAM Identity Center (successor to AWS Single Sign-On) to centrally manage single sign-on (SSO) access to all of your Amazon Web Services (AWS) accounts and cloud applications, reporting and auditing those assignments through the AWS Management Console can be tedious and time consuming. This is especially true if you're reporting on permissions for a user or group across dozens or hundreds of AWS accounts.

For many, the ideal tool to view this information would be in a spreadsheet application, such as Microsoft Excel. This can help you filter, search, and visualize the data for your entire organization, managed by AWS Organizations.

This pattern describes how to use AWS Tools for PowerShell to generate a report of SSO identity configurations in IAM Identity Center. The report is formatted as a CSV file, and it includes the identity name (principal), identity type (user or group), accounts the identity can access, and permission sets. After generating this report, you can open it in your preferred application to search, filter, and audit the data as needed. The following image shows sample data in a spreadsheet application.

AccountName	PermissionSet	Principal	Type
Audit	SecurityAudit	AWSAuditAccountAdmins	GROUP
Log Archive	AWSPowerUserAccess	AWSSecurityAuditPowerUsers	GROUP
Dev-Test	AWSAdministratorAccess	dev-Users	GROUP
Prod	AWSReadOnlyAccess	Prod@example.com	USER

Important: Because this report contains sensitive information, we highly recommend you store it securely and share it only on a need-to-know basis.

Prerequisites and limitations

Prerequisites

- IAM Identity Center and AWS Organizations, configured and enabled.

- PowerShell, installed and configured. For more information, see Installing PowerShell (Microsoft documentation).
- AWS Tools for PowerShell, installed and configured. For performance reasons, we highly recommend that you install the modularized version of AWS Tools for PowerShell, called `AWS.Tools`. Each AWS service is supported by its own individual, small module. In the PowerShell shell, enter the following commands to install the modules needed for this pattern: `AWS.Tools.Installer`, `Organizations`, `SSOAdmin`, and `IdentityStore`.

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule -Name Organizations, SSOAdmin, IdentityStore
```

  For more information, see Install AWS.Tools on Windows or Install AWS.Tools on Linux or macOS (AWS Tools for PowerShell documentation). If you receive an error when installing the modules, see the Troubleshooting section of this pattern.
- AWS Command Line Interface (AWS CLI) or the AWS SDK must be previously configured with working credentials by doing one of the following:
  - Use the AWS CLI `aws configure` For more information, see Quick configuration (AWS CLI documentation).
  - Configure AWS CLI or AWS Cloud Development Kit (AWS CDK) to get temporary access through an AWS Identity and Access Management (IAM) role. For more information, see Getting IAM role credentials for CLI access (IAM Identity Center documentation).
- A named profile for the AWS CLI that has saved credentials for an IAM principal that:
  - Has access to the AWS Organizations management account or the delegated administrator account for IAM Identity Center
  - Has the `AWSSSOReadOnly` and `AWSSSODirectoryReadOnly` AWS managed policies applied to it

  For more information, see Using named profiles (AWS CLI documentation) and AWS managed policies (IAM documentation).

Limitations

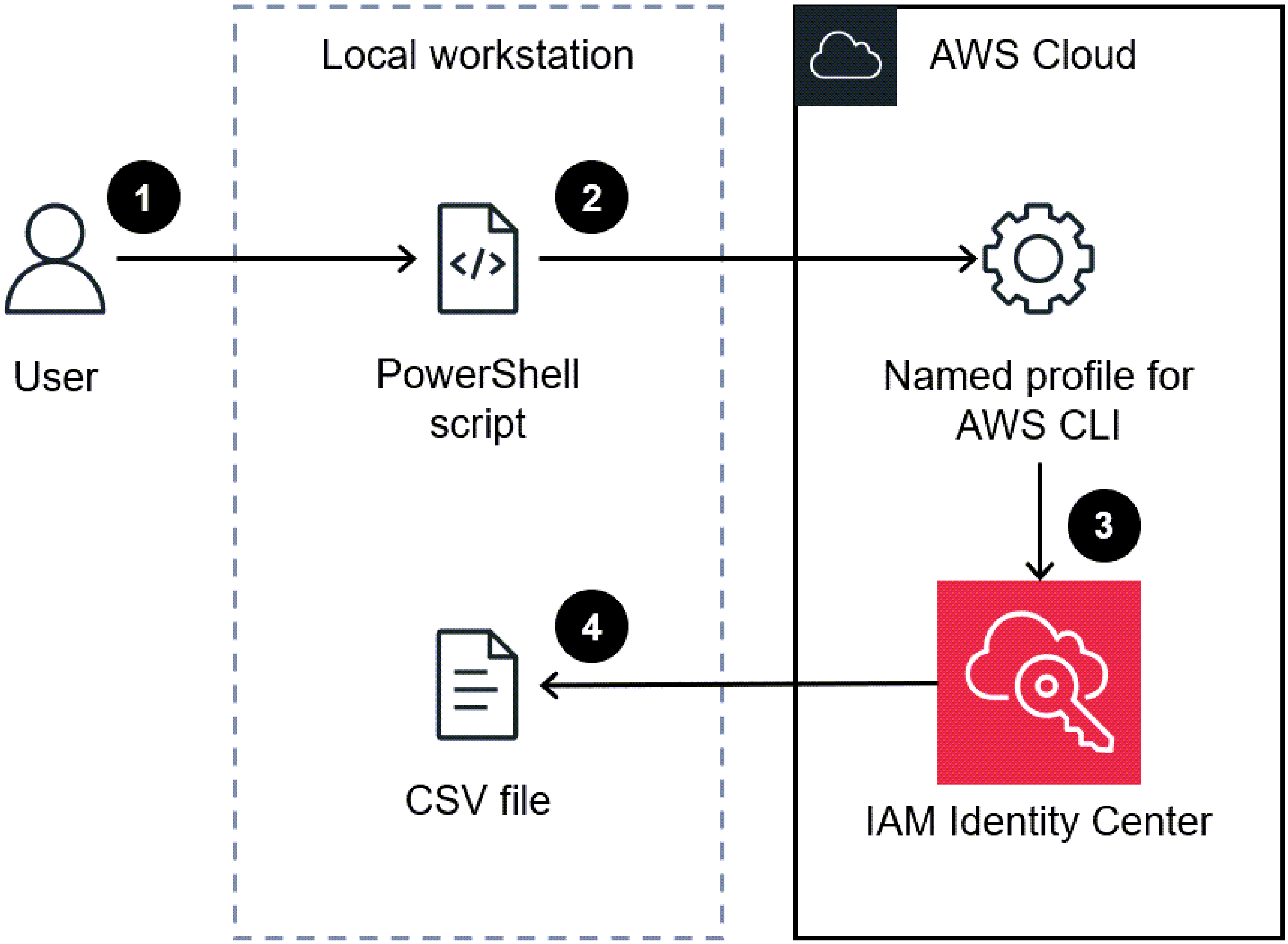
- The target AWS accounts must be managed as an organization in AWS Organizations.

Product versions

- For all operating systems, it is recommended that you use [PowerShell version 7.0]() or later.

Architecture

Target architecture



1. The user runs the script in a PowerShell command line.
2. The script assumes the named profile for AWS CLI. This grants access to IAM Identity Center.
3. The script retrieves the SSO identity configurations from IAM Identity Center.
4. The script generates a CSV file in the same directory on the local workstation where the script is saved.

Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to all of your AWS accounts and cloud applications.
- AWS Tools for PowerShell are a set of PowerShell modules that help you script operations on your AWS resources from the PowerShell command line.

Other tools

- PowerShell is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS.

Epics

Generate the report

Task	Description	Skills required
Prepare the script.	1. Copy the PowerShell script in the Additional information section of this pattern. 2. In the `Param` section, for your AWS environment, define the values for the following variables: • `OutputFile` – The file name of the report. • `ProfileName` – The AWS CLI named profile that you want to use to generate the report.	Cloud administrator

Task	Description	Skills required
	• `Region` – The AWS Region in which IAM Identity Center is deployed. For a complete list of Regions and their codes, see [Regional endpoints](). 3. Save the script with the file name `SSO-Report.ps1` .	

Run the script.	It is recommended that you run your custom script in the PowerShell shell with the following command.
	`.\SSO-Report.ps1`
	Alternatively, you can run the script from another shell by entering the following command.
	`pwsh .\SSO-Report.ps1`
	The script generates a CSV file in the same directory as the script file.
		Cloud administrator

Task	Description	Skills required
Analyze report data.	The output CSV file has the headers **AccountName**, **PermissionSet**, **Principal**, and **Type**. Open this file in your preferred spreadsheet application. You can create a data table to filter and sort the output.	Cloud administrator

Troubleshooting

Issue	Solution

`The term ‘Get-<parameter>’ is not recognized as the name of a cmdlet, function, script file, or operable program.` error	AWS Tools for PowerShell or its modules are not installed. In the PowerShell shell, enter the following commands to install AWS Tools for PowerShell and the modules needed for this pattern: `AWS.Tools.Installer`, `Organizations`, `SSOAdmin`, and `IdentityStore`.
	`Install-Module AWS.Tools.Installer`
	`Install-AWSToolsModule -Name Organizations, SSOAdmin, IdentityStore`

`No credentials specified or obtained from persisted/shell defaults` error	In *Prepare the script* in the [Epics](#) section, confirm that you have correctly entered the `ProfileName` and `Region` variables. Make sure that the settings and credentials in the named profile have sufficient permissions to administer IAM Identity Center.

<table>
<tr><th>Issue</th><th>Solution</th></tr>
<tr><td><code>Authenticode Issuer …</code> error when installing the AWS.Tools modules</td><td>Add the <code>-SkipPublisherCheck</code> parameter to the end of the <code>Install-AWSToolsModule</code> command.</td></tr>
<tr><td><code>Get-ORGAccountList : Assembly AWSSDK.SSO could not be found or loaded.</code> error</td><td>This error can occur when named AWS CLI profiles are specified, AWS CLI is configured to authenticate users with IAM Identity Center, and AWS CLI is configured to automatically retrieve refreshed authentication tokens. To resolve this error, do the following:

1. Enter the following command to confirm that the SSO and SSOOIDC modules are installed.
<pre><code>Install-AWSToolsModule SSO, SSOOIDC</code></pre>2. Insert the following lines into the script below the <code>param()</code> block.
<pre><code>Import-Module AWS.Tools.SSO</code></pre><pre><code>Import-Module AWS.Tools.SSOOIDC</code></pre></td></tr>
</table>

Related resources

- Where are configuration settings stored? (AWS CLI documentation)
- Configuring the AWS CLI to use AWS IAM Identity Center (AWS CLI documentation)
- Using named profiles (AWS CLI documentation)

Additional information

In the following script, determine whether you need to update the values for the following parameters:

- If you're using a named profile in AWS CLI to access the account in which IAM Identity Center is configured, update the `$ProfileName` value.
- If IAM Identity Center is deployed in a different AWS Region than the default Region for your AWS CLI or AWS SDK configuration, update the `$Region` value to use the Region where IAM Identity Center is deployed.
- If neither of these situations apply, then no script update is required.

```
param (
    # The name of the output CSV file
    [String] $OutputFile  = "SSO-Assignments.csv",
    # The AWS CLI named profile
    [String] $ProfileName = "",
    # The AWS Region in which IAM Identity Center is configured
    [String] $Region      = ""
)
$Start = Get-Date; $OrgParams = @{}
If ($Region){ $OrgParams.Region = $Region}
if ($ProfileName){$OrgParams.ProfileName = $ProfileName}
$SSOParams   = $OrgParams.Clone(); $IdsParams = $OrgParams.Clone()
$AccountList = Get-ORGAccountList @OrgParams | Select-Object Id, Name
$SSOinstance = Get-SSOADMNInstanceList @OrgParams
$SSOParams['InstanceArn']       = $SSOinstance.InstanceArn
$IdsParams['IdentityStoreId']   = $SSOinstance.IdentityStoreId
$PSsets       = @{}; $Principals   = @{}
$Assignments  = @(); $AccountCount = 1; Write-Host ""
foreach ($Account in $AccountList) {
    $Duration = New-Timespan -Start $Start -End (Get-Date) | ForEach-Object
 {[Timespan]::New($_.Days, $_.Hours, $_.Minutes, $_.Seconds)}
    Write-Host "`r$Duration - Account $AccountCount of $($AccountList.Count)
 (Assignments:$($Assignments.Count))        " -NoNewline
    $AccountCount++
    foreach ($PS in Get-SSOADMNPermissionSetsProvisionedToAccountList -AccountId
 $Account.Id @SSOParams) {
        if (-not $PSsets[$PS]) {$PSsets[$PS] = (Get-SSOADMNPermissionSet @SSOParams -
PermissionSetArn $PS).Name;$APICalls++}
        $AssignmentsResponse = Get-SSOADMNAccountAssignmentList @SSOParams -
PermissionSetArn $PS -AccountId $Account.Id
        if ($AssignmentsResponse.NextToken) {$AccountAssignments =
 $AssignmentsResponse.AccountAssignments}
        else {$AccountAssignments = $AssignmentsResponse}
        While ($AssignmentsResponse.NextToken) {
```

```
            $AssignmentsResponse = Get-SSOADMNAccountAssignmentList @SSOParams -
PermissionSetArn $PS -AccountId $Account.Id -NextToken $AssignmentsResponse.NextToken
            $AccountAssignments += $AssignmentsResponse.AccountAssignments}
        foreach ($Assignment in $AccountAssignments) {
            if (-not $Principals[$Assignment.PrincipalId]) {
                $AssignmentType = $Assignment.PrincipalType.Value
                $Expression     = "Get-IDS"+$AssignmentType+" @IdsParams -"+
$AssignmentType+"Id "+$Assignment.PrincipalId
                $Principal      = Invoke-Expression $Expression
                if ($Assignment.PrincipalType.Value -eq "GROUP")
 { $Principals[$Assignment.PrincipalId] = $Principal.DisplayName }
                else { $Principals[$Assignment.PrincipalId] = $Principal.UserName }
            }
            $Assignments += [PSCustomObject]@{
                AccountName     = $Account.Name
                PermissionSet   = $PSsets[$PS]
                Principal       = $Principals[$Assignment.PrincipalId]
                Type            = $Assignment.PrincipalType.Value}
        }
    }
}
$Duration = New-Timespan -Start $Start -End (Get-Date) | ForEach-Object
 {[Timespan]::New($_.Days, $_.Hours, $_.Minutes, $_.Seconds)}
Write-Host "`r$($AccountList.Count) accounts done in $Duration. Outputting result to
 $OutputFile"
$Assignments | Sort-Object Account | Export-CSV -Path $OutputFile -Force
```

Restrict access based on IP address or geolocation by using AWS WAF

Louis Hourcade, Amazon Web Services

Summary

AWS WAF is a web application firewall that helps protect web applications and APIs against common web exploits and bots that can affect availability, compromise security, or consume excessive resources. Web access control lists (web ACLs) in AWS WAF give you control over how traffic reaches your applications. In a web ACL, you add rules or rule groups that are designed to permit legitimate traffic, control bot traffic, and block common attack patterns. For more information, see How AWS WAF works.

You can associate the following types of rules to your AWS WAF web ACLs:

- Managed rule groups – AWS Managed Rules teams and AWS Marketplace sellers offer preconfigured sets of rules. Some managed rule groups are designed to help protect specific types of web applications. Others offer broad protection against known threats or common vulnerabilities.
- Custom rules and custom rule groups – You can also create rules and rule groups that customize access to your web applications and APIs. For example, you can restrict traffic based on a specific list of IP addresses or on a list of countries.

By using this pattern and the associated code repository, you can use the AWS Cloud Development Kit (AWS CDK) to deploy AWS WAF web ACLs with custom rules. These rules restrict access to web application resources based on the end user's IP address or geolocation. You can also optionally attach several managed rule groups.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Permissions to deploy AWS WAF resources
- AWS CDK, installed and configured in your account
- Git, installed

Limitations

- You can use this pattern only in AWS Regions where AWS WAF is available. For Region availability, see AWS services by Region.

Tools

AWS services

- AWS Cloud Development Kit (AWS CDK) is a software development framework that helps you define and provision AWS Cloud infrastructure in code.
- AWS WAF is a web application firewall that helps you monitor HTTP and HTTPS requests that are forwarded to your protected web application resources.

Code repository

The code for this pattern is available in the GitHub IP and geolocation restriction with AWS WAF repository. The code deploys two AWS WAF web ACLs. The first is a regional web ACL that is intended for Amazon API Gateway resources. The second is global web ACL for Amazon CloudFront resources. Both web ACLs contain the following custom rules:

- `IPMatch` blocks requests from non-allowed IP addresses.
- `GeoMatch` blocks requests from non-allowed countries.

During deployment, you can optionally attach all of the following managed rule groups to your web ACLs:

- Core rule set (CRS) – This rule group contains rules that are generally applicable to web applications. It helps protect against exploitation of a wide range of vulnerabilities, including some of the high risk and commonly occurring vulnerabilities described in OWASP publications, such as OWASP Top 10.
- Admin protection – This rule group contains rules that help you block external access to exposed administrative pages.
- Known bad inputs – This rule group helps block request patterns that are known to be invalid and are associated with the exploitation or discovery of vulnerabilities.

- Amazon IP reputation list – This rule group contains rules that are based on Amazon internal threat intelligence. It helps you block IP addresses that are typically associated with bots or other threats.
- Linux operating system managed rule group – This rule group helps block request patterns that are associated with the exploitation of Linux vulnerabilities, including Linux-specific Local File Inclusion (LFI) attacks.
- SQL database managed rule group – This rule group helps block request patterns that are associated with the exploitation of SQL databases, such as SQL injection attacks.

Epics

Configure the AWS WAF web ACLs

Task	Description	Skills required
Clone the repository.	Enter the following command to clone the IP and geolocation restriction with AWS WAF repository to your local workstation: `git clone https://github.com/aws-samples/ip-and-geolocation-restriction-with-waf-cdk.git`	Git

Configure the rules.	1. In the cloned repository, open the **app.py** file.
	2. Modify the values of the following variables to customize the rules:
	`aws_acccount = "AWS_ACCOUNT"`
	`region = "AWS_REGION"`
		General AWS, Python

Task	Description	Skills required

	<pre>ip_list = ["CIDR_RANGE_1", "CIDR_RANGE_2"]
	geo_list = ["COUNTRY_CODE_1", "COUNTRY_CODE_2"]
	aws_managed_rules = True</pre> Where: `aws_account` is the ID of the target AWS account. `region` is the target AWS Region for the web ACL for API Gateway resources. **Note** The web ACL for CloudFront resources is global and will be deployed in the `us-east-1` Region. `ip_list` is the list of CIDR ranges that are allowed access. `geo_list` is the list of countries that are allowed access. For valid values, see the AWS WAF documentation.	

Task	Description	Skills required
	• `aws_managed_rules` controls whether the managed rule groups are added to the web ACL. If this value is `True`, they are added. If this value is `False`, they are excluded. 3. Save and close the **app.py** file.	

Bootstrap and deploy the code

Task	Description	Skills required
Bootstrap your AWS environment.	If not already done, you need to [bootstrap](#) your AWS environment before you can deploy the AWS CDK applicati on. 1. In the AWS CDK CLI, enter the following command to bootstrap the `us-east-1` Region: `cdk bootstrap aws://<account-id>/us-east-1` 2. If you're deploying the web ACL for API Gateway in a Region other than `us-east-1` , enter the following command to	General AWS

Task	Description	Skills required
	bootstrap the target Region: ```cdk bootstrap aws://<account-id>/<region>```	
Deploy the AWS CDK application.	1. Enter the following command to deploy the AWS CDK application: ```cdk deploy --all``` 2. Wait until the AWS CloudFormation stack deployment is complete.	General AWS

Validate the deployment

Task	Description	Skills required
Confirm that the web ACLs successfully deployed.	1. Sign in to the AWS Management Console, and then open the [AWS WAF console](https://console.aws.amazon.com/wafv2/). 2. In the navigation pane, choose **Web ACLs**. 3. In the list of AWS Regions, choose **Global (CloudFront)**. 4. Verify that the new CloudFront web ACL was deployed, and confirm that it has the IP address	General AWS

Task	Description	Skills required
	and geolocation rules that you defined. The default name of this web ACL is `WebACLCloudfront-<ID>` . 5. In the list of AWS Regions, choose the Region in which you deployed the stack. 6. Verify that a new web ACL for API Gateway resources was deployed. Confirm that it has the IP address and geolocation rules that you defined. The default name of this web ACL is `WebACLApiGW-<ID>` .	
(Optional) Associate the web ACLs to your resources.	Associate the AWS WAF web ACLs with your AWS resources , such as an Application Load Balancer, API Gateway, or CloudFront distribution. For instructions, see Associating or disassociating a web ACL with an AWS resource. For an example, see class CfnWebACLAssociation (construct) in the AWS CDK documentation.	General AWS

Clean up resources

Task	Description	Skills required
Delete the stacks.	1. Disassociate the web ACL from any AWS resources . For instructions, see the AWS WAF documentation. 2. In the AWS CDK CLI, enter the following command to delete the AWS CDK application. `cdk destroy --all`	General AWS

Related resources

- API Reference (AWS CDK documentation)
- aws-cdk-lib.aws_wafv2 module (AWS CDK documentation)
- Working with web ACLs (AWS WAF documentation)
- Managing your own rule groups (AWS WAF documentation)
- Rules (AWS WAF documentation)

Scan Git repositories for sensitive information and security issues by using git-secrets

Saurabh Singh, Amazon Web Services

Summary

This pattern describes how to use the open-source git-secrets tool from AWS Labs to scan Git source repositories and find code that might potentially include sensitive information, such as user passwords or AWS access keys, or that has any other security issues.

`git-secrets` scans commits, commit messages, and merges to prevent sensitive information such as secrets from being added to your Git repositories. For example, if a commit, commit message, or any commit in a merge history matches one of your configured, prohibited regular expression patterns, the commit is rejected.

Prerequisites and limitations

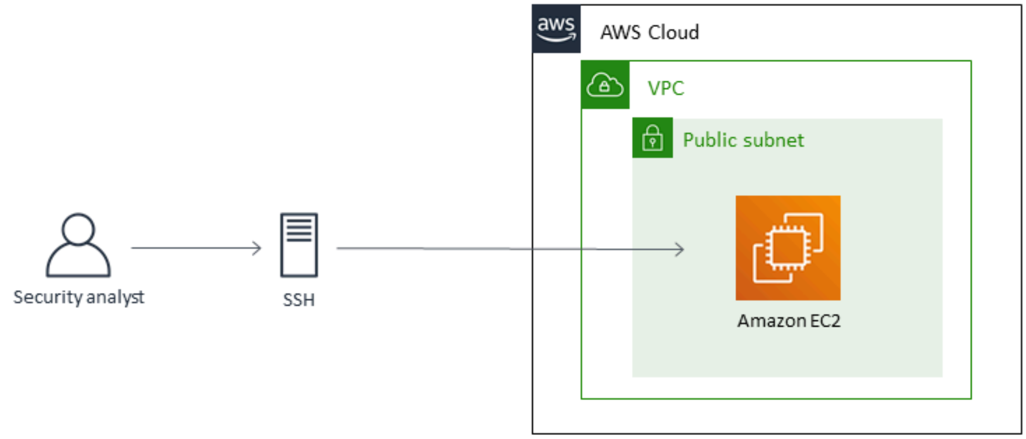
Prerequisites

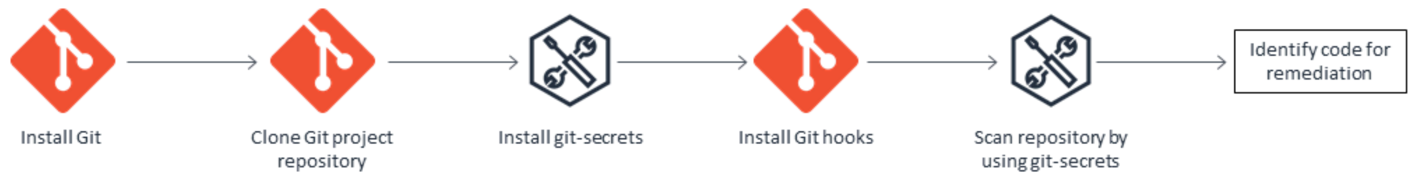
- An active AWS account
- A Git repository that requires a security scan
- A Git client (version 2.37.1 and later) installed

Architecture

Target architecture

- Git
- `git-secrets`





Tools

- git-secrets is a tool that prevents you from committing sensitive information into Git repositories.
- Git is an open-source distributed version control system.

Best practices

- Always scan a Git repository by including all revisions:

```
git secrets --scan-history
```

Epics

Connect to an EC2 instance

Task	Description	Skills required
Connect to an EC2 instance by using SSH.	Connect to an Amazon Elastic Compute Cloud (Amazon EC2) instance by using SSH and a key pair file. You can skip this step if you are scanning a repository on your local machine.	General AWS

Install Git

Task	Description	Skills required
Install Git.	Install Git by using the command: `yum install git -y` If you are using your local machine, you can install a Git client for a specific OS version. For more informati on, see the [Git website]().	General AWS

Clone the source repository and install git-secrets

Task	Description	Skills required
Clone the Git source repositor y.	To clone the Git repository that you want to scan, choose	General AWS

Task	Description	Skills required
	the **Git clone** command from your home directory.	
Clone git-secrets.	Clone the `git-secrets` Git repository. `git clone https://github.com/awslabs/git-secrets.git` Place `git-secrets` somewhere in your `PATH` so that Git picks it up when you run `git-secrets` .	General AWS

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Install git-secrets.</td><td>

For Unix and variants (Linux/macOS):

You can use the <code>install</code> target of the <code>Makefile</code> (provided in the <code>git-secrets</code> repository) to install the tool. You can customize the installation path by using the <code>PREFIX</code> and <code>MANPREFIX</code> variables.

<pre><code>make install</code></pre>

For Windows:

Run the PowerShell <code>install.ps1</code> script provided in the <code>git-secrets</code> repository. This script copies the installation files to an installation directory (<code>%USERPROFILE%/.git-secrets</code> by default) and adds the directory to the current user PATH.

<pre><code>PS > ./install.ps1</code></pre>

For Homebrew (macOS users):

Run:

</td><td>General AWS</td></tr>
</table>

Task	Description	Skills required
	`brew install git-secrets`	

Scan git code repository

Task	Description	Skills required
Go to the source repository.	Switch to the directory for the Git repository that you want to scan: `cd my-git-repository`	General AWS
Register the AWS rule set (Git hooks).	To configure `git-secrets` to scan your Git repository on each commit, run the command: `git secrets --register-aws`	General AWS
Scan the repository.	Run the following command to start scanning your repository: `git secrets --scan`	General AWS
Review the output file.	The tool generates an output file if it finds a vulnerability in your Git repository. For example:	General AWS

Task	Description	Skills required
	``` example.sh:4:AWS_S ECRET_ACCESS_KEY = ********* [ERROR] Matched one or more prohibited patterns Possible mitigations: - Mark false positives as allowed using: git config --add secrets.a llowed ... - Mark false positives as allowed by adding regular expressions to .gitallowed at repository's root directory - List your configure d patterns: git config --get-all secrets.p atterns - List your configure d allowed patterns: git config --get-all secrets.allowed - List your configure d allowed patterns in .gitallowed at repository's root directory - Use --no-verify if this is a one-time false positive ```	

Related resources

- git-secrets tool

Secure file transfers by using Transfer Family, Amazon Cognito, and GuardDuty

Manoj Kumar, Amazon Web Services

Summary

This solution helps you securely transfer files through an SFTP server by using AWS Transfer Family. It includes automated malware scanning capabilities through [Malware Protection for S3](), a feature of Amazon GuardDuty. It is designed for organizations that need to securely exchange files with external parties and validate that all incoming files are scanned for malware before being processed.

The infrastructure as code (IaC) templates provided with this pattern help you deploy the following:

- A secure SFTP server with Amazon Cognito authentication through AWS Lambda
- Amazon Simple Storage Service (Amazon S3) buckets for uploads and incoming files that have been scanned for malware
- A virtual private cloud (VPC)-based architecture with public and private subnets across multiple Availability Zones
- IP-based access control for both ingress and egress traffic, with configurable allow and deny lists
- Automated malware scanning through GuardDuty
- Intelligent file routing based on scan results through Amazon EventBridge and Lambda
- Real-time notifications for security incidents through Amazon Simple Notification Service (Amazon SNS)
- Encryption for Amazon S3 buckets and Lambda environment variables through AWS Key Management Service (AWS KMS)
- Amazon Virtual Private Cloud (Amazon VPC) endpoints for access without internet exposure
- Comprehensive logging through Amazon CloudWatch integration

Prerequisites and limitations

Prerequisites

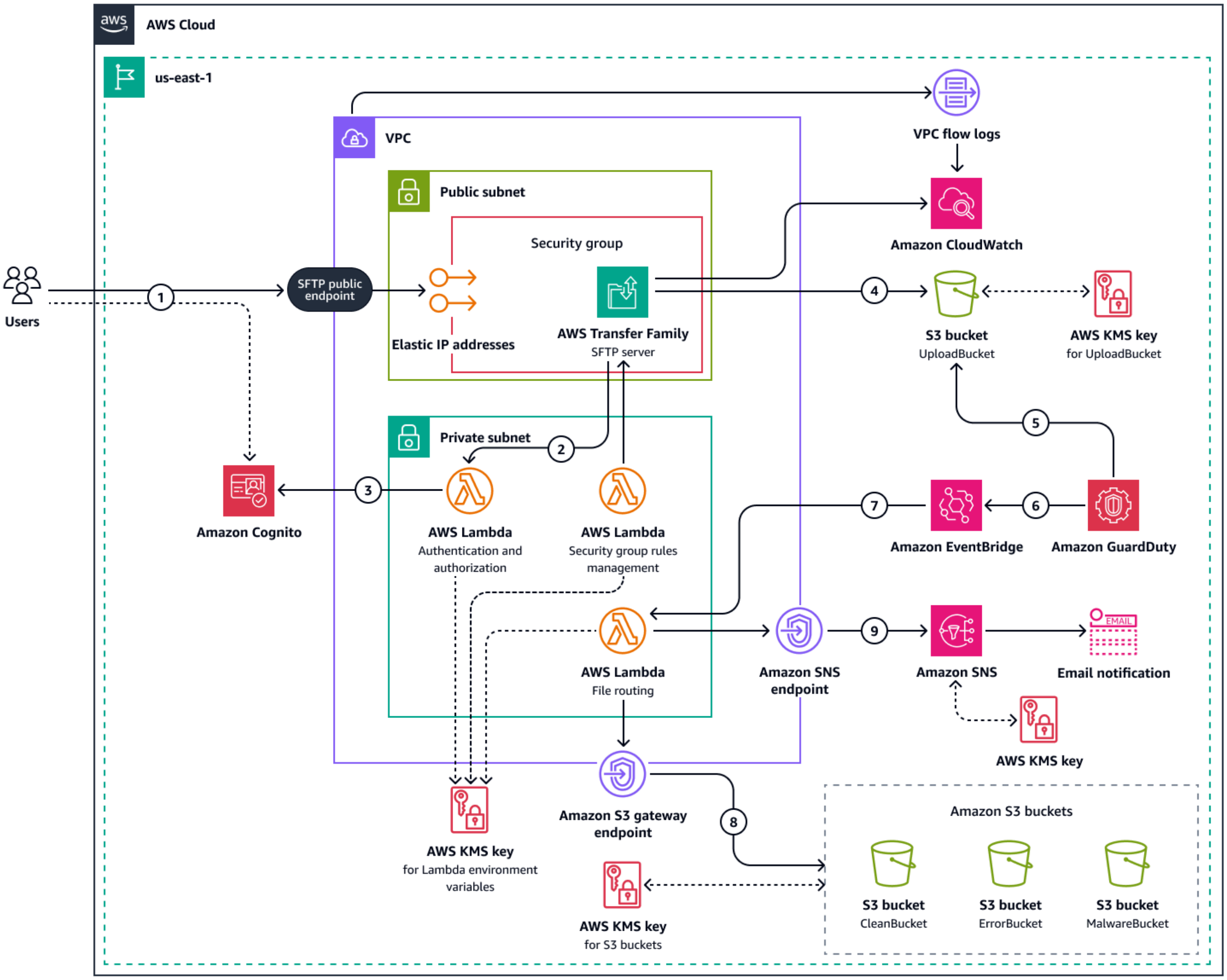
- An active AWS account
- Permissions in AWS Identity and Access Management (IAM) to perform the actions described in this pattern, including deploying AWS CloudFormation templates that provision IAM roles
- GuardDuty, enabled in the target account
- Malware Protection for S3, enabled in the target account
- Service quotas allow you to create the following in the target account:
  - One VPC
  - One private subnet
  - One public subnet
  - Three elastic IP addresses
  - Sufficient Lambda concurrency limits
- A valid email address for security-related notifications
- (Optional) A list of IP addresses or CIDR ranges that you want to allow or deny
- (Optional) AWS Command Line Interface (AWS CLI), installed and configured

Limitations

- Malware Protection for S3 is subject to quotas, such as maximum file sizes. For more information, see Quotas in Malware Protection for S3 and Supportability of Amazon S3 features in the GuardDuty documentation.
- This solution uses Amazon Cognito username and password authentication only. Certificate-based or other authentication methods are not supported in this template. By default, this solution does not configure multi-factor authentication (MFA).
- The solution implements IP-based access control through security groups only.

Architecture

The following architecture diagram shows the resources that are deployed in this pattern. This solution uses Amazon Cognito for user authentication and authorization. An AWS Transfer Family SFTP server is used for file uploads. Files are stored in Amazon S3 buckets, and Amazon GuardDuty scans the files for malware. Amazon SNS sends an email notification if malware is detected.



The diagram shows the following workflow:

1. A user connects to the SFTP server endpoint in AWS Transfer Family. This initiates the authentication process with the Amazon Cognito user pool.
2. A Lambda function initiates the authentication and authorization process and validates the user's credentials with Amazon Cognito.
3. The Lambda function returns the `UploadBucket` Amazon S3 bucket as the home directory. The user assumes the IAM role for the Transfer Family server, and the Lambda function notifies the user that they have been successfully authenticated.
4. The user uploads a file to the Transfer Family SFTP server. The file is stored in the `UploadBucket` Amazon S3 bucket.

5. GuardDuty scans the file for malware. The potential scan results are `NO_THREATS_FOUND`, `THREATS_FOUND`, `UNSUPPORTED`, `ACCESS_DENIED`, and `FAILED`. For sample results, see [S3 object scan result](#) in the GuardDuty documentation.
6. An EventBridge rule detects the scan result event.
7. EventBridge initiates the file-routing Lambda function.
8. The Lambda function processes the event and filters the files based on the scan results as follows:
   - Files that have a `NO_THREATS_FOUND` scan result are sent to the `CleanBucket` Amazon S3 bucket.
   - Files that have a `THREATS_FOUND` scan result are sent to the `MalwareBucket` Amazon S3 bucket.
   - Files that have an `UNSUPPORTED` scan result are sent to the `ErrorBucket` Amazon S3 bucket.
   - Files that have an `ACCESS_DENIED` scan result are sent to the `ErrorBucket` Amazon S3 bucket.
   - Files that have a `FAILED` scan result are sent to the `ErrorBucket` Amazon S3 bucket.

   All files are encrypted with an AWS KMS key.
9. If a file was sent to the `MalwareBucket` Amazon S3 bucket, the Lambda function initiates an Amazon SNS topic. The Amazon SNS topic sends an email notification to an email address that you configure.

Tools

AWS services

- [Amazon CloudWatch](#) helps you monitor the metrics of your AWS resources and the applications you run on AWS in real time.
- [Amazon Cognito](#) provides authentication, authorization, and user management for web and mobile apps.
- [Amazon EventBridge](#) is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- [Amazon GuardDuty](#) is a continuous security monitoring service that analyzes and processes logs to identify unexpected and potentially unauthorized activity in your AWS environment.

- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- Amazon Simple Notification Service (Amazon SNS) helps you coordinate and manage the exchange of messages between publishers and clients, including web servers and email addresses.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS Transfer Family helps you transfer files into and out of AWS storage services over the SFTP, FTPS, or FTP protocols.
- Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Code repository

The code for this pattern is available in the GitHub AWS Transfer Family and GuardDuty Malware Scanning Solution repository.

Best practices

The CloudFormation template provided is designed to incorporate many AWS best practices, such as least-privilege permissions for IAM roles and policies, encryption at rest and in transit, and automatic key rotation. For production environments, consider implementing the following additional recommendations:

- Enable MFA for Amazon Cognito users
- Implement AWS Shield for distributed denial of service (DDoS) protection
- Configure AWS Config for continuous compliance monitoring
- Implement AWS CloudTrail for comprehensive API logging
- Set up Amazon GuardDuty for threat detection beyond malware scanning
- Implement AWS Security Hub for centralized security management

- Use AWS Secrets Manager for credential management
- Implement network traffic monitoring with Traffic Mirroring
- Configure Amazon Macie for sensitive data discovery and protection in Amazon S3
- Implement regular security assessments and penetration testing
- Establish a formal incident response plan
- Implement automated patching for all components
- Conduct regular security training for administrators
- Set up AWS Organizations for multi-account security management

Epics

Deploy the resources

Task	Description	Skills required
Clone the repository.	Enter the following command to clone the AWS Transfer Family and GuardDuty malware scanning solution repository to your local workstation: `git clone https://github.com/aws-samples/sample-secure-transfer-family-code.git`	App developer, DevOps engineer
Create the CloudFormation stack.	1. Follow the instructions in the CloudFormation documentation to create a stack through the CloudFormation console or through the AWS CLI. Configure the following parameters:	Cloud administrator, DevOps engineer

Task	Description	Skills required
	• `UploadBucketName` - Upload bucket name • `CleanBucketName` - Clean files bucket name • `MalwareBucketName` - Malware files bucket name • `ErrorBucketName` - Error files bucket name • `SecurityTeamEmail` - Email address for alerts • `VpcCIDR` - CIDR range for VPC • `PublicSubnet1CIDR` - CIDR range for public subnet 1 • `PublicSubnet2CIDR` - CIDR range for public subnet 2 • `PrivateSubnet1CIDR` - CIDR range for private subnet 1 • `PrivateSubnet2CIDR` - CIDR range for private subnet 2 • `AllowedIPAddresses` - Allowed IP addresses or CIDR ranges • `EnableEgressRules` – If you want to enable egress rules for security groups, enter `true`.	

Task	Description	Skills required
	If you want disable the egress rules, enter `false`. 2. Wait for the stack deployment to complete. 3. In the stack creation outputs, make note of the SFTP server endpoint. You need this value later.	

Configure the resources

Task	Description	Skills required
Turn on malware protection.	1. Sign in to the AWS Management Console and open the [GuardDuty console](https://console.aws.amazon.com/guardduty/). 2. In the navigation pane, choose **Malware Protection for S3**. 3. In the **Protected buckets** section, choose **Enable**. 4. Under **Enter S3 bucket details**, enter the **Amazon S3 bucket** name. Alternatively, choose **Browse S3** to select a bucket. Choose the `UploadBucket` Amazon S3 bucket. 5. Under **Prefix**, select **All the objects in the S3 bucket**.	Cloud administrator, AWS administrator

Task	Description	Skills required
	6. Under **Tag scanned objects**, select **Tag objects**. 7. In the **Service access** section, choose **Create and use a new service role**. 8. Choose **Enable**.	
Add users to the user pool.	Add one or more users to the Amazon Cognito user pool. For instructions, see [Managing users in your user pool](#) in the Amazon Cognito documentation.	Cloud administrator, AWS administrator

Test the SFTP server

Task	Description	Skills required
Connect to the SFTP server endpoint.	1. Enter the following command to get the SFTP server endpoint, where `<stack-name>` is the name of your CloudFormation stack: `TRANSFER_ENDPOINT=$(aws cloudformation describe-stacks \ --stack-name <stack-name> \ --query 'Stacks[0].Outputs[?OutputKey==`TransferServerEndpoint`].OutputValue' \`	App developer, Cloud administrator, Cloud architect, DevOps engineer

Task	Description	Skills required

	`--output text)`
	2. Enter the following command to connect to the SFTP server:
	`sftp user@company.com@$TRANSFER_ENDPOINT`
	3. Validate that you receive the following prompt:
	`sftp>`
	This indicates that you can now enter commands to perform file transfers.	

Troubleshooting

Issue	Solution
User authentication fails	1. Verify that the user exists in the Amazon Cognito user pool. 2. Validate that the user is confirmed and enabled. 3. Make sure the user is providing the correct password. 4. Check for errors in CloudWatch Logs. For a list of AWS CLI commands that can help you perform these troubleshooting steps, see

Issue	Solution
	Useful commands for troubleshooting in the Additional information section.
SFTP authentication fails	1. Verify that the user exists in the Amazon Cognito user pool. 2. Validate that the user is confirmed and enabled. 3. Make sure the user is providing the correct password. 4. Verify that the IP address is allowed in the security group for the AWS Transfer Family server. 5. Check for errors in the logs for the authentication Lambda function. 6. Verify that the IAM role for AWS Transfer Family has correct permissions. For a list of AWS CLI commands that can help you perform these troubleshooting steps, see *Useful commands for troubleshooting* in the Additional information section.

Issue	Solution
File upload access denied	1. Verify that the AWS KMS key permissions are properly configured. 2. Check the Amazon S3 bucket policies. 3. Confirm that the IAM role permissions are properly configured. 4. Verify that the VPC endpoints are properly configured 5. Verify that the IAM role for AWS Transfer Family has correct permissions. For a list of AWS CLI commands that can help you perform these troubleshooting steps, see *Useful commands for troubleshooting* in the [Additional information] section.
No malware scanning	1. Enable GuardDuty in your account. 2. Enable Malware Protection for S3. 3. Verify that the Amazon EventBridge rule is active. 4. Check for errors in the logs for the file-rout ing Lambda function. 5. Verify that notifications are configured correctly in Amazon SNS. For a list of AWS CLI commands that can help you perform these troubleshooting steps, see *Useful commands for troubleshooting* in the [Additional information] section.

Issue	Solution
Lambda function errors	1. Check the VPC configuration. 2. Verify that the VPC endpoints are properly configured. 3. Verify that the IAM role for AWS Transfer Family has correct permissions. 4. Check for errors in CloudWatch Logs. 5. Verify that the AWS KMS key permissions are properly configured. For a list of AWS CLI commands that can help you perform these troubleshooting steps, see *Useful commands for troubleshooting* in the [Additional information](#) section.

Related resources

- [Transfer Family web apps](#) (Transfer Family documentation)

Additional information

Useful commands for troubleshooting

Check the status of a CloudFormation stack:

```
aws cloudformation describe-stacks \
  --stack-name <STACK_NAME>
```

List all users in an Amazon Cognito user pool:

```
aws cognito-idp list-users \
  --user-pool-id <USER_POOL_ID>
```

View logs for Lambda functions:

```
aws logs describe-log-groups \
  --log-group-name-prefix /aws/lambda/
```

Check the status of GuardDuty:

```
aws guardduty list-detectors
```

Check security group rules:

```
aws ec2 describe-security-groups \
  --group-ids <SECURITY_GROUP_ID> \
  --output table
```

Check the status of the AWS Transfer Family server:

```
aws transfer describe-server \
  --server-id <SERVER_ID>
```

List all files in an Amazon S3 bucket:

```
aws s3 ls s3://<BUCKET_NAME>/ \
  --recursive
```

Check the status of an EventBridge rule:

```
aws events describe-rule \
  --name <RULE_NAME>
```

Send alerts from AWS Network Firewall to a Slack channel

Venki Srivatsav and Aromal Raj Jayarajan, Amazon Web Services

Summary

This pattern describes how to deploy a firewall by using Amazon Web Services (AWS) Network Firewall with the distributed deployment model and how to propagate the alerts generated by AWS Network Firewall to a configurable Slack channel.

Compliance standards such as Payment Card Industry Data Security Standard (PCI DSS) require that you install and maintain a firewall to protect customer data. In the AWS Cloud, a virtual private cloud (VPC) is considered the same as a physical network in the context of these compliance requirements. You can use Network Firewall to monitor network traffic between VPCs and to protect your workloads that run in VPCs governed by a compliance standard. Network Firewall blocks access or generates alerts when it detects unauthorized access from other VPCs in the same account. However, Network Firewall supports a limited number of destinations for delivering the alerts. These destinations include Amazon Simple Storage Service (Amazon S3) buckets, Amazon CloudWatch log groups, and Amazon Data Firehose delivery streams. Any further action on these notifications requires offline analysis by using either Amazon Athena or Amazon Kinesis.

This pattern provides a method for propagating alerts that are generated by Network Firewall to a configurable Slack channel for further action in near real time. You can also extend the functionality to other alerting mechanisms such as PagerDuty, Jira, and email. (Those customizations are outside the scope of this pattern.)

Prerequisites and limitations

Prerequisites

- Slack channel (see Getting started in the Slack help center)
- Required privileges to send a message to the channel
- The Slack endpoint URL with an API token (select your app and choose an incoming webhook to see its URL; for more information, see Creating an Incoming Webhook in the Slack API documentation)
- An Amazon Elastic Compute Cloud (Amazon EC2) test instance in the workload subnets
- Test rules in Network Firewall

- Actual or simulated traffic to trigger the test rules
- An S3 bucket to hold the source files to be deployed

Limitations

- Currently this solution supports only a single Classless Inter-Domain Routing (CIDR) range as a filter for source and destination IPs.

Architecture

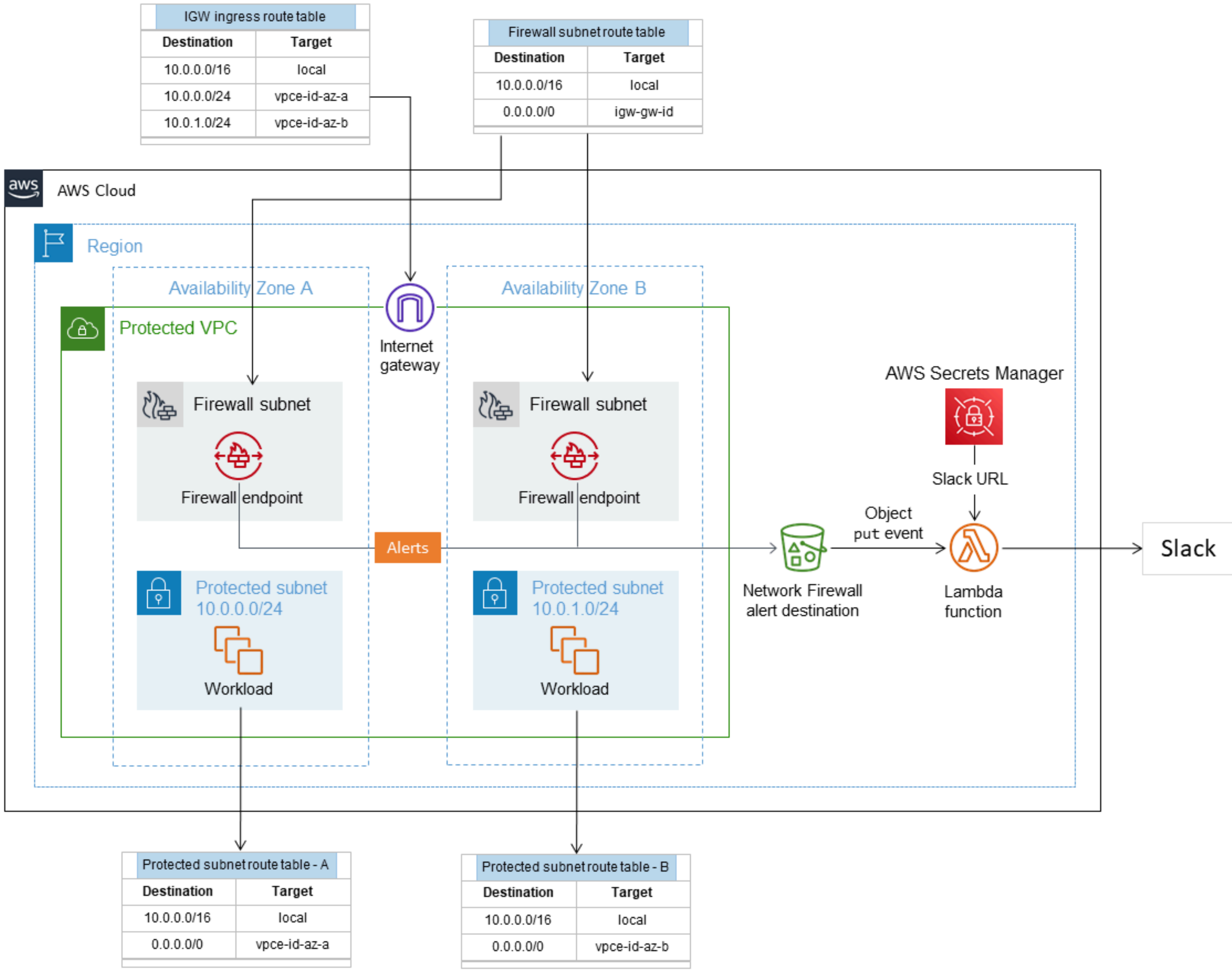
Target technology stack

- One VPC
- Four subnets (two for the firewall and two for workloads)
- Internet gateway
- Four route tables with rules
- S3 bucket used as an alert destination, configured with a bucket policy and event settings to run a Lambda function
- Lambda function with an execution role, to send Slack notifications
- AWS Secrets Manager secret for storing the Slack URL
- Network firewall with alert configuration
- Slack channel

All components except for the Slack channel are provisioned by the CloudFormation templates and the Lambda function that are provided with this pattern (see the [Code](#)section).

Target architecture

This pattern sets up a decentralized network firewall with Slack integration. This architecture consists of a VPC with two Availability Zones. The VPC includes two protected subnets and two firewall subnets with network firewall endpoints. All traffic going into and out of the protected subnets can be monitored by [creating firewall policies](#) and rules. The network firewall is configured to place all alerts in an S3 bucket. This S3 bucket is configured to call a Lambda function when it receives a `put` event. The Lambda function fetches the configured Slack URL from Secrets Manager and sends the notification message to the Slack workspace.



For more information about this architecture, see the AWS blog post [Deployment models for AWS Network Firewall](#).

Tools

AWS services

- [AWS Network Firewall](#) is a stateful, managed, network firewall and intrusion detection and prevention service for VPCs in the AWS Cloud. You can use Network Firewall to filter traffic at the perimeter of your VPC and protect your workloads on AWS.
- [AWS Secrets Manager](#) is a service for credential storage and retrieval. Using Secrets Manager, you can replace hardcoded credentials in your code, including passwords, with an API call to Secrets

Manager to retrieve the secret programmatically. This pattern uses Secrets Manager to store the Slack URL.

- [Amazon Simple Storage Service (Amazon S3)](https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. This pattern uses Amazon S3 to store the CloudFormation templates and Python script for the Lambda function. It also uses an S3 bucket as the network firewall alert destination.
- [AWS CloudFormation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html) helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. This pattern uses AWS CloudFormation to automatically deploy a distributed architecture for Firewall Manager.

Code

The code for this pattern is available on GitHub, in the [Network Firewall Slack Integration](https://github.com/aws-samples/aws-network-firewall-automation-examples) repository. In the `src` folder of the repository, you'll find:

- A set of CloudFormation files in YAML format. You use these templates to provision the components for this pattern.
- A Python source file (`slack-lambda.py`) to create the Lambda function.
- A .zip archive deployment package (`slack-lambda.py.zip`) to upload your Lambda function code.

To use these files, follow the instructions in the next section.

Epics

Set up the S3 bucket

Task	Description	Skills required
Create an S3 bucket.	1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.	App developer, App owner, Cloud administrator

Task	Description	Skills required
	2. Choose or create an S3 bucket to host the code. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes. We recommend that you use a prefix to organize the code for this pattern. For more information, see Creating a bucket in the Amazon S3 documentation.	
Upload the CloudFormation templates and Lambda code.	1. Download the following files from the GitHub repository for this pattern: • `base.yml` • `igw-ingress-route.yml` • `slack-lambda.py` • `slackLambda.yml` • `decentralized-deployment.yml` • `protected-subnet-route.yml` • `slack-lambda.py.zip` 2. Upload the files to the S3 bucket you created.	App developer, App owner, Cloud administrator

Deploy the CloudFormation template

Task	Description	Skills required
Launch the CloudFormation template.	Open the [AWS CloudFormation console]() in the same AWS Region as your S3 bucket and deploy the template `base.yml`. This template creates the required AWS resources and Lambda functions for the alerts to be transmitted to the Slack channel. For more information about deploying CloudFormation templates, see [Creating a stack on the AWS CloudFormation console]() in the CloudFormation documentation.	App developer, App owner, Cloud administrator
Complete the parameters in the template.	Specify the stack name and configure the parameter values. For a list of parameters, their descriptions, and default values, see *CloudFormation parameters* in the [Additional information]() section.	App developer, App owner, Cloud administrator
Create the stack.	1. Review stack details and update values based on your environment requirements.	App developer, App owner, Cloud administrator

Task	Description	Skills required
	2. Choose **Create stack** to deploy the template.	

Verify the solution

Task	Description	Skills required
Test the deployment.	Use the AWS CloudFormation console or the AWS Command Line Interface (AWS CLI) to verify that the resources listed in the [Target technology stack](#) section have been created. If the CloudFormation template fails to deploy successfully, check the values you provided for the `pAvailabilityZone1` and `pAvailabilityZone2` parameters. These should be appropriate for the AWS Region you're deploying the solution in. For a list of Availability Zones for each Region, see [Regions and Zones](#) in the Amazon EC2 documentation.	App developer, App owner, Cloud administrator
Test functionality.	1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.	App developer, App owner, Cloud administrator

Task	Description	Skills required

	2. Create an EC2 instance in one of the protected subnets. Choose an Amazon Linux 2 AMI (HVM) to use as an HTTPS server. For instructions, see [Launch an instance](#) in the Amazon EC2 documentation.
	
	Note
	Amazon Linux 2 is nearing end of support. For more information, see the [Amazon Linux 2 FAQs](#).
	
	3. Use the following user data to install a web server on the EC2 instance:
	
	`#!/bin/bash`
	`yum install httpd -y`
	`systemctl start httpd`
	`systemctl stop firewalld`
	`cd /var/www/html`
	`echo "Hello!! this is a NFW alert test page, 200 OK" > index.html`
	
	4. Create the following network firewall rules:
	
	Stateless rule:	

Task	Description	Skills required
	``` Source: 0.0.0.0/0 Destination 10.0.3.65 /32 (private IP of the EC2 instance) Action: Forward ``` *Stateful rule:* ``` Protocol: HTTP Source ip/port: Any / Any Destination ip/port: Any /Any ``` 5. Get the public IP of the web server you created in step 3. 6. Access the public IP in a browser. You should see the following message in the browser: ``` Hello!! this is a NFW alert test page, 200 OK ``` You will also get a notificat ion in the Slack channel. The notification might be delayed, depending on the size of the message. For testing purposes, consider providing a CIDR filter that is not too narrow (for example, a CIDR value with /32 would	

Task	Description	Skills required
	be considered too narrow, and /8 would be too broad). For more information, see the *Filter behavior* section in [Additional information](#).	

Related resources

- [Deployment models for AWS Network Firewall](#) (AWS blog post)
- [AWS Network Firewall policies](#) (AWS documentation)
- [Network Firewall Slack Integration](#) (GitHub repository)
- [Create a Slack workspace](#) (Slack help center)

Additional information

CloudFormation parameters

Parameter	Description	Default or sample value
`pVpcName`	The name of the VPC to create.	Inspection
`pVpcCidr`	The CIDR range for the VPC to create.	10.0.0.0/16
`pVpcInstanceTenancy`	How EC2 instances are distributed across physical hardware. Options are `default` (shared tenancy) or `dedicated` (single tenancy).	default
`pAvailabilityZone1`	The first Availability Zone for the infrastructure.	us-east-2a

`pAvailabilityZone2`	The second Availability Zone for the infrastructure.	us-east-2b
`pNetworkFirewallSubnet1Cidr`	The CIDR range for the first firewall subnet (minimum /28).	10.0.1.0/24
`pNetworkFirewallSubnet2Cidr`	The CIDR range for the second firewall subnet (minimum /28).	10.0.2.0/24
`pProtectedSubnet1Cidr`	The CIDR range for the first protected (workload) subnet.	10.0.3.0/24
`pProtectedSubnet2Cidr`	The CIDR range for the second protected (workload) subnet.	10.0.4.0/24
`pS3BucketName`	The name of the existing S3 bucket where you uploaded the Lambda source code.	us-w2-yourname-lambda-functions
`pS3KeyPrefix`	The prefix of the S3 bucket where you uploaded the Lambda source code.	aod-test
`pAWSSecretName4Slack`	The name of the secret that holds the Slack URL.	SlackEnpoint-Cfn
`pSlackChannelName`	The name of the Slack channel you created.	somename-notifications
`pSlackUserName`	Slack user name.	Slack User
`pSecretKey`	This can be any key. We recommend that you use the default.	webhookUrl

`pWebHookUrl`	The value of the Slack URL.	https://hooks.slack.com/services/T???9T??/A031885JRM7/9D4Y??????
`pAlertS3Bucket`	The name of the S3 bucket to be used as the network firewall alert destination. This bucket will be created for you.	us-w2-yourname-security-aod-alerts
`pSecretTagName`	The tag name for the secret.	AppName
`pSecretTagValue`	The tag value for the specified tag name.	LambdaSlackIntegration
`pdestCidr`	The filter for the destination CIDR range. For more information, see the next section, *Filter behavior*.	10.0.0.0/16
`pdestCondition`	A flag to indicate whether to exclude or include the destination match. For more information, see the next section. Valid values are `include` and `exclude`.	include
`psrcCidr`	The filter for the source CIDR range to alert. For more information, see the next section.	118.2.0.0/16
`psrcCondition`	The flag to exclude or include the source match. For more information, see the next section.	include

Filter behavior

If you haven't configured any filters in AWS Lambda, all generated alerts are sent to your Slack channel. The source and destination IPs of the generated alerts are matched against the CIDR ranges you configured when you deployed the CloudFormation template. If a match is found, the condition is applied. If either the source or the destination falls within the configured CIDR range and at least one of them is configured with the condition `include`, an alert is generated. The following tables provide examples of CIDR values, conditions, and results.

	Configured CIDR	Alert IP	Configured	Alert
Source	10.0.0.0/16	10.0.0.25	include	Yes
Destination	100.0.0.0/16	202.0.0.13	include	

	Configured CIDR	Alert IP	Configured	Alert
Source	10.0.0.0/16	10.0.0.25	exclude	No
Destination	100.0.0.0/16	202.0.0.13	include	

	Configured CIDR	Alert IP	Configured	Alert
Source	10.0.0.0/16	10.0.0.25	include	Yes
Destination	100.0.0.0/16	100.0.0.13	include	

	Configured CIDR	Alert IP	Configured	Alert
Source	10.0.0.0/16	90.0.0.25	include	Yes
Destination	Null	202.0.0.13	include	

	Configured CIDR	Alert IP	Configured	Alert
Source	10.0.0.0/16	90.0.0.25	include	No

Destination	100.0.0.0/16	202.0.0.13	include

Send custom attributes to Amazon Cognito and inject them into tokens

Carlos Alessandro Ribeiro and Mauricio Mendoza, Amazon Web Services

Summary

Sending custom attributes to an Amazon Cognito authentication process can provide additional context to an application, enable more granular access controls, and make it easier to manage user profiles and authentication requirements. These features are useful in a wide range of applications and scenarios, and they can help you improve the overall security and functionality of an application.

This pattern shows how to send custom attributes to an Amazon Cognito authentication process when an application needs to provide additional context to the access token or identity (ID) token. You use the Node.js as the backend application. The application authenticates a user from an Amazon Cognito user pool and passes custom attributes that are needed for token generation. You can use AWS Lambda triggers for Amazon Cognito to customize your authentication process without major code customization or significant effort.

Important

The code and samples in this pattern are not recommended for production workloads because they are intended for demonstration purposes only. For production workloads, additional configuration is required on the client side. Use this pattern as a reference for pilot or proof-of-concept purposes only.

Prerequisites and limitations

Prerequisites

- An active AWS account
- Permissions to create and manage Amazon Cognito user pools and AWS Lambda functions
- AWS Command Line Interface (AWS CLI), installed and configured
- An integrated development environment (IDE) that supports Node.js

- Node.js version 18 or later, installed
- npm version 8 or later, installed
- TypeScript, installed

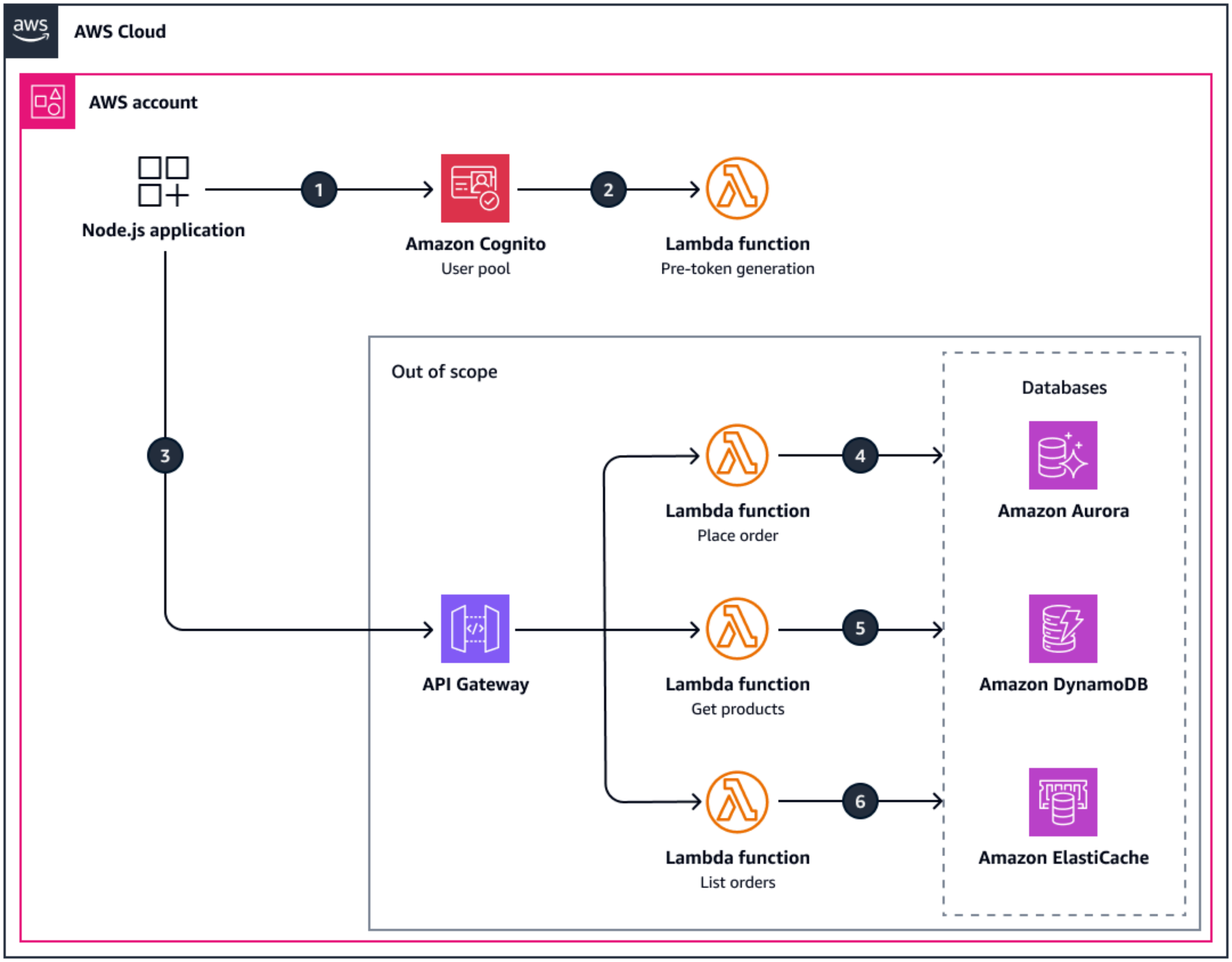
Limitations

- This pattern is not applicable for application integration through the Client Credentials authentication flow.
- The pre-token generation trigger can add or change only some attributes of the access token and identity token. For more information, see Pre token generation Lambda trigger in the Amazon Cognito documentation.

Architecture

Target architecture

The following diagram shows the target architecture for this pattern. It also shows how the Node.js application might work with a backend to update databases. However, the backend database updates are outside the scope of this pattern.



The diagram shows the following workflow:

1. The Node.js application issues an access token with custom attributes to the Amazon Cognito user pool.
2. The Amazon Cognito user pool initiates the pre-token generation Lambda function, which customizes the access and ID tokens.
3. The Node.js application makes an API call through Amazon API Gateway.

> **Note**
>
> The other architectural components shown in this architecture are for example only and are out of scope for this pattern.

Automation and scale

You can automate the provisioning of Amazon Cognito user pools, AWS Lambda functions, database instances, and other resources by using AWS CloudFormation, the AWS Cloud Development Kit (AWS CDK), HashiCorp Terraform, or any supported infrastructure as code (IaC) tool. If you want to scale your deployments, use continuous integration and continuous delivery (CI/CD) pipelines, which help prevent errors associated with manual deployments.

Tools

AWS services

- Amazon API Gateway helps you create, publish, maintain, monitor, and secure REST, HTTP, and WebSocket APIs at any scale.
- Amazon Cognito provides authentication, authorization, and user management for web and mobile apps.
- Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS SDK for JavaScript provides a JavaScript API for AWS services. You can use it to build libraries or applications for Node.js or the browser.

Other tools

- Node.js is an event-driven JavaScript runtime environment that is designed for building scalable network applications.
- npm is a software registry that runs in a Node.js environment and is used to share or borrow packages and manage deployment of private packages.

Best practices

We recommend that you implement the following best practices:

- **Secrets and sensitive data** – Do not store secrets or sensitive data within the application. Use an external system that the application can pull the data from, such as AWS AppConfig, AWS Secrets Manager, or AWS Systems Manager Parameter Store.
- **Standardized deployment** – Use CI/CD pipelines to deploy your applications. You can use services such as AWS CodeBuild and AWS CodePipeline.
- **Token expiration** – Set a short expiration date for the access token.
- **Use a secure connection** – All communication between the client application and the backend should be encrypted by using SSL/TLS. Use AWS Certificate Manager (ACM) to generate and manage SSL/TLS certificates, and use Amazon CloudFront or Elastic Load Balancing to handle SSL/TLS termination.
- **Validate user input** – Make sure that all user input is validated to prevent injection attacks and other security vulnerabilities. Use input validation libraries and services such as Amazon API Gateway and AWS WAF to prevent common attack vectors.
- **Use IAM roles** – Use AWS Identity and Access Management (IAM) roles to control access to AWS resources and make sure that only authorized users have access. Follow the principle of least privilege and make sure that each user has only the necessary permissions to perform their role.
- **Use a password policy** – Configure a password policy that meets your security requirements, such as minimum length, complexity, and expiration. Use Secrets Manager or AWS Systems Manager Parameter Store to store and manage passwords securely.
- **Enable multi-factor authentication (MFA)** – Enable MFA for all users to provide an additional layer of security and reduce the risk of unauthorized access. Use AWS IAM Identity Center or Amazon Cognito to enable MFA and other authentication methods.
- **Store sensitive information securely** – Store sensitive information, such as passwords and access tokens, securely by using AWS Key Management Service (AWS KMS) or other encryption services.
- **Use strong authentication methods** – To increase the security of the authentication process, use strong authentication methods, such as biometric authentication or multi-factor authentication.
- **Monitor for suspicious activity** – Use AWS CloudTrail and other monitoring tools to monitor for suspicious activity and potential security threats. Set up automated alerts for unusual activity, and use Amazon GuardDuty or AWS Security Hub to detect potential threats.
- **Regularly review and update security policies** – Regularly review and update your security policies and procedures to make sure that they meet your changing security requirements

and best practices. Use AWS Config to track and audit changes to your security policies and procedures.

- **Automated sign-up** – Do not enable automated sign-up to an Amazon Cognito user pool. For more information, see Reduce risks of user sign-up fraud and SMS pumping with Amazon Cognito user pools (AWS blog post).

For additional best practices, see Security best practices for Amazon Cognito user pools in the Amazon Cognito documentation.

Epics

Set up the AWS resources

Task	Description	Skills required

Create a user pool.	1. In a CLI terminal, enter the following command to create an Amazon Cognito user pool:
	`aws cognito-idp create-user-pool \ --pool-name <MyUserPool>`
	2. Enter the following command to create a client ID and secret that will be used with the SDK for JavaScript:
	`aws cognito-idp create-user-pool-client \ --user-pool-id <UserPoolID> \ --client-name <MyNewClient> \`
		App developer, AWS DevOps

Task	Description	Skills required
	` --no-generate-secret \ --explicit-auth-flows "USER_PASSWORD_AUTH" "ADMIN_NO_SRP_AUTH"` For more information and instructions for how to set up a user pool in the AWS Management Console, see Getting started with user pools and Add more features and security options to your user pool. **Tip** To reduce costs, use the Essentials plan or Lite plan to test this pattern. For more information, see Amazon Cognito pricing.	

Task	Description	Skills required
Add a user to the user pool.	Enter the following command to create one user in the Amazon Cognito user pool: `aws cognito-idp sign-up \ --client-id <ClientID> \ --username <jane@example.com> \ --password <PASSWORD> \ --user-attributes Name="email",Value="<jane@example.com>" Name="name",Value="<Jane>"`	App developer, AWS DevOps

Task	Description	Skills required
Add the app client to the user pool.	1. Navigate to the [Amazon Cognito console](). 2. Choose **User pools**. 3. Choose the user pool that you created previously. 4. Under **Define your application**, choose the **Application type** that best fits the application scenario that you want to create authentication and authorization services for. 5. In **Name your application**, enter a descriptive name or proceed with the default name. 6. Under **Add a return URL**, enter a redirect path to your application for after users complete authentic ation. 7. Choose **Create your application**. You can set up the advanced options after you create your app client.	AWS systems administrator, AWS administrator, AWS DevOps, App developer

Task	Description	Skills required
Create a Lambda trigger for pre-token generation.	1. Open the [Functions page](#) of the Lambda console. 2. Choose **Create function**. 3. Select **Author from scratch**. 4. In the **Basic informati on** pane, for **Function name**, enter `myPreTokenGenerationLambdaFunction` . 5. For **Runtime**, choose **Node.js 22**. 6. Leave **architecture** set to **x86_64**. 7. Choose **Create function**. 8. On the **Code** tab, under **Code Source**, enter the sample Lambda function from the [Additional resources](#) section of this pattern. This Lambda function injects the custom attributes into the ID token and access token.	AWS DevOps, App developer

Task	Description	Skills required
Customize the user pool workflow.	1. Navigate to the [Amazon Cognito console](#). 2. Choose **User pools**. 3. Choose the user pool that you created previously. 4. In the left navigation pane, under **Authentication**, choose **Extensions**. 5. Choose **Add Lambda trigger**. 6. Add or edit a **Pre token generation trigger**. 7. For **Trigger type**, choose **Authentication**. 8. Under **Authentication**, choose **Pre token generation trigger**. 9. For **Assign Lambda function**, choose `myPreTokenGenerationLambdaFunction` . 10. Choose **Add Lambda trigger**. 11. Choose a **Trigger event version** of **Basic features + access token customization for user identities** or **Basic features + access token customization for user and machine identities**. This setting updates the request	AWS DevOps, App developer

Task	Description	Skills required
	parameters that Amazon Cognito sends to your function to include fields for access token customization. For more information, see [Customizing user pool workflows with Lambda triggers](#) in the Amazon Cognito documentation.	

Create the Node.js application

Task	Description	Skills required

Create the Node.js application.	1. In a terminal, enter the following commands:
	
	`mkdir my-app`
	`cd my-app`
	`npm init -y`
	`npm install --save-dev typescript`
	`npm i --save-dev @types/node`
	`npm i amazon-cognito-identity-js`
	`npx tsc --init`
	`mkdir src`
	`touch src/index.ts`
	
	2. Modify the `package.json` file to include the TypeScript compilation scripts:
		App developer

Task	Description	Skills required

	``
	"scripts": {
	  "start": "node src/index.js",
	  "build": "tsc"
	},
	``	

Implement the authentication logic.	1. Open the `index.ts` file. 2. Paste the sample Typescript provided in the [Additional resources](#) section of this pattern. 3. Save and close the `index.ts` file. **Note** You can create your own TypeScript file or modify the sample provided as needed for your use case.	App developer

Task	Description	Skills required

Configure the environment variables and configuration file.	In a terminal, enter the following commands to create the environment variables:
	`export USERNAME="<COGNITO_USER_NAME>"`
	`export PASSWORD="<COGNITO_USER_PASSWORD>"`
	`export USER_POOL_ID="<COGNITO_USER_ID>"`
	`export CLIENT_ID="<COGNITO_CLIENT_ID>"`
	Important
	Do not hardcode secrets or expose your credentials.
		App developer

Run the application.	Enter the following commands to run the application and confirm that it is working:
	`npm run build`
	`npm start`
		App developer

Task	Description	Skills required
Confirm that the custom attributes are injected into the tokens.	Use the debugging features for your IDE to view the access and ID tokens. Confirm that the custom attribute s were added. For sample tokens, see the Additional information section of this pattern.	App developer

Troubleshooting

Issue	Solution
Invalid client ID when trying to authenticate the user	This error typically occurs when you are using a client ID with a generated client secret. You must create a client ID without a secret attached to it. For more information, see Application-specific settings with app clients.

Related resources

- Customizing user pool workflows with Lambda triggers (Amazon Cognito documentation)
- Pre token generation Lambda trigger (Amazon Cognito documentation)
- CognitoIdentityProviderClient (AWS SDK for JavaScript documentation)
- cognito-idp (AWS CLI documentation)

Additional information

Sample TypeScript file

The following code sample is a TypeScript file that invokes the authentication process by using an AWS SDK to send custom attributes to Amazon Cognito:

```
import * as AmazonCognitoIdentity from "amazon-cognito-identity-js";

const userPoolId: string = process.env.USER_POOL_ID ?? '';
const clientId: string = process.env.CLIENT_ID ?? '';

const poolData = {
  UserPoolId: userPoolId,
  ClientId: clientId
};
const userPool = new AmazonCognitoIdentity.CognitoUserPool(poolData);

export const loginWithCognitoSDK = function (userName: string, password: string) {
  const authenticationDetails = new AmazonCognitoIdentity.AuthenticationDetails({
    Username: userName,
    Password: password,
    ClientMetadata: {
        customGroup: "MyCustomGroup",
        customApplicationData: "Custom data from a custom application"
    }
  });
  const userData = {
    Username: userName,
    Pool: userPool
  };

  const cognitoUser = new AmazonCognitoIdentity.CognitoUser(userData);

  // Authenticate the user using the authenticationDetails object
  cognitoUser.authenticateUser(authenticationDetails, {
    onSuccess: function (result: any) {},
    onFailure: function (err: any) {},
  });
}
loginWithCognitoSDK(process.env.USERNAME ?? '', process.env.PASSWORD ?? '');
```

The sample uses the `AuthenticationDetails` model from the SDK for JavaScript to provide the username, password, and the `ClientMetadada`. After the authentication in Amazon Cognito, the client metadata can be retrieved from the access and ID tokens.

Sample Lambda function

The following code sample is a Lambda function that is linked to the Pre-Generation Token from Amazon Cognito. It helps you customize the access token and ID token that Amazon Cognito uses.

The tokens are passed through the integrations between your architecture. This sample includes a custom claim attribute called `customApplicationData` and a custom group name called `MyCustomGroup`:

```
export const handler = async(event, context, callback) => {
    event.response = {
        claimsOverrideDetails: {
            claimsToAddOrOverride: { customApplicationData:
 event.request.clientMetadata.customApplicationData },
            groupOverrideDetails: { groupsToOverride:
 [event.request.clientMetadata.customGroup] }
        }
    };
    callback(null, event);
};
```

Sample access token

You can decode the access token to visualize the custom attributes that were added. The following is a sample access token:

```
{
  "sub": "6daf331f-4451-48b4-abde-774579299204",
  "cognito:groups": [
    "MyCustomGroup"
  ],
  "iss": "https://cognito-idp.<REGION>.amazonaws.com/<USERPOOL_ID>",
  "client_id": "<YOUR_CLIENT_ID>",
  "origin_jti": "acff7e91-09f9-4fde-8eec-38b0f8c47cdc",
  "event_id": "c5113a9c-1f01-435b-9b73-a5cd3e88514e",
  "token_use": "access",
  "scope": "aws.cognito.signin.user.admin",
  "auth_time": 1677979246,
  "exp": 1677982846,
  "iat": 1677979246,
  "jti": "5c9c2708-a871-4428-bd9b-18ad261bea90",
  "username": "<USER_NAME>"
}
```

Sample ID token

You can decode the access token to visualize the custom attributes that were added. The following is a sample access token:

```
{
  "sub": "6daf331f-4451-48b4-abde-774579299204",
  "cognito:groups": [
    "MyCustomGroup"
  ],
  "iss": "https://cognito-idp.<REGION>.amazonaws.com/<USERPOOL_ID>",
  "cognito:username": "<USER_NAME>",
  "origin_jti": "acff7e91-09f9-4fde-8eec-38b0f8c47cdc",
  "customApplicationData": "Custom data from a custom application",
  "aud": "<YOUR_CLIENT_ID>",
  "event_id": "c5113a9c-1f01-435b-9b73-a5cd3e88514e",
  "token_use": "id",
  "auth_time": 1677979246,
  "exp": 1677982846,
  "iat": 1677979246,
  "jti": "f7ca006b-f25b-44d2-a7a4-6e6423f4201f",
  "email": "<USER_EMAIL>"
}
```

Simplify private certificate management by using AWS Private CA and AWS RAM

Everett Hinckley and Vivek Goyal, Amazon Web Services

Summary

You can use AWS Private Certificate Authority (AWS Private CA) to issue private certificates for authenticating internal resources and signing computer code. This pattern provides an AWS CloudFormation template for the rapid deployment of a multi-level CA hierarchy and consistent provisioning experience. Optionally, you can use AWS Resource Access Manager (AWS RAM) to securely share the CA within your organizations or organizational units (OUs) in AWS Organizations, and centralize the CA while using AWS RAM to manage permissions. There is no need for a private CA in every account, so this approach saves you money. Additionally, you can use Amazon Simple Storage Service (Amazon S3) to store the certificate revocation list (CRL) and access logs.

This implementation provides the following features and benefits:

- Centralizes and simplifies the management of the private CA hierarchy by using AWS Private CA.
- Exports certificates and keys to customer-managed devices on AWS and on premises.
- Uses an AWS CloudFormation template for a rapid deployment and consistent provisioning experience.
- Creates a private root CA along with 1, 2, 3, or 4 subordinate CA hierarchy.
- Optionally, uses AWS RAM to share the end-entity subordinate CA with other accounts at the organization or OU level.
- Saves money by removing the need for a private CA in every account by using AWS RAM.
- Creates an optional S3 bucket for the CRL.
- Creates an optional S3 bucket for CRL access logs.

Prerequisites and limitations

Prerequisites

If you want to share the CA within an AWS Organizations structure, identify or set up the following:

- A security account for creating the CA hierarchy and share.
- A separate OU or account for testing.
- Sharing enabled within the AWS Organizations management account. For more information, see [Enable resource sharing within AWS Organizations] in the AWS RAM documentation.

Limitations

- CAs are regional resources. All CAs reside in a single AWS account and in a single AWS Region.
- User-generated certificates and keys are not supported. For this use case, we recommend that you customize this solution to use an external root CA.
- A public CRL bucket is not supported. We recommend that you keep the CRL private. If internet access to the CRL is required, see the section on using Amazon CloudFront to serve CRLs in [Enabling the S3 Block Public Access (BPA) feature] in the AWS Private CA documentation.
- This pattern implements a single-Region approach. If you require a multi-Region certificate authority, you can implement subordinates in a second AWS Region or on premises. That complexity is outside the scope of this pattern, because the implementation depends on your specific use case, workload volume, dependencies, and requirements.

Architecture

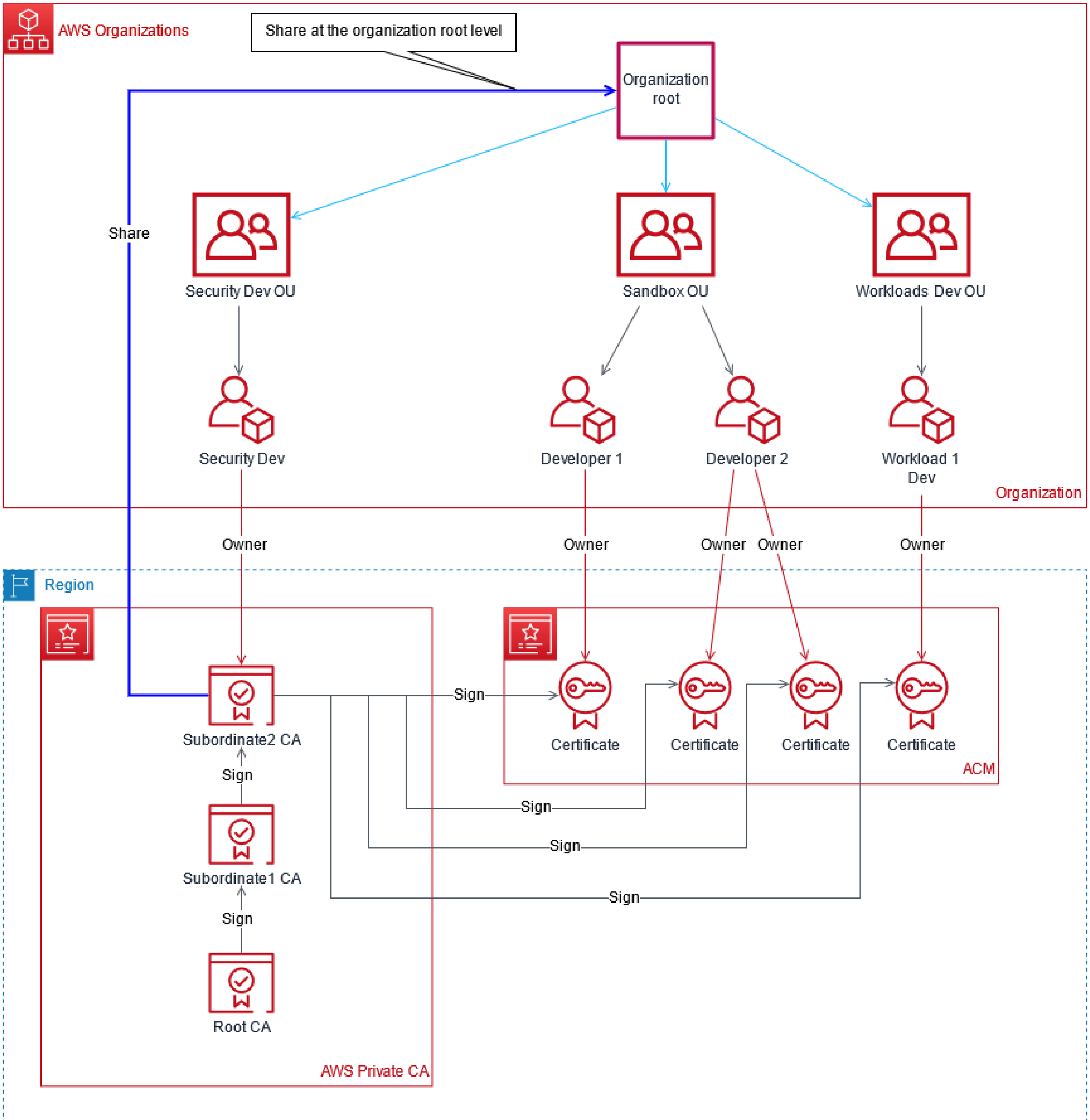
Target technology stack

- AWS Private CA
- AWS RAM
- Amazon S3
- AWS Organizations
- AWS CloudFormation

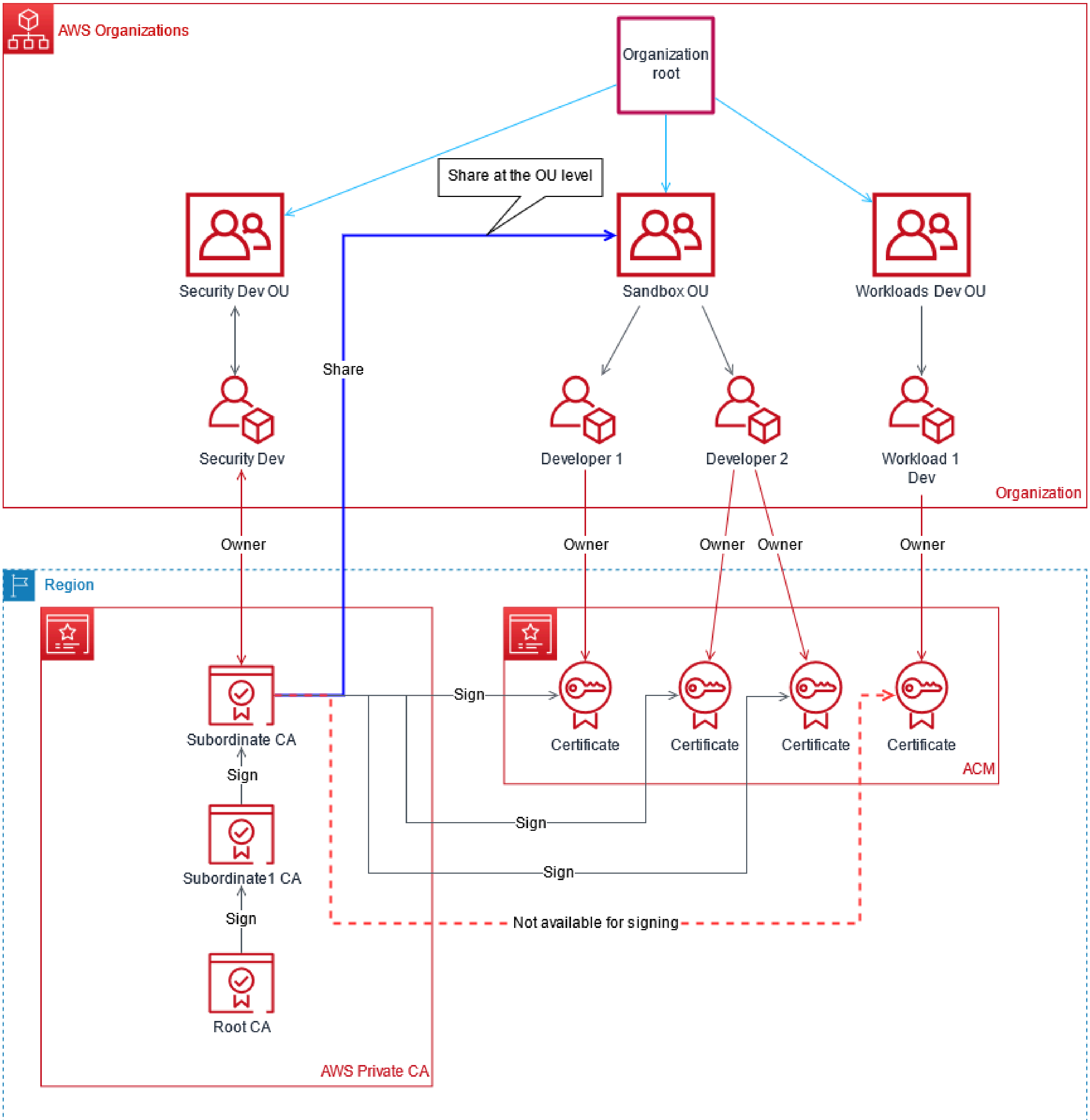
Target architecture

This pattern provides two options for sharing to AWS Organizations:

Option 1 – Create the share at the organization level. All accounts in the organization can issue the private certificates by using the shared CA, as shown in the following diagram.



Option 2 – Create the share at the organizational unit (OU) level. Only the accounts in the specified OU can issue the private certificates by using the shared CA. For example, in the following diagram, if the share is created at the Sandbox OU level, both Developer 1 and Developer 2 can issue private certificates by using the shared CA.



Tools

AWS services

- AWS Private CA – AWS Private Certificate Authority (AWS Private CA) is a hosted private CA service for issuing and revoking private digital certificates. It helps you create private CA hierarchies, including root and subordinate CAs, without the investment and maintenance costs of operating an on-premises CA.
- AWS RAM – AWS Resource Access Manager (AWS RAM) helps you securely share your resources across AWS accounts and within your organization or OUs in AWS Organizations. To reduce operational overhead in a multi-account environment, you can create a resource and use AWS RAM to share that resource across accounts.
- AWS Organizations – AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web. This pattern uses Amazon S3 to store the certificate revocation list (CRL) and access logs.
- AWS CloudFormation – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually. This pattern uses AWS CloudFormation to automatically deploy a multi-level CA hierarchy.

Code

The source code for this pattern is available on GitHub, in the AWS Private CA hierarchy repository. The repository includes:

- The AWS CloudFormation template `ACMPCA-RootCASubCA.yaml`. You can use this template to deploy the CA hierarchy for this implementation.
- Test files for use cases such as requesting, exporting, describing, and deleting a certificate.

To use these files, follow the instructions in the *Epics* section.

Epics

Architect the CA hierarchy

Task	Description	Skills required
Collect certificate subject information.	Gather certificate subject information about the certificate owner: organizat ion name, organization unit, country, state, locality, and common name.	Cloud architect, Security architect, PKI engineer
Collect optional information about AWS Organizations.	If the CA will be part of an AWS Organizations structure and you want to share the CA hierarchy inside that structure , collect the managemen t account number, the organization ID, and optionall y the OU ID (if you want to share the CA hierarchy only with a specific OU). Also, determine the AWS Organizat ions accounts or OUs, if any, that you want to share the CA with.	Cloud architect, Security architect, PKI engineer
Design the CA hierarchy.	Determine which account will house the root and subordina te CAs. Determine how many subordinate levels the hierarchy requires between the root and the end-entit y certificates. For more information, see Designing	Cloud architect, Security architect, PKI engineer

Task	Description	Skills required
	a CA hierarchy in the AWS Private CA documentation.	
Determine naming and tagging conventions for the CA hierarchy.	Determine the names for the AWS resources: the root CA and each subordinate CA. Determine which tags should be assigned to each CA.	Cloud architect, Security architect, PKI engineer
Determine required encryption and signing algorithms.	Determine the following: • Your organization's encryption algorithm requirements for the public keys that your CA uses when it issues a certificate. The default is RSA_2048. • The key algorithm that your CA uses for certificate signing. The default is SHA256WITHRSA.	Cloud architect, Security architect, PKI engineer
Determine certificate revocation requirements for the CA hierarchy.	If certificate revocation capabilities are required, establish a naming convention for the S3 bucket that contains the certificate revocation list (CRL).	Cloud architect, Security architect, PKI engineer
Determine the logging requirements for the CA hierarchy.	If access logging capabilities are required, establish a naming convention for the S3 bucket that contains the access logs.	Cloud architect, Security architect, PKI engineer

Task	Description	Skills required
Determine certificate expiration periods.	Determine the expiration date for the root certificate (the default is 10 years), end-entity certificates (the default is 13 months), and subordinate CA certificates (the default is 3 years). Subordinate CA certificates should expire earlier than the CA certificates at higher levels in the hierarchy. For more information, see Managing the private CA lifecycle in the AWS Private CA documentation.	Cloud architect, Security architect, PKI engineer

Deploy the CA hierarchy

Task	Description	Skills required
Complete prerequisites.	Complete the steps in the Prerequisites section of this pattern.	Cloud administrator, Security engineers, PKI engineers
Create CA roles for various personas.	1. Determine the types of AWS Identity and Access Management (IAM) roles or users in AWS IAM Identity Center (successor to AWS Single Sign-On) required to administer the various levels of the CA hierarchy , such as RootCAAdmin,	Cloud administrator, Security engineers, PKI engineers

Task	Description	Skills required
	SubordinateCAAdmin, and CertificateConsumer. 2. Determine the granularity of policies needed to separate duties. 3. Create the required IAM roles or users in IAM Identity Center in the account that the CA hierarchy resides in.	
Deploy the CloudFormation stack.	1. From the GitHub repository for this pattern, download the AWSPCA-RootCASubCA .yaml template. 2. Deploy the template from the AWS CloudFormation console or from the AWS Command Line Interface (AWS CLI). For more information, see Working with stacks in the CloudFormation documentation. 3. Complete the parameters in the template, including the organization name, the OU name, the key algorithm, the signing algorithm, and other options.	Cloud administrator, Security engineers, PKI engineers

Task	Description	Skills required
Architect a solution for updating certificates used by user-managed resources.	Resources of integrated AWS services, such as Elastic Load Balancing, update certifica tes automatically before expiration. However, user-managed resources, such as web servers that are running on Amazon Elastic Compute Cloud (Amazon EC2) instances , require another mechanism. 1. Determine which user-managed resources require end-entity certificates from the private CA. 2. Plan a process to be notified about the expiratio n of user-managed resources and certifica tes. For examples, see the following: • Using an AWS Config managed rule • Using Amazon CloudWatch and Amazon EventBridge 3. Write custom scripts to update certificates on user-managed resources and integrate them with AWS services to automate the updates. For more information about integrated AWS Services,	Cloud administrator, Security engineers, PKI engineers

Task	Description	Skills required
	see [Services integrated with AWS Certificate Manager](#) in the ACM documentation.	

Validate and document the CA hierarchy

Task	Description	Skills required
Validate optional AWS RAM sharing.	If the CA hierarchy is shared with other accounts in AWS Organizations, log in to one of those accounts from the AWS Management Console, navigate to the [AWS Private CA console](#), and confirm that the newly created CA is shared to this account. Only the lowest-level CA in the hierarchy will be visible, because that is the CA that generates the end-entity certificates. Repeat for a sampling of the accounts that the CA is shared with.	Cloud administrator, Security engineers, PKI engineers
Validate the CA hierarchy with certificate lifecycle tests.	In the [GitHub repository](#) for this pattern, locate the lifecycle tests. Run the tests from the AWS CLI to request a certificate, export a certificate, describe a certificate, and delete a certificate.	Cloud administrator, Security engineers, PKI engineers

Task	Description	Skills required
Import the certificate chain into trust stores.	For browsers and other applications to trust a certificate, the certificate's issuer must be included in the browser's trust store, which is a list of trusted CAs. Add the certificate chain for the new CA hierarchy to your browser's and application's trust store. Confirm that the end-entity certificates are trusted.	Cloud administrator, Security engineers, PKI engineers
Create a runbook to document the CA hierarchy.	Create a runbook document to describe the architect ure of the CA hierarchy, the account structure that can request end-entity certifica tes, the build process, and basic management tasks such as issuing end-entity certifica tes (unless you want to allow self-service by child accounts), usage, and tracking.	Cloud administrator, Security engineers, PKI engineers

Related resources

- Designing a CA hierarchy (AWS Private CA documentation)
- Creating a private CA (AWS Private CA documentation)
- How to use AWS RAM to share your AWS Private CA cross-account (AWS blog post)
- AWS Private CA best practices (AWS blog post)
- Enable resource sharing within AWS Organizations (AWS RAM documentation)
- Managing the private CA lifecycle (AWS Private CA documentation)

- [acm-certificate-expiration-check for AWS Config](#) (AWS Config documentation)
- [AWS Certificate Manager now provides certificate expiry monitoring through Amazon CloudWatch](#) (AWS announcement)
- [Services integrated with AWS Certificate Manager](#) (ACM documentation)

Additional information

When you export certificates, use a passphrase that is cryptographically strong and aligns with your organization's data loss prevention strategy.

Streamline Amazon EC2 compliance management with Amazon Bedrock agents and AWS Config

Anand Bukkapatnam Tirumala, Amazon Web Services

Summary

This pattern describes how to integrate Amazon Bedrock with AWS Config rules to facilitate compliance management for Amazon Elastic Compute Cloud (Amazon EC2) instances. The approach uses advanced generative AI capabilities to provide tailored recommendations that are aligned with the AWS Well-Architected Framework, to ensure optimal instance type selection and system efficiency. Key features of this pattern include:

- Automated compliance monitoring: AWS Config rules continuously assess EC2 instances against predefined criteria for desired instance types.
- AI-driven recommendations: The generative AI models in Amazon Bedrock analyze infrastructure patterns. These models provide intelligent suggestions for improvements based on best practices that are outlined in the AWS Well-Architected Framework.
- Remediation: Amazon Bedrock action groups enable automated remediation steps to swiftly address non-compliant instances and minimize potential performance or cost inefficiencies.
- Scalability and adaptability: The solution is designed to scale with your infrastructure and adapt to your evolving cloud architecture needs.
- Enhanced security recommendations: Compliance with AWS Well-Architected principles contributes to improved security posture and system performance.

You can use this pattern as a blueprint to deploy your own generative AI-based infrastructure into multiple environments with minimal changes, using DevOps practices as necessary.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An AWS Identity and Access Management (IAM) role with permissions to create and manage resources in Amazon Simple Storage Service (Amazon S3) buckets, AWS Config, AWS Lambda functions, Amazon Bedrock, IAM, Amazon CloudWatch Logs, and Amazon EC2.

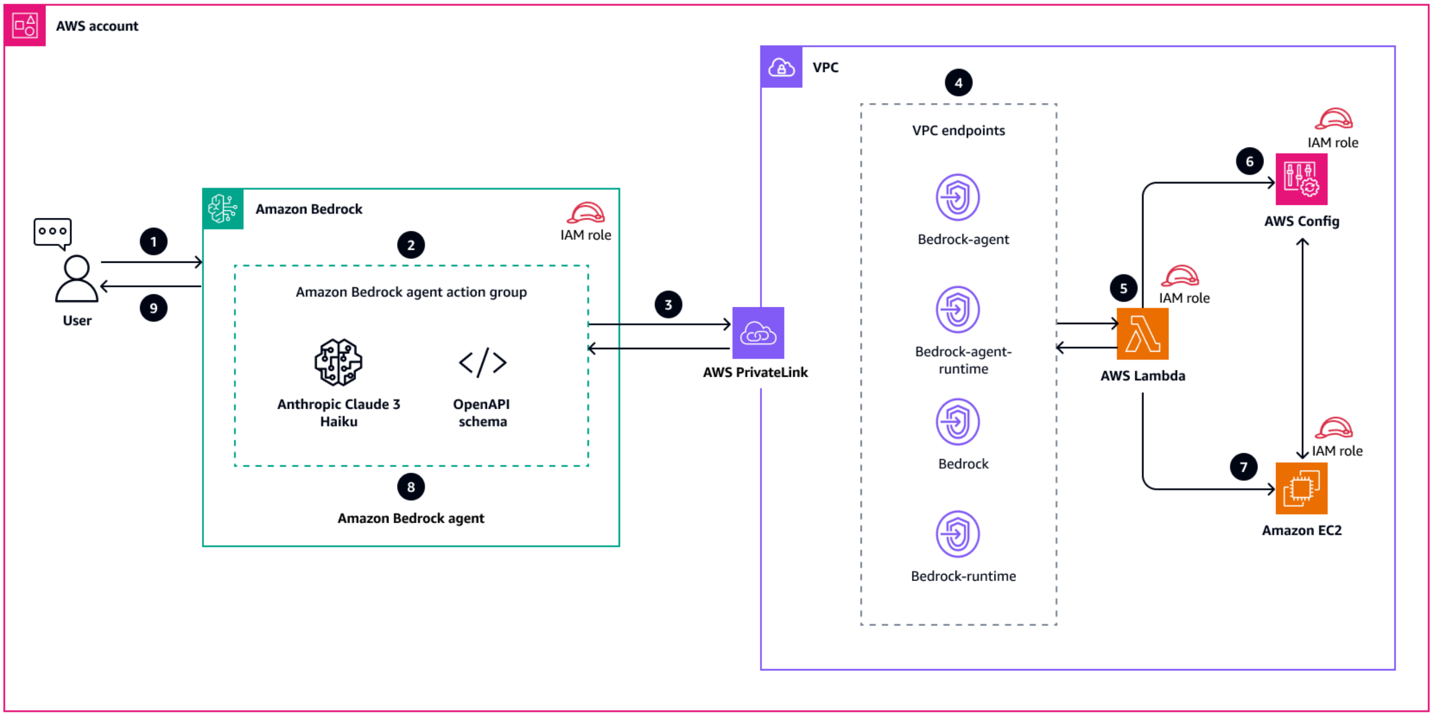
- An EC2 instance to flag as non-compliant. Do not use the `t2.small` type for this instance.
- Amazon Titan Text Embeddings V2 and Anthropic Claude 3 Haiku models enabled in your AWS account. To enable model access for the AWS Region where you are deploying the solution, see Add or remove access to Amazon Bedrock foundation models in the Amazon Bedrock documentation.
- Terraform, installed and configured.
- The AWS Command Line Interface (AWS CLI) v2 installed and configured in the deployment environment.
- Completed review of the Amazon Responsible AI policy.

Limitations

- Some AWS services aren't available in all AWS Regions. For Region availability, see AWS services by Region. For specific endpoints, see Service endpoints and quotas, and choose the link for the service.
- This solution has been tested by using the Amazon Titan Text Embeddings V2 and Claude 3 Haiku models. If you prefer to use other models, you can customize the Terraform code, which is parameterized for easy changes.
- This solution does not include a chat history feature, and the chat isn't stored.

Architecture

The following diagram shows the workflow and architecture components for this pattern.



The workflow consists of these steps:

1. The user interacts with the model through the Amazon Bedrock chat console. The user asks questions such as:
   - `What can you help me with?`
   - `List non-complaint resources`
   - `Suggest security best practices`
2. If the model is pre-trained, it responds to the prompts directly from its existing knowledge. Otherwise, the prompt goes to the Amazon Bedrock action group.
3. The action group reaches the VPC endpoints by using AWS PrivateLink for secure service communication.
4. The request reaches the Lambda function through the VPC endpoints for Amazon Bedrock services.
5. The Lambda function is the primary execution engine. Based on the request, the function calls the API to perform actions on the AWS services. It also handles operation routing and execution.
6. The Lambda function calls AWS Config to determine non-complaint resources (the non-compliant EC2 instance that you created as a prerequisite).
7. AWS Config flags the non-complaint resource. This pattern deploys the AWS Config desired-instance-type rule to find the ideal EC2 instance size.

8. AWS Config prompts the user to pause or remediate the instance, and takes action accordingly on the EC2 instance. Amazon Bedrock understands this return payload.
9. The user receives a response on the Amazon Bedrock chat console.

Automation and scale

This solution uses Terraform as an infrastructure as code (IaC) tool to enable easy deployment to AWS accounts and to function as a standalone utility across multiple accounts. This approach simplifies management and improves consistency in deployments.

Tools

AWS services

- AWS Config enables you to assess, audit, and evaluate the configurations of your AWS resources for compliance and desired settings.
- Amazon Bedrock is a fully managed AI service that provides access to many high-performing foundation models through a unified API.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.

Other tools

- Git is an open source, distributed version control system.
- Terraform is an infrastructure as code (IaC) tool from HashiCorp that helps you create and manage cloud and on-premises resources.

Code repository

The code for this pattern is available in the GitHub sample-awsconfig-bedrock-compliance-manager repository.

Best practices

- Follow the principle of least privilege and grant the minimum permissions required to perform a task. For more information, see Grant least privilege and Security best practices and use cases in the IAM documentation.
- Monitor Lambda execution logs regularly. For more information, see Monitoring, debugging, and troubleshooting Lambda functions and Best practices for working with AWS Lambda functions in the Lambda documentation.

Epics

Deploy the solution

Task	Description	Skills required
Clone the repository.	To clone the repository for this pattern, use the following command: `git clone "git@github.com:aws-samples/sample-awsconfig-bedrock-compliance-manager.git"`	AWS DevOps, Build lead, DevOps engineer, Cloud administrator
Edit the environment variables.	In the root directory of the cloned repository on your local machine, edit the `terraform.tfvars` file. Review the placeholders that are marked with `[XXXXX]`, and edit them based on your environment.	AWS systems administrator, AWS DevOps, DevOps engineer, AWS administrator
Create the infrastructure.	1. To create the infrastructure for this solution, run the following commands:	AWS DevOps, DevOps engineer, AWS systems

Task	Description	Skills required
	``` terraform init terraform plan ``` 2. Review the execution plan, and then run the command: ``` terraform apply -- auto-approve ```	administrator, Cloud administrator

Use the agent

Task	Description	Skills required
Chat with the agent.	Deploying the solution in the previous step deploys `security-bot-agent` , which is an Amazon Bedrock agent with a chat console. To use the agent: 1. Sign in to the AWS Management Console with an IAM identity that has permissions to use the Amazon Bedrock console. Then, open the [Amazon Bedrock console](https://console.aws.amazon.com/bedrock/). 2. In the navigation pane, choose **Build**, **Agents**. Select **security-bot-agent**. 3. Start conversing with the agent. For example:	AWS DevOps, DevOps engineer, AWS systems administrator, Cloud administrator

Task	Description	Skills required
	`List non-complaint EC2 instances` and: `Replace these non-complaint EC2 instances with compliant instances`	

Clean up resources

Task	Description	Skills required
Delete the infrastructure and resources.	When you've completed your work with this solution, you can delete the infrastructure created by this pattern by running the command: `terraform destroy --auto-approve`	AWS DevOps, DevOps engineer, AWS systems administrator, Cloud administrator

Troubleshooting

Issue	Solution
Agent behavior issues	For troubleshooting information, see Test and troubleshoot agent behavior in the Amazon Bedrock documentation.

Issue	Solution
AWS Lambda network issues	For more information, see Troubleshoot networking issues in Lambda in the Lambda documentation.
IAM permissions	For more information, see Troubleshoot IAM in the IAM documentation.

Related resources

- Amazon Bedrock agents
- Use action groups to define actions for your agent to perform (Amazon Bedrock documentation)
- desired-instance-type rule (AWS Config documentation)
- How AWS Config works (AWS Config documentation)

Turn off security standard controls across all Security Hub member accounts in a multi-account environment

Michael Fuellbier and Ahmed Bakry, Amazon Web Services

Summary

> **Important**
>
> AWS Security Hub now supports central configuration for security standards and controls, across accounts. This new feature addresses many of the scenarios that are covered by the solution in this AWS Prescriptive Guidance pattern. Before you deploy the solution in this pattern, see Central configuration in Security Hub.

In the Amazon Web Services (AWS) Cloud, AWS Security Hub standard controls, such as CIS AWS Foundations Benchmark or AWS Foundational Security Best Practices, can only be turned off (disabled) manually from within a single AWS account. In a multi-account environment, you can't turn off the controls across multiple Security Hub member accounts with "one click" (that is, one API call). This pattern demonstrates how to use one click to turn off the Security Hub standard controls across all the Security Hub member accounts managed by your Security Hub administrator account.

Prerequisites and limitations

Prerequisites

- A multi-account environment consisting of a Security Hub administrator account that manages multiple member accounts
- AWS Command Line Interface (AWS CLI) version 2, installed
- AWS Serverless Application Model Command Line Interface (AWS SAM CLI), installed

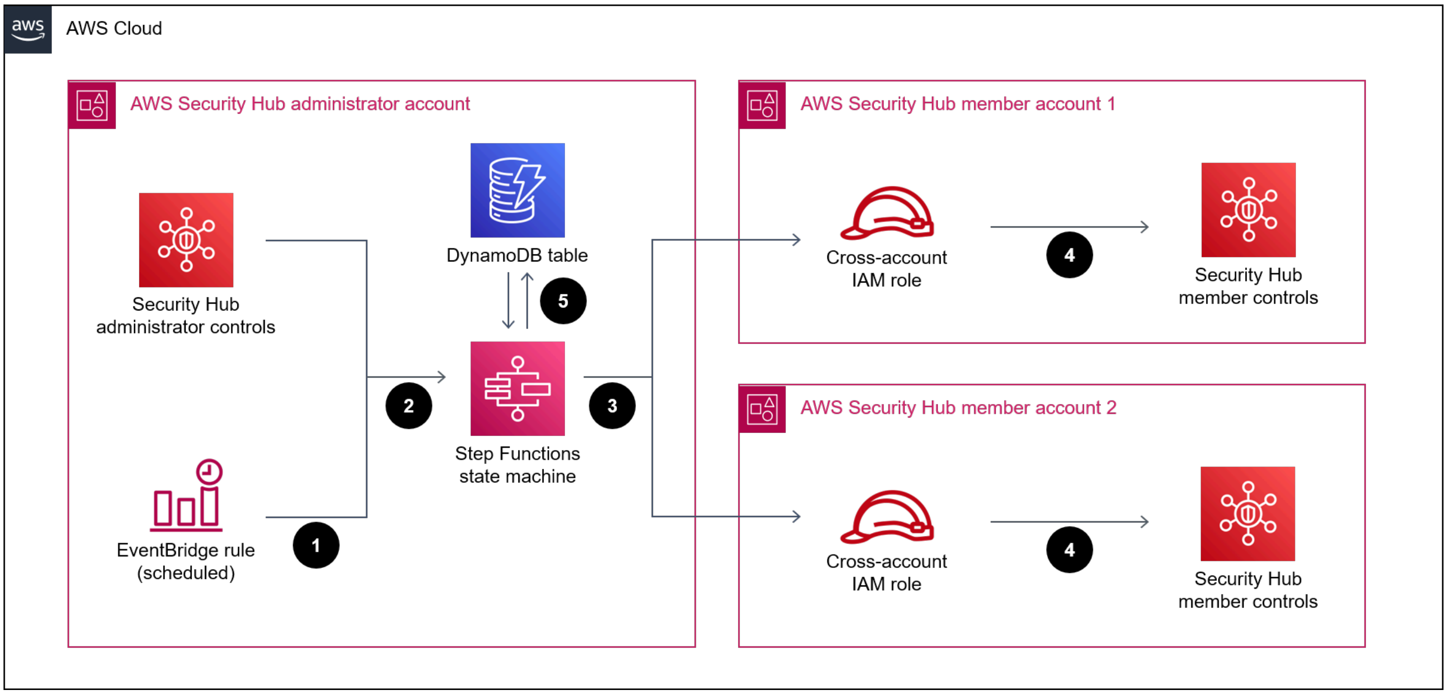
Limitations

- This pattern works only in a multi-account environment where a single Security Hub administrator account manages multiple member accounts.

- The event initiation causes multiple parallel invocations if you change a lot of controls in a very short time frame. This can lead to API throttling and cause the invocations to fail. For example, this scenario can happen if you programmatically change a lot of controls by using the Security Hub Controls CLI.

Architecture

The following diagram shows an example of a AWS Step Functions workflow that turns off Security Hub standard controls across multiple Security Hub member accounts (as viewed from the Security Hub administrator account).



The diagram includes the following workflow:

1. An Amazon EventBridge rule is initiated on a daily schedule and invokes the state machine. You can modify the timing of the rule by updating the `Schedule` parameter in your AWS CloudFormation template.
2. An EventBridge rule is initiated whenever a control is turned on or off in the Security Hub administrator account.
3. A Step Functions state machine propagates the status of the security standard controls (that is, controls that are turned on or off) from the Security Hub administrator account to the member accounts.

4. A cross-account AWS Identity and Access Management (IAM) role is deployed in each member account and assumed by the state machine. The state machine turns the controls on or off in each member account.
5. A Amazon DynamoDB table contains exceptions and information about which controls to turn on or off in a particular account. This information overrides the configurations fetched from the Security Hub administrator account for the specified member account.

Note

The purpose of the scheduled EventBridge rule is to ensure that newly added Security Hub member accounts have the same control status as existing accounts.

Tools

AWS services

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- Amazon EventBridge is a serverless event bus service that helps you connect your applications with real-time data from a variety of sources. For example, AWS Lambda functions, HTTP invocation endpoints using API destinations, or event buses in other AWS accounts.
- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- AWS Serverless Application Model (AWS SAM) is an open-source framework that helps you build serverless applications in the AWS Cloud.
- AWS Security Hub provides a comprehensive view of your security state in AWS. It also helps you check your AWS environment against security industry standards and best practices.
- AWS Step Functions is a serverless orchestration service that helps you combine AWS Lambda functions and other AWS services to build business-critical applications.

Code repository

The code for this pattern is available on the GitHub AWS Security Hub Cross-Account Controls Disabler repository. The code repository contains the following files and folders:

- **UpdateMembers/template.yaml** – This file contains components deployed in the Security Hub administrator account, including the Step Functions state machine and the EventBridge rules.
- **member-iam-role/template.yaml** – This file contains the code to deploy the cross-account IAM role in a member account.
- **stateMachine.json** – This file defines the state machine's workflow.
- **GetMembers/index.py** – This file contains the code for the **GetMembers** state machine. A script retrieves the status of the security standard controls in all existing Security Hub member accounts.
- **UpdateMember/index.py** – This file contains a script that updates the control status in each member account.
- **CheckResult/index.py** – This file contains a script that checks the status of the workflow invocation (accepted or failed).

Epics

Deploy a cross-account IAM role in the Security Hub member accounts

Task	Description	Skills required
Identify the account ID of the Security Hub administrator account.	Set up a Security Hub administrator account and then note the account ID of the administrator account.	Cloud architect
Deploy the CloudFormation template that includes the cross-account IAM role in the member accounts.	To deploy the **member-iam-role/template.yaml** template in all the member accounts managed by the Security Hub administrator account, run the following command: `aws cloudformation deploy \`	AWS DevOps

Task	Description	Skills required
	`--template-file member-iam-role/template.yaml \ --capabilities CAPABILITY_NAMED_IAM \ --stack-name <your-stack-name> \ --parameter-overrides SecurityHubAdminAccountId=<account-ID>` The `SecurityHubAdminAccountId` parameter must match the Security Hub administrator account ID that you noted earlier.	

Deploy a state machine in the Security Hub administrator account

Task	Description	Skills required
Package the CloudFormation template that includes the state machine with AWS SAM.	To package the **UpdateMembers/template.yaml** template in the Security Hub administrator account, run the following command: `sam package \ --template-file UpdateMembers/template.yaml \ --output-template-file UpdateMembers/template-out.yaml \ --s3-bucket <amzn-s3-demo-bucket>`	AWS DevOps

Task	Description	Skills required
	Note Your Amazon Simple Storage Service (Amazon S3) bucket must be in the same AWS Region where you deploy the CloudFormation template.	

<table>
<tr><th>Task</th><th>Description</th><th>Skills required</th></tr>
<tr><td>Deploy the packaged CloudFormation template in the Security Hub administr ator account.</td><td>To deploy the CloudForm ation template in the Security Hub administrator account, run the following command:

<pre><code>aws cloudformation
 deploy \
   --template-file
 UpdateMembers/temp
late-out.yaml \
   --capabilities
 CAPABILITY_IAM \
   --stack-name <stack-
name></code></pre>

In the member-iam-role/ template.yaml template, the <code>MemberIAMRolePath</code> parameter must match the <code>IAMRolePath</code> parameter and <code>MemberIAMRoleName</code> must match <code>IAMRoleName</code> .

Note

Because Security Hub is a regional service, you must deploy the template individually in each AWS Region. Be sure to first package the solution into an Amazon S3 bucket in each Region.</td><td>AWS DevOps</td></tr>
</table>

Related resources

- Designating a Security Hub administrator account (Security Hub documentation)
- Handling Errors, Retries, and adding Alerting to AWS Step Functions State Machine Executions (AWS blog post)

Update AWS CLI credentials from AWS IAM Identity Center by using PowerShell

Chad Miles and Andy Bowen, Amazon Web Services

Summary

If you want to use AWS IAM Identity Center (successor to AWS Single Sign-On) credentials with AWS Command Line Interface (AWS CLI), AWS SDKs, or AWS Cloud Development Kit (AWS CDK), you typically have to copy and paste the credentials from the IAM Identity Center console into the command line interface. This process can take a considerable amount of time and has to be repeated for each account that requires access.

One common solution is to use the AWS CLI `aws sso configure` command. This command adds an IAM Identity Center enabled profile to your AWS CLI or AWS SDK. However, the disadvantage of this solution is that you must run the command `aws sso login` for each AWS CLI profile or account that you have configured this way.

As an alternative solution, this pattern describes how to use AWS CLI named profiles and AWS Tools for PowerShell to store and refresh credentials for multiple accounts from a single IAM Identity Center instance simultaneously. The script also stores IAM Identity Center session data in memory for refreshing credentials without logging into IAM Identity Center again.

Prerequisites and limitations

Prerequisites

- PowerShell, installed and configured. For more information, see Installing PowerShell (Microsoft documentation).
- AWS Tools for PowerShell, installed and configured. For performance reasons, we highly recommend that you install the modularized version of AWS Tools for PowerShell, called `AWS.Tools`. Each AWS service is supported by its own individual, small module. In the PowerShell prompt, enter the following commands to install the modules needed for this pattern: `AWS.Tools.Installer`, `SSO`, and `SSOIDC`.

```
Install-Module AWS.Tools.Installer
Install-AWSToolsModule SSO, SSOIDC
```

For more information, see Install AWS.Tools on Windows or Install AWS.Tools on Linux or macOS.

- AWS CLI or the AWS SDK must be previously configured with working credentials by doing one of the following:
  - Use the AWS CLI `aws configure` command. For more information, see Quick configuration (AWS CLI documentation).
  - Configure AWS CLI or AWS CDK to get temporary access through an IAM role. For more information, see Getting IAM role credentials for CLI access (IAM Identity Center documentation).

Limitations

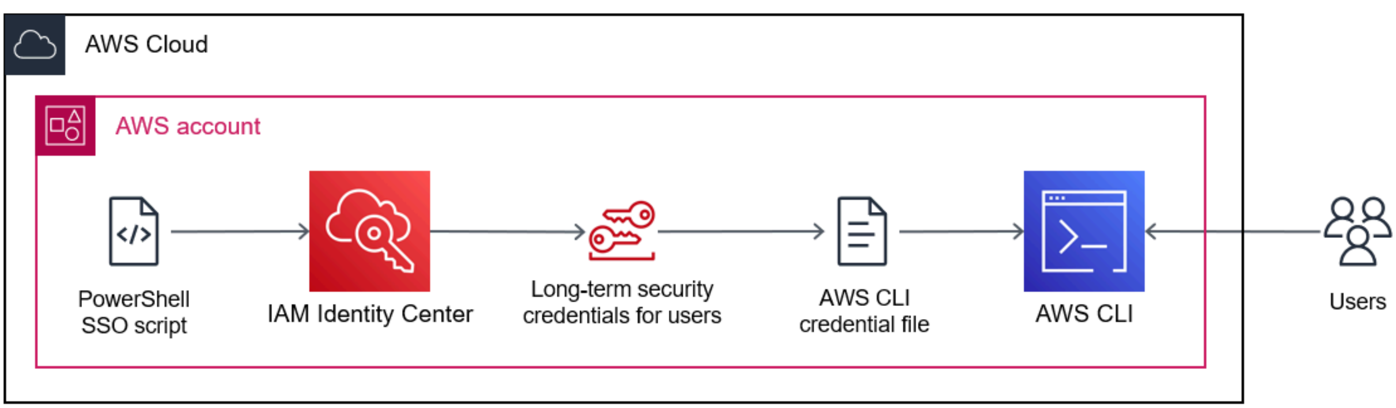
- This script can't be used in a pipeline or fully automated solution. When you deploy this script, you must manually authorize access from IAM Identity Center. The script then continues automatically.

Product versions

- For all operating systems, it is recommended that you use PowerShell version 7.0 or later.

Architecture

You can use the script in this pattern to simultaneously refresh multiple IAM Identity Center credentials, and you can create a credential file for use with AWS CLI, AWS SDKs, or AWS CDK.



Tools

AWS services

- AWS Command Line Interface (AWS CLI) is an open-source tool that helps you interact with AWS services through commands in your command-line shell.
- AWS IAM Identity Center helps you centrally manage single sign-on (SSO) access to all of your AWS accounts and cloud applications.
- AWS Tools for PowerShell are a set of PowerShell modules that help you script operations on your AWS resources from the PowerShell command line.

Other tools

- PowerShell is a Microsoft automation and configuration management program that runs on Windows, Linux, and macOS.

Best practices

Keep one copy of this script for each IAM Identity Center instance. Using one script for multiple instances is not supported.

Epics

Run the SSO script

Task	Description	Skills required
Customize the SSO script.	1. Copy the SSO script in the Additional information section. 2. In the `Param` section, for your AWS environment, define the values for the following variables: • `DefaultRoleName` – The IAM role or	Cloud administrator

Task	Description	Skills required
	permission set to use by default. • `Region` – The AWS Region in which IAM Identity Center is deployed. For a complete list of Regions and their codes, see [Regional endpoints](#). • `StartUrl` – The URL used to access your IAM Identity Center login page. Use the same format as the example value in the script. • `EnvironmentName` – A short name to reference this copy of the script, to be used when you're running multiple script copies in the same session. 3. Under line 10, which reads `# Add your Account Information`, edit the following values in the hash tables to reflect your environment: • `Profile` – The AWS CLI profile name in which to store the temporary credentials.	

Task	Description	Skills required
	`AccountId` – The ID of the AWS account for which you are retrieving credentials. `RoleName` – The name of the IAM Identity Center role or permission set you want to use. You can leave this as `$DefaultRoleName` if you want to use the same role you defined in the `Param` section. Each line in the hash table must end with a comma except the last one.	

Run the SSO script.	It is recommended that you run your custom script in the PowerShell shell with the following command.
	`./Set-AwsCliSsoCredentials.ps1`
	Alternatively, you can run the script from another shell by entering the following command.
	`pwsh Set-AwsCliSsoCredentials.ps1`
		Cloud administrator

Troubleshooting

Issue	Solution
`No Access` error	The IAM role that you are using doesn't have permissions to access the role or permission set that you defined in a `RoleName` parameter . Update the permissions for the role you are using, or define a different role or permission set in the script.

Related resources

- Where are configuration settings stored? (AWS CLI documentation)
- Configuring the AWS CLI to use AWS IAM Identity Center (AWS CLI documentation)
- Using named profiles (AWS CLI documentation)

Additional information

SSO script

In the following script, replace placeholders in angle brackets (<>) with your own information and remove the angle brackets.

```
Set-AwsCliSsoCredentials.ps1
Param(
    $DefaultRoleName = '<AWSAdministratorAccess>',
    $Region          = '<us-west-2>',
    $StartUrl        = "<https://d-12345abcde.awsapps.com/start/>",
    $EnvironmentName = "<CompanyName>"
)
Try {$SsoAwsAccounts = (Get-Variable -name "$($EnvironmentName)SsoAwsAccounts" -Scope
 Global -ErrorAction 'SilentlyContinue').Value.Clone()}
Catch {$SsoAwsAccounts = $False}
if (-not $SsoAwsAccounts) { $SsoAwsAccounts = @(
# Add your account information in the list of hash tables below, expand as necessary,
 and do not forget the commas
```

```
    @{Profile = "<Account1>"      ; AccountId = "<012345678901 >"; RoleName =
 $DefaultRoleName },
    @{Profile = "<Account2>"      ; AccountId = "<123456789012>"; RoleName =
 "<AWSReadOnlyAccess>" }
)}
$ErrorActionPreference = "Stop"
if (-not (Test-Path ~\.aws))      { New-Item ~\.aws -type Directory }
if (-not (Test-Path ~\.aws\credentials)) { New-Item ~\.aws\credentials -type File }
$CredentialFile = Resolve-Path ~\.aws\credentials
$PsuedoCreds    = @{AccessKey =
 'AKAEXAMPLE123ACCESS';SecretKey='PsuedoS3cret4cceSSKey123PsuedoS3cretKey'} # Pseudo
 Creds, do not edit.
Try {$SSOTokenExpire = (Get-Variable -Scope Global -Name
 "$($EnvironmentName)SSOTokenExpire" -ErrorAction 'SilentlyContinue').Value} Catch
 {$SSOTokenExpire = $False}
Try {$SSOToken       = (Get-Variable -Scope Global -Name "$($EnvironmentName)SSOToken"
 -ErrorAction 'SilentlyContinue').Value }      Catch {$SSOToken       = $False}
if ( $SSOTokenExpire -lt (Get-Date) ) {
    $SSOToken = $Null
    $Client   = Register-SSOOIDCClient -ClientName cli-sso-client -ClientType public -
Region $Region @PsuedoCreds
    $Device   = $Client | Start-SSOOIDCDeviceAuthorization -StartUrl $StartUrl -Region
 $Region @PsuedoCreds
    Write-Host "A Browser window should open. Please login there and click ALLOW." -
NoNewline
    Start-Process $Device.VerificationUriComplete
    While (-Not $SSOToken){
        Try {$SSOToken = $Client | New-SSOOIDCToken -DeviceCode $Device.DeviceCode -
GrantType "urn:ietf:params:oauth:grant-type:device_code" -Region $Region @PsuedoCreds}
        Catch {If ($_.Exception.Message -notlike "*AuthorizationPendingException*")
{Write-Error $_.Exception} ; Start-Sleep 1}
    }
    $SSOTokenExpire = (Get-Date).AddSeconds($SSOToken.ExpiresIn)
    Set-Variable -Name "$($EnvironmentName)SSOToken" -Value $SSOToken -Scope Global
    Set-Variable -Name "$($EnvironmentName)SSOTokenExpire" -Value $SSOTokenExpire -
Scope Global
}
$CredsTime     = $SSOTokenExpire - (Get-Date)
$CredsTimeText = ('{0:D2}:{1:D2}:{2:D2} left on SSO Token' -f $CredsTime.Hours,
 $CredsTime.Minutes, $CredsTime.Seconds).TrimStart("0 :")
for ($i = 0; $i -lt $SsoAwsAccounts.Count; $i++) {
    if (([DateTimeOffset]::FromUnixTimeSeconds($SsoAwsAccounts[$i].CredsExpiration /
 1000)).DateTime -lt (Get-Date).ToUniversalTime()) {
```

```
        Write-host "`r
    `rRegistering Profile $($SsoAwsAccounts[$i].Profile)" -NoNewline
        $TempCreds = $SSOToken | Get-SSORoleCredential -AccountId
 $SsoAwsAccounts[$i].AccountId -RoleName $SsoAwsAccounts[$i].RoleName -Region $Region
 @PsuedoCreds
        [PSCustomObject]@{AccessKey = $TempCreds.AccessKeyId; SecretKey =
 $TempCreds.SecretAccessKey; SessionToken = $TempCreds.SessionToken
        } | Set-AWSCredential -StoreAs $SsoAwsAccounts[$i].Profile -ProfileLocation
 $CredentialFile
        $SsoAwsAccounts[$i].CredsExpiration = $TempCreds.Expiration
    }
}
Set-Variable -name "$($EnvironmentName)SsoAwsAccounts" -Value $SsoAwsAccounts.Clone() -
Scope Global
Write-Host "`r$($SsoAwsAccounts.Profile) Profiles registered, $CredsTimeText"
```

Use AWS Config to monitor Amazon Redshift security configurations

Lucas Kauffman and abhishek sengar, Amazon Web Services

Summary

Using AWS Config, you can evaluate the security configurations for your AWS resources. AWS Config can monitor the resources, and if configuration settings violate your defined rules, AWS Config flags the resource as *noncompliant*.

You can use AWS Config to evaluate and monitor your Amazon Redshift clusters and databases. For more information about security recommendations and features, see Security in Amazon Redshift. This pattern includes custom AWS Lambda rules for AWS Config. You can deploy these rules in your account to monitor the security configurations of your Amazon Redshift clusters and databases. The rules in this pattern help you use AWS Config to confirm that:

- Audit logging is enabled for the databases in the Amazon Redshift cluster
- SSL is required to connect to the Amazon Redshift cluster
- Federal Information Processing Standards (FIPS) ciphers are in use
- Databases in the Amazon Redshift cluster are encrypted
- User activity monitoring is enabled

Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Config must be enabled in your AWS account. For more information, see Setting Up AWS Config with the Console or Setting Up AWS Config with the AWS CLI.
- Python version 3.9 or later must be used for the AWS Lambda handler. For more information, see Working with Python (AWS Lambda documentation).

Product versions

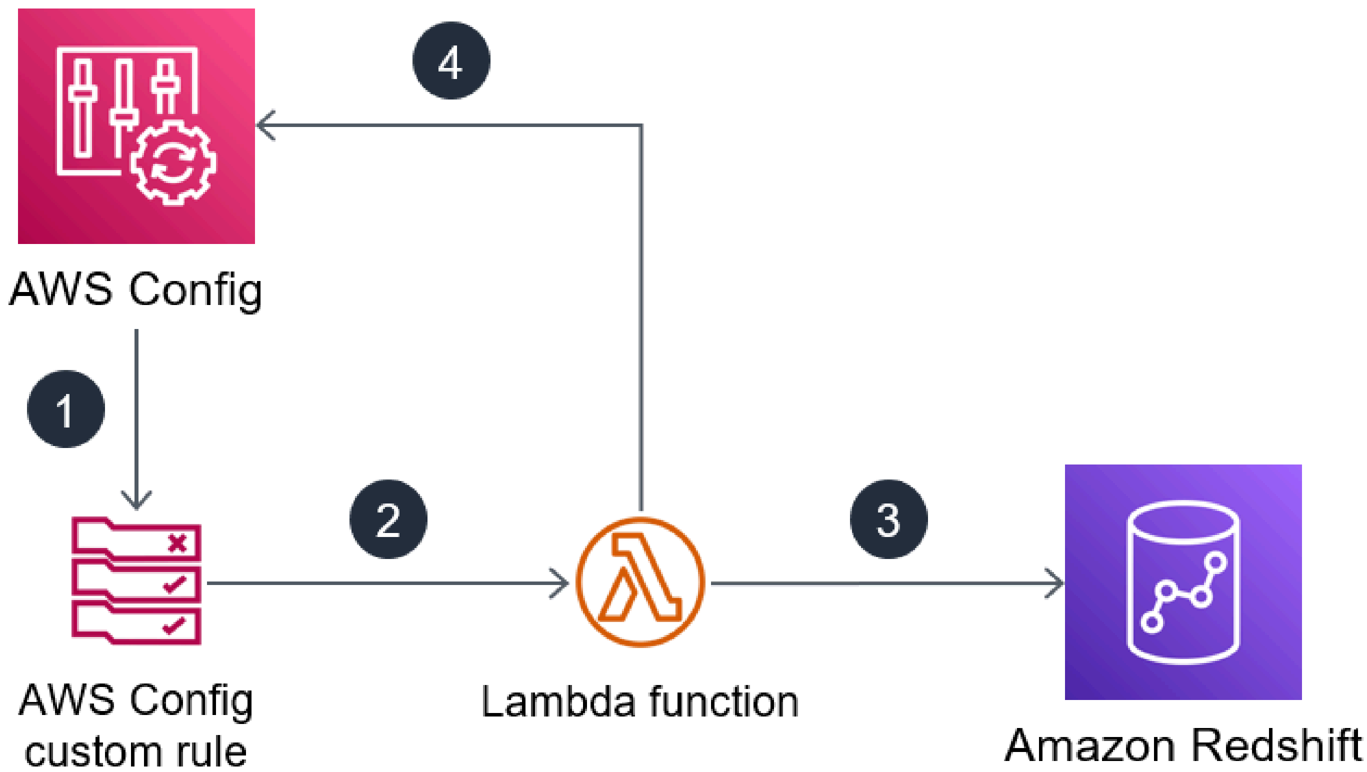
- Python version 3.9 or later

Architecture

Target technology stack

- AWS Config

Target architecture



1. AWS Config periodically runs the custom rule.
2. The custom rule invokes the Lambda function.
3. The Lambda function checks the Amazon Redshift clusters for non-compliant configurations.
4. The Lambda function reports the compliance state of each Amazon Redshift cluster to AWS Config.

Automation and scale

The AWS Config custom rules scale to assess all Amazon Redshift clusters in your account. No additional action is required to scale this solution.

Tools

AWS services

- [AWS Config](#) provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time.
- [AWS Identity and Access Management (IAM)](#) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- [AWS Lambda](#) is a compute service that helps you run code without needing to provision or manage servers. It runs your code only when needed and scales automatically, so you pay only for the compute time that you use.
- [Amazon Redshift](#) is a managed petabyte-scale data warehouse service in the AWS Cloud.

Code repository

The code for this pattern is available in the GitHub [aws-config-rules](#) repository. The custom rules in this repository are Lambda rules in the Python programming language. This repository contains many custom rules for AWS Config. Only the following rules are used in this pattern:

- `REDSHIFT_AUDIT_ENABLED` – Confirm that audit logging is enabled on the Amazon Redshift cluster. If you also want to confirm that user activity monitoring is enabled, deploy the `REDSHIFT_USER_ACTIVITY_MONITORING_ENABLED` rule instead.
- `REDSHIFT_SSL_REQUIRED` – Confirm that SSL is required to connect to the Amazon Redshift cluster. If you also want to confirm that Federal Information Processing Standards (FIPS) ciphers are in use, deploy the `REDSHIFT_FIPS_REQUIRED` rule instead.
- `REDSHIFT_FIPS_REQUIRED` – Confirm that SSL is required and FIPS ciphers are in use.
- `REDSHIFT_DB_ENCRYPTED` – Confirm that the databases in the Amazon Redshift cluster are encrypted.
- `REDSHIFT_USER_ACTIVITY_MONITORING_ENABLED` – Confirm that audit logging and user activity monitoring is enabled.

Epics

Prepare to deploy the rules

Task	Description	Skills required

Configure IAM policies.	1. Create a custom IAM identity-based policy that allows the Lambda execution role to read the Amazon Redshift cluster configurations. For more information, see [Managing access to resources](#) (Amazon Redshift documentation) and [Creating IAM policies](#) (IAM documentation).
	
	`{`
	`    "Version":`
	` "2012-10-17",`
	`    "Statement": [`
	`      {`
	`        "Effect":`
	` "Allow",`
	`        "Action": [`
	`            "redshift`
	`:DescribeClusterPa`
	`rameterGroups",`
	`            "redshift`
	`:DescribeClusterPa`
	`rameters",`
	`            "redshift`
	`:DescribeClusters",`
	`            "redshift`
	`:DescribeClusterSe`
	`curityGroups",`
		AWS administrator

Task	Description	Skills required

	``
	                "redshift
	:DescribeClusterSn
	apshots",
	                "redshift
	:DescribeClusterSu
	bnetGroups",
	                "redshift
	:DescribeEventSubs
	criptions",
	                "redshift
	:DescribeLoggingSt
	atus"
	            ],
	            "Resource":
	 "*"
	        }
	    ]
	}
	``
	2. Assign the AWSLambda Execute and AWSConfig RulesExecutionRole managed policies as a permissions policy for the Lambda execution role. For instructions, see Adding IAM identity permissions (IAM documentation).	

Clone the repository.	In a Bash shell, run the following command. This clones the aws-config-rules repository from GitHub. ``` git clone https://g ithub.com/awslabs/ aws-config-rules.git ```	General AWS

Deploy the rules in AWS Config

Task	Description	Skills required
Deploy the rules in AWS Config.	Following the instructions in Creating custom Lambda rules (AWS Config documentation), deploy one or more of the following rules in your account: • `REDSHIFT_AUDIT_ENABLED` • `REDSHIFT_SSL_REQUIRED` • `REDSHIFT_FIPS_REQUIRED` • `REDSHIFT_DB_ENCRYPTED` • `REDSHIFT_USER_ACTIVITY_MONITORING_ENABLED`	AWS administrator
Verify the rules are functional.	After deploying the rules, follow the instructions in Evaluating your resources (AWS Config documentation) to confirm that AWS Config is correctly evaluating your Amazon Redshift resources.	General AWS

Related resources

AWS service documentation

- Security in Amazon Redshift (Amazon Redshift documentation)

- Managing database security (Amazon Redshift documentation)
- AWS Config custom rules (AWS Config documentation)

AWS Prescriptive Guidance

- Verify that new Amazon Redshift clusters have required SSL endpoints
- Ensure an Amazon Redshift cluster is encrypted upon creation

Additional information

You can use the following AWS Managed Rules in AWS Config to confirm the following security configurations for Amazon Redshift:

- redshift-cluster-configuration-check – Use this rule to confirm that audit logging is enabled for the databases in the Amazon Redshift cluster and confirm that the databases are encrypted.
- redshift-require-tls-ssl – Use this rule to confirm that SSL is required to connect to the Amazon Redshift cluster.

Use Network Firewall to capture the DNS domain names from the Server Name Indication for outbound traffic

Kirankumar Chandrashekar, Amazon Web Services

Summary

This pattern shows you how to use AWS Network Firewall to collect the DNS domain names that are provided by the Server Name Indication (SNI) in the HTTPS header of your outbound network traffic. Network Firewall is a managed service that makes it easy to deploy critical network protections for Amazon Virtual Private Cloud (Amazon VPC), including the ability to secure outbound traffic with a firewall that blocks packets that fail to meet certain security requirements. Securing outbound traffic to specific DNS domain names is called egress filtering, which is the practice of monitoring and potentially restricting the flow of outbound information from one network to another.

After you capture the SNI data that passes through Network Firewall, you can use Amazon CloudWatch Logs and AWS Lambda to publish the data to an Amazon Simple Notification Service (Amazon SNS) topic that generates email notifications. The email notifications include the server name and other relevant SNI information. Additionally, you can use the output of this pattern to allow or restrict outbound traffic by domain name in the SNI by using firewall rules. For more information, see Working with stateful rule groups in AWS Network Firewall in the Network Firewall documentation.

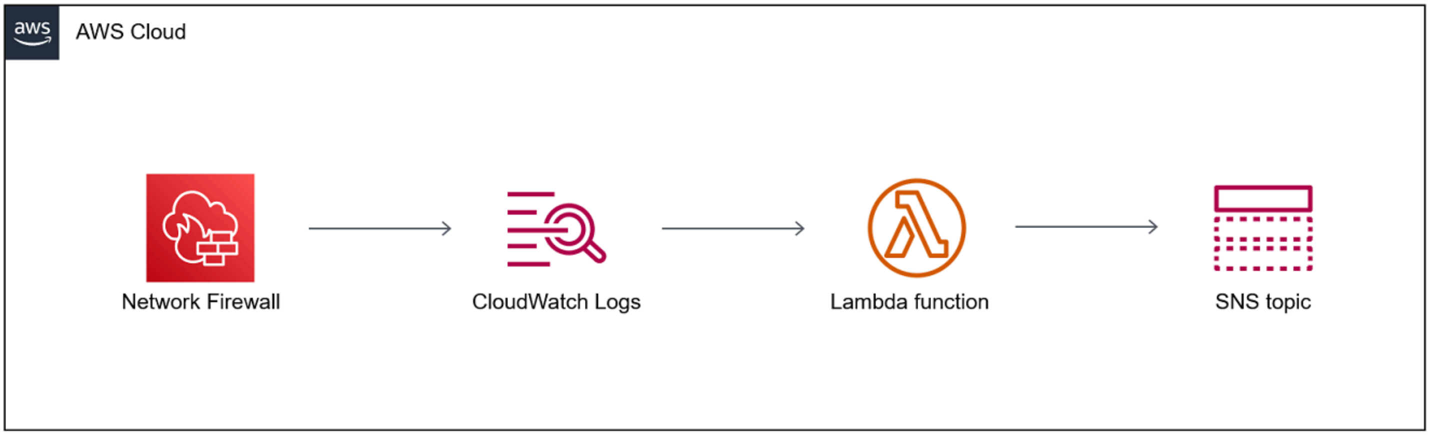
Prerequisites and limitations

Prerequisites

- An active AWS account.
- AWS Command Line Interface (AWS CLI) version 2, installed and configured on Linux, macOS, or Windows.
- Network Firewall, set up and configured in Amazon VPC and in use for inspecting outbound traffic. You can configure Network Firewall to use any of the following VPC configurations:
  - Simple single zone architecture with an internet gateway
  - Multi zone architecture with an internet gateway
  - Architecture with an internet gateway and a NAT gateway

Architecture

The following diagram shows how to use Network Firewall to collect SNI data from outbound network traffic, and then publish that data to an SNS topic by using CloudWatch Logs and Lambda.



The diagram shows the following workflow:

1. Network Firewall collects domain names from the SNI data in the HTTPS header of your outbound network traffic.
2. CloudWatch Logs monitors the SNI data and invokes a Lambda function whenever the outbound network traffic passes through Network Firewall.
3. The Lambda function reads the SNI data captured by CloudWatch Logs and then publishes that data to an SNS topic.
4. The SNS topic sends you an email notification that includes the SNI data.

Automation and scale

- You can use AWS CloudFormation to create this pattern by using infrastructure as code.

Technology stack

- Amazon CloudWatch Logs
- Amazon SNS
- Amazon VPC
- AWS Lambda
- AWS Network Firewall

Tools

AWS services

- Amazon CloudWatch Logs – You can use Amazon CloudWatch Logs to monitor, store, and access your log files from Amazon Elastic Compute Cloud (Amazon EC2) instances, AWS CloudTrail, Amazon Route 53, and other sources.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) is a managed service that provides message delivery from publishers to subscribers (also known as producers and consumers).
- Amazon VPC – Amazon Virtual Private Cloud (Amazon VPC) provisions a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.
- AWS Lambda – AWS Lambda is a compute service that lets you run code without provisioning or managing servers.
- AWS Network Firewall – AWS Network Firewall is a managed service that makes it easy to deploy essential network protections for all of your Amazon VPCs.

Epics

Create a CloudWatch log group for Network Firewall

Task	Description	Skills required
Create a CloudWatch log group.	1. Sign in to the AWS Management Console and open the CloudWatch console. 2. In the navigation pane, choose **Log groups**. 3. Choose **Actions**, and then choose **Create log group**.	Cloud administrator

Task	Description	Skills required
	4. Enter a name for the log group, and then choose **Create log group**. For more information, see Working with log groups and log streams in the CloudWatc h documentation.	

Create an SNS topic and subscription

Task	Description	Skills required
Create an SNS topic.	To create an SNS topic, follow the instructions in the Amazon SNS documentation.	Cloud administrator
Subscribe an endpoint to the SNS topic.	To subscribe an email address as an endpoint to the SNS topic that you created, follow the instructions in the Amazon SNS documentation. For **Protocol**, choose Email/ Email-JSON. **Note** You can also choose a different endpoint based on your requirements.	Cloud administrator

Set up logging in Network Firewall

Task	Description	Skills required
Enable firewall logging.	1. Sign in to the AWS Management Console and open the [Amazon VPC console](). 2. In the navigation pane, under **NETWORK FIREWALL**, choose **Firewalls**. 3. In the **Firewalls** section, choose the firewall where you want to capture the server name from the SNI for outbound traffic. 4. Choose the **Firewall details** tab, and then choose **Edit** in the **Logging** section. 5. For **Log type**, select **Alert**. For **Log destinati on for alerts**, select **CloudWatch log group**. 6. For **CloudWatch log group**, search for and choose the log group that you created earlier, and then choose **Save**. For more information about using CloudWatch Logs as a log destination for Network Firewall, see [Amazon]()	Cloud administrator

Task	Description	Skills required
	CloudWatch Logs in the Network Firewall documentation.	

Set up a stateful rule in Network Firewall

Task	Description	Skills required
Create a stateful rule.	1. Sign in to the AWS Management Console and open the Amazon VPC console. 2. In the navigation pane, under **NETWORK FIREWALL**, choose **Network Firewall Rule Groups**. 3. Choose **Create Network Firewall rule group**. 4. On the **Create Network Firewall rule group** page, for the **Rule group type**, choose **Stateful rule group**. **Note**: For more information, see Working with stateful rule groups in AWS Network Firewall. 5. In the **Stateful rule group** section, enter a name and description for the rule group. 6. For **Capacity**, set the maximum capacity that	Cloud administrator

Task	Description	Skills required
	you want to allow for the stateful rule group (up to the maximum of 30,000). **Note**: You can't change this setting after you create the rule group. For information about how to calculate capacity, see [Setting rule group capacity in AWS Network Firewall](). For information about the maximum setting, see [AWS Network Firewall quotas](). 7. For **Stateful rule group options**, select **5-tuple**. 8. In the **Stateful rule order** section, choose **Default**. 9. In the **Rule variables** section, keep the default values. 10. In the **Add rule** section, choose **TLS** for **Protocol**. For **Source**, choose **Any**. For **Source port**, choose **Any port**. For **Destination**, choose **Any**. For **Destination port**, choose **Any port**. For **Traffic direction**, choose **Forward**. For **Action**, choose **Alert**. Choose **Add rule**.	

Task	Description	Skills required
	11. Choose **Create stateful rule group**.	
Associate the stateful rule to Network Firewall.	1. Sign in to the AWS Management Console and open the [Amazon VPC console](https://console.aws.amazon.com/vpc/). 2. In the navigation pane, under **NETWORK FIREWALL**, choose **Firewalls**. 3. Choose the firewall where you want to capture the server name from the SNI for outbound traffic. 4. In the **Stateful rule groups** section, choose **Actions**, and then choose **Add unmanaged stateful rule groups**. 5. On the **Add unmanaged stateful rule groups** page, select the stateful rule group that you created earlier, and then choose **Add stateful rule group**.	Cloud administrator

Create a Lambda function to read the logs

Task	Description	Skills required
Create the code for the Lambda function.	In an integrated developme nt environment (IDE) that can read the CloudWatch Logs	App developer

Task	Description	Skills required
	event from Network Firewall for outbound traffic, paste in the following Python 3 code and replace `<SNS-topic-ARN>` with your value:	

```
import json
import gzip
import base64
import boto3
sns_client = boto3.cli
ent('sns')
def lambda_handler(eve
nt, context):
    decoded_event =
 json.loads(gzip.de
compress(base64.b6
4decode(event['aws
logs']['data'])))
    body = '''
    {filtermatch}
    '''.format(
        loggroup=
decoded_event['log
Group'],
        logstream
=decoded_event['lo
gStream'],
        filtermat
ch=decoded_event['
logEvents'][0]['me
ssage'],
    )
    print(body)
    filterMatch =
 json.loads(body)
    data = []
    if 'http' in
 filterMatch['event']:
        data.appe
nd(filterMatch['ev
```

Task	Description	Skills required

	``
	ent']['http']['hos
	tname'])
	    elif 'tls' in
	 filterMatch['event']:
	        data.appe
	nd(filterMatch['ev
	ent']['tls']['sni'])
	    result = 'Domain
	 accessed ' + 1*' ' +
	 (data[0]) + 1*' ' 'via
	 AWS Network Firewall
	 ' + 1*' ' + (filterMa
	tch['firewall_name'])
	    print(result)
	    message = {'ServerN
	ame': result}
	    send_to_sns =
	 sns_client.publish(
	        TargetArn=<SNS-
	topic-ARN>,
	 #Replace with the SNS
	 topic ARN
	        Message=j
	son.dumps({'default':
	 json.dumps(message),
	
	    'sms': json.dump
	s(message),
	
	   'email': json.dump
	s(message)}),
	        Subject='Server
	 Name passed through the
	 Network Firewall',
	        MessageSt
	ructure='json'
	    )
	``
	
	This code sample parses the CloudWatch Logs content and captures the server name	

Task	Description	Skills required
	provided by the SNI in the HTTPS header.	
Create the Lambda function.	To create the Lambda function, follow the instructions in the Lambda documentation and choose **Python 3.9** for **Runtime**.	Cloud administrator
Add the code to the Lambda function.	To add your Python code to the Lambda function that you created earlier, follow the instructions in the Lambda documentation.	Cloud administrator

Task	Description	Skills required
Add CloudWatch Logs as a trigger to the Lambda function.	1. Sign in to the AWS Management Console and open the Lambda console. 2. In the navigation pane, choose **Functions**, and then choose the function that you created earlier. 3. In the **Function overview** section, choose **Add trigger**. 4. On the **Add trigger** page, in the **Trigger configura tion** section, choose **CloudWatch Logs**, and then choose **Add**. 5. For **Log group**, choose the CloudWatch log group that you created earlier. 6. For **Filter name**, enter a name for your filter. 7. Choose **Add**. 8. On the **Configuration** tab of your function's page, in the **Triggers** section, select the trigger that you just added, and then choose **Enable**. For more information, see Using Lambda with CloudWatch Logs in the Lambda documentation.	Cloud administrator

Task	Description	Skills required
Add SNS publish permissions.	Add the **sns:Publish** permission to the Lambda execution role, so that Lambda can make API calls to publish messages to SNS. 1. Find the execution role of the Lambda function that you created earlier. 2. Add the following policy to your AWS Identity and Access Management (IAM) role: `{ "Version": "2012-10-17", "Statement": [ { "Sid": "AllowSNSPublish", "Effect": "Allow", "Action": [ "sns:GetTopicAttributes", "sns:Subscribe", "sns:Unsubscribe", "sns:Publish" ], "Resource": "*" } ]`	Cloud administrator

Task	Description	Skills required
	`}`	

Test the functionality of your SNS notification

Task	Description	Skills required

Send traffic through Network Firewall.	1. Send or wait for HTTPS traffic to pass through Network Firewall.
	2. Check the SNS notification email that you receive from AWS when traffic passes through Network Firewall. The email includes the SNI details for outbound traffic. For example, the email generated from the Lambda code above will have the following content if the accessed domain name is **https://aws.amazon.com** and the subscription protocol is **EMAIL-JSON**:
	
	`{`
	`    "Type": "Notifica`
	`tion",`
	`    "MessageId":`
	` "<messageID>",`
	`    "TopicArn":`
	` "arn:aws:sns:us-we`
	`st-2:123456789:tes`
	`tSNSTopic",`
		Test engineer

Task	Description	Skills required
	"Subject": "Server Name passed through the Network Firewall", "Message": "{\"ServerName\": \"Domain 'aws.amazon.com' accessed via AWS Network Firewall 'AWS-Network-Firewall-Multi-AZ-firewall\"}", "Timestamp": "2022-03-22T04:10:04.217Z", "SignatureVersion": "1", "Signature": "<Signature>", "SigningCertURL": "<SigningCertUrl>", "UnsubscribeURL": "<UnsubscribeURL>" } Then, check the Network Firewall alert log in Amazon CloudWatch by following the instructions in the [Amazon CloudWatch documentation](). The alert log shows the following output: { "firewall_name": "AWS-Network-Firewall-Multi-AZ-firewall", "availability_zone": "us-east-2b",	

Task	Description	Skills required
	"event_timestamp": "<event timestamp>", "event": { "timestamp": "2021-03-22T04:10:04.214222+0000", "flow_id": <flow ID>, "event_type": "alert", "src_ip": "10.1.3.76", "src_port": 22761, "dest_ip": "99.86.59.73", "dest_port": 443, "proto": "TCP", "alert": { "action": "allowed", "signature_id": 2, "rev": 0, "signature": "", "category": "", "severity": 3 }, "tls": { "subject": "CN=aws.amazon.com", "issuerdn": "C=US, O=Amazon, OU=Server CA 1B, CN=Amazon", "serial": "<serial number>",	

Task	Description	Skills required
	`"fingerprint": "<fingerprint ID>", "sni": "aws.amazon.com", "version": "TLS 1.2", "notbefore": "2020-09-30T00:00:00", "notafter": "2021-09-23T12:00:00", "ja3": {}, "ja3s": {} }, "app_proto": "tls" } }`	

Use Terraform to automatically enable Amazon GuardDuty for an organization

Aarthi Kannan, Amazon Web Services

Summary

Amazon GuardDuty continuously monitors your Amazon Web Services (AWS) accounts and uses threat intelligence to identify unexpected and potentially malicious activity within your AWS environment. Manually enabling GuardDuty for multiple accounts or organizations, across multiple AWS Regions, or through the AWS Management Console can be cumbersome. You can automate the process by using an infrastructure as code (IaC) tool, such as Terraform, which can provision and manage multi-account, multi-Region services and resources in the cloud.

AWS recommends using AWS Organizations to set up and manage multiple accounts in GuardDuty. This pattern adheres to that recommendation. One benefit of this approach is that, when new accounts are created or added to the organization, GuardDuty will be auto-enabled in these accounts for all supported Regions, without the need for manual intervention.

This pattern demonstrates how to use HashiCorp Terraform to enable Amazon GuardDuty for three or more Amazon Web Services (AWS) accounts in an organization. The sample code provided with this pattern does the following:

- Enables GuardDuty for all AWS accounts that are current members of the target organization in AWS Organizations
- Turns on the *Auto-Enable* feature in GuardDuty, which automatically enables GuardDuty for any accounts that are added to the target organization in the future
- Allows you select the Regions where you want to enable GuardDuty
- Uses the organization's security account as the GuardDuty delegated administrator
- Creates an Amazon Simple Storage Service (Amazon S3) bucket in the logging account and configures GuardDuty to publish the aggregated findings from all accounts in this bucket
- Assigns a life-cycle policy that transitions findings from the S3 bucket to Amazon S3 Glacier Flexible Retrieval storage after 365 days, by default

You can manually run this sample code, or you can integrate it into your continuous integration and continuous delivery (CI/CD) pipeline.

Target audience

This pattern is recommended for users who have experience with Terraform, Python, GuardDuty, and AWS Organizations.

Prerequisites and limitations

Prerequisites

- An active AWS account.
- An organization is set up in AWS Organizations, and it contains at least the following three accounts:
  - **A management account –** This is the account from which you deploy the Terraform code, either standalone or as part of the CI/CD pipeline. The Terraform state is also stored in this account.
  - **A security account –** This account is used as the GuardDuty delegated administrator. For more information, see Important considerations for GuardDuty delegated administrators (GuardDuty documentation).
  - **A logging account –** This account contains the S3 bucket where GuardDuty publishes the aggregated findings from all member accounts.

  For more information about how to set up the organization with the required configuration, see Create an account structure (AWS Well-Architected Labs).
- An Amazon S3 bucket and an Amazon DynamoDB table that serve as a remote backend to store Terraform's state in the management account. For more information on using remote backends for the Terraform state, see S3 Backends (Terraform documentation). For a code sample that sets up remote state management with an S3 backend, see remote-state-s3-backend (Terraform Registry). Note the following requirements:
  - The S3 bucket and DynamoDB table must be in the same Region.
  - When creating the DynamoDB table, the partition key must be `LockID` (case-sensitive), and the partition key type must be **String**. All other table settings must be at their default values. For more information, see About primary keys and Create a table (DynamoDB documentation).
- An S3 bucket that will be used to store access logs for the S3 bucket in which GuardDuty will publish findings. For more information, see Enabling Amazon S3 server access logging (Amazon S3 documentation). If you're deploying to an AWS Control Tower landing zone, you can reuse the S3 bucket in the **log archive** account for this purpose.

- Terraform version 0.14.6 or later is installed and configured. For more information, see Get Started – AWS (Terraform documentation).
- Python version 3.9.6 or later is installed and configured. For more information, see Source releases (Python website).
- AWS SDK for Python (Boto3) is installed. For more information, see Installation (Boto3 documentation).
- jq is installed and configured. For more information, see Download jq (jq documentation).

Limitations

- This pattern supports macOS and Amazon Linux 2 operating systems. This pattern has not been tested for use in Windows operating systems.

> **Note**
>
> Amazon Linux 2 is nearing end of support. For more information, see the Amazon Linux 2 FAQs.

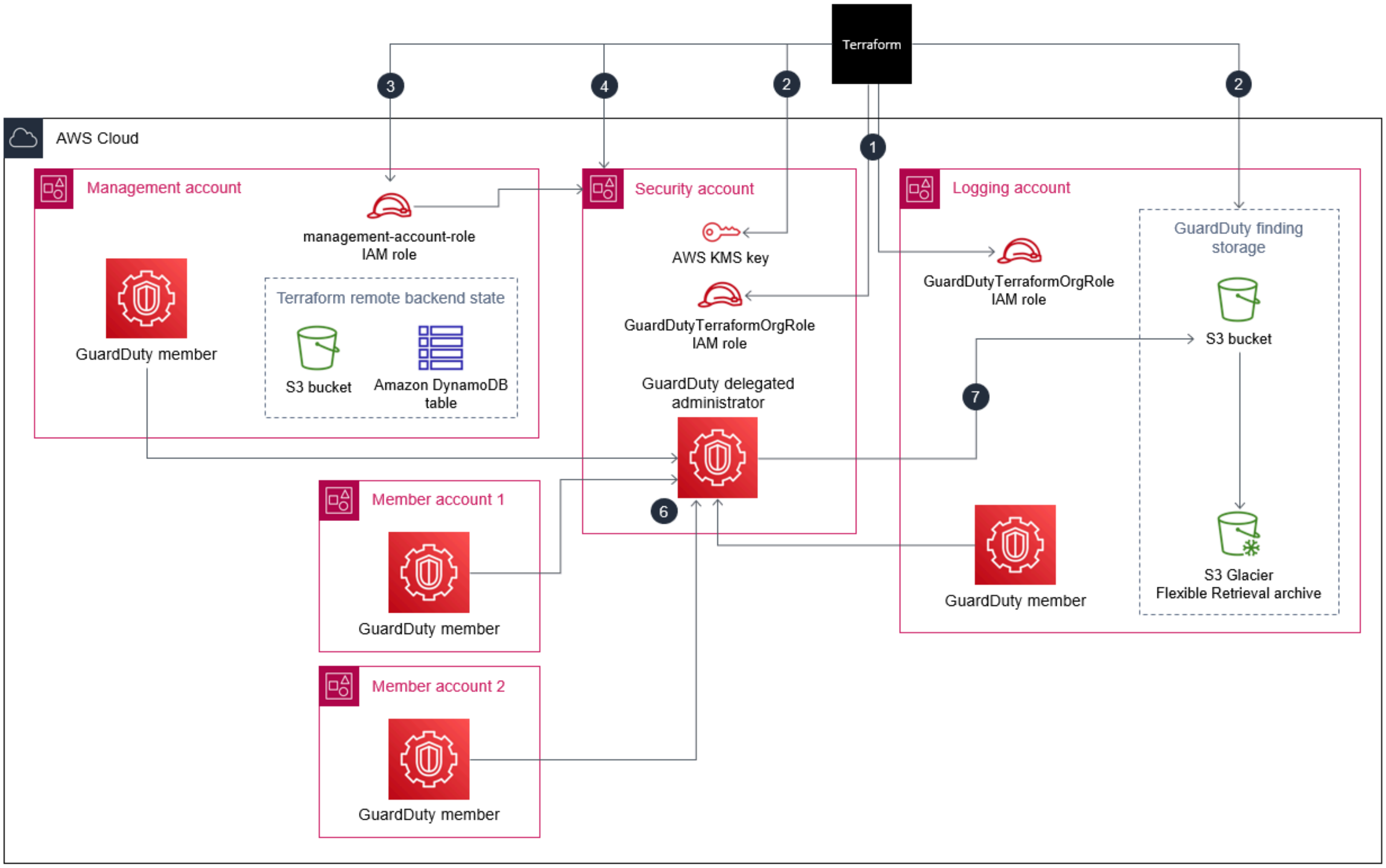
- GuardDuty must not already be enabled in any of the accounts, in any of the target Regions.
- The IaC solution in this pattern does not deploy the prerequisites.
- This pattern is designed for an AWS landing zone that adheres to the following best practices:
  - The landing zone was created by using AWS Control Tower.
  - Separate AWS accounts are used for security and logging.

Product versions

- Terraform version 0.14.6 or later. The sample code has been tested for version 1.2.8.
- Python version 3.9.6 or later.

Architecture

This section gives a high-level overview of this solution and the architecture established by the sample code. The following diagram shows the resources deployed across the various accounts in the organization, within a single AWS Region.



1. Terraform creates the **GuardDutyTerraformOrgRole** AWS Identity and Access Management (IAM) role in the security account and the logging account.
2. Terraform creates an S3 bucket in the default AWS Region in the logging account. This bucket is used as the publishing destination to aggregate all GuardDuty findings across all Regions and from all accounts in the organization. Terraform also creates an AWS Key Management Service (AWS KMS) key in the security account that is used to encrypt the findings in the S3 bucket and configures automatic archiving of findings from the S3 bucket into S3 Glacier Flexible Retrieval storage.
3. From the management account, Terraform designates the security account as the delegated administrator for GuardDuty. This means that the security account now manages the GuardDuty service for all member accounts, including the management account. Individual member accounts cannot suspend or disable GuardDuty by themselves.
4. Terraform creates the GuardDuty detector in the security account, for the GuardDuty delegated administrator.
5. If it is not already enabled, Terraform enables S3 protection in GuardDuty. For more information, see [Amazon S3 protection in Amazon GuardDuty](#) (GuardDuty documentation).

6. Terraform enrolls all current, active member accounts in the organization as GuardDuty members.
7. Terraform configures the GuardDuty delegated administrator to publish the aggregated findings from all member accounts to the S3 bucket in the logging account.
8. Terraform repeats steps 3 through 7 for each AWS Region you choose.

Automation and scale

The sample code provided is modularized so that you can integrate it into your CI/CD pipeline for automated deployment.

Tools

AWS services

- Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance.
- Amazon GuardDuty is a continuous security monitoring service that analyzes and processes logs to identify unexpected and potentially unauthorized activity in your AWS environment.
- AWS Identity and Access Management (IAM) helps you securely manage access to your AWS resources by controlling who is authenticated and authorized to use them.
- AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to protect your data.
- AWS Organizations is an account management service that helps you consolidate multiple AWS accounts into an organization that you create and centrally manage.
- Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.
- AWS SDK for Python (Boto3) is a software development kit that helps you integrate your Python application, library, or script with AWS services.

Other tools and services

- HashiCorp Terraform is a command-line interface application that helps you use code to provision and manage cloud infrastructure and resources.
- Python is a general-purpose programming language.
- jq is a command-line processor that helps you work with JSON files.

Code repository

The code for this pattern is available on GitHub, in the amazon-guardduty-for-aws-organizations-with-terraform repository.

Epics

Enable GuardDuty in the organization

Task	Description	Skills required
Clone the repository.	In a Bash shell, run the following command. In *Clone the repository* in the Additional information section, you can copy the full command containing the URL of the GitHub repository. This clones the amazon-guardduty-for-aws-organizations-with-terraform repository from GitHub. `git clone <github-repository-url>`	DevOps engineer
Edit the Terraform configuration file.	1. In the `root` folder of the cloned repository, replicate the **configuration.json.sample** file by running the following command. `cp configuration.json.sample configuration.json` 2. Edit the new **configuration.json** file, and define	DevOps engineer, General AWS, Terraform, Python

Task	Description	Skills required
	the values for each of the following variables: • `management_acc_id` – Account ID of the management account. • `delegated_admin_acc_id` – Account ID of the security account. • `logging_acc_id` – Account ID of the logging account. • `target_regions` – Comma-separated list of AWS Regions where you want to enable GuardDuty. • `organization_id` – AWS Organizations ID of the organization in which you are enabling GuardDuty. • `default_region` – The Region where your Terraform state is stored in the management account. This is the same Region where you deployed the S3 bucket and DynamoDB table for the Terraform backend. • `role_to_assume_for_role_creation` – Name that you want	

Task	Description	Skills required
	to assign to a new IAM role in the security and logging accounts. You create this new role in the next story. Terraform assumes this role to create the `GuardDuty TerraformOrgRole` IAM role in the security and logging accounts. • `finding_publishing_frequency` – Frequency at which GuardDuty publishes findings to the S3 bucket. • `guardduty_findings_bucket_region` – Preferred Region where you want to create the S3 bucket for published findings. • `logging_acc_s3_bucket_name` – Preferred name for the S3 bucket for published findings. • `security_acc_kms_key_alias` – AWS KMS alias for the key used to encrypt GuardDuty findings. • `s3_access_log_bucket_name` – Name of a preexisting S3 bucket	

Task	Description	Skills required
	where you want to collect access logs for the S3 bucket used for GuardDuty findings. This bucket should be in the same AWS Region as the GuardDuty findings bucket. • `tfm_state_backend_s3_bucket` – Name of the preexisting S3 bucket to store the Terraform remote backend state. • `tfm_state_backend_dynamodb_table` – Name of the preexisting DynamoDB table for locking the Terraform state. 3. Save and close the configuration file.	

Task	Description	Skills required
Generate CloudFormation templates for new IAM roles.	This pattern includes an IaC solution to create two CloudFormation templates. These templates create two IAM roles that Terraform uses during the setup process. These templates adhere to the security best practice of [least-privilege permissions](). 1. In a Bash shell, in the repository `root` folder, navigate to `cfn-templates/` . This folder contains CloudFormation templates files with stubs. 2. Run the following command. This replaces the stubs with the values you provided in the **configuration.json** file. `bash scripts/replace_config_stubs.sh` 3. Confirm that the following CloudFormation templates were created in the `cfn-templates/` folder: • **management-account-role.yaml** – This file contains the role definition and the associated permissions	DevOps engineer, General AWS

Task	Description	Skills required
	for the IAM role in the management account, which has the minimum permissions required to complete this pattern. • **role-to-assume-for-role-creation.yaml** – This file contains the role definition and the associated permissions for the IAM role in the security and logging accounts. Terraform assumes this role in order to create the **GuardDutyTerraformOrgRole** role in these accounts.	

Task	Description	Skills required
Create the IAM roles.	Following the instructions in [Creating a stack](#) (CloudFormation documentation), do the following: 1. Deploy the **role-to-assume-for-role-creation.yaml** stack in both the security and logging accounts. 2. Deploy the **management-account-role.yaml** stack in the management account. When you successfully create the stack and see the `CREATE_COMPLETE` stack status, in the output, make note of the Amazon Resource Name (ARN) of this new role.	DevOps engineer, General AWS
Assume the IAM role in the management account.	As a security best practice, we recommend that you assume the new **management-account-role** IAM role before proceeding. In the AWS Command Line Interface (AWS CLI), enter the command in *Assume the management account IAM role* in the [Additional Information](#) section.	DevOps engineer, General AWS

Task	Description	Skills required
Run the setup script.	In the repository `root` folder, run the following command to start the setup script. `bash scripts/full-setup.sh` The **full-setup.sh** script performs the following actions: • Exports all configuration values as environment variables • Generates the **backend.tf** and **terraform.tfvars** code files for each Terraform module • Enables trusted access for GuardDuty in the organization through the AWS CLI. • Imports the organization state into the Terraform state • Creates the S3 bucket for publishing findings in the logging account • Creates the AWS KMS key for encrypting findings in the security account • Enables GuardDuty across the organization, in all selected Regions, as	DevOps engineer, Python

Task	Description	Skills required
	described in the Architecture section	

(Optional) Disable GuardDuty in the organization

Task	Description	Skills required
Run the clean-up script.	If you used this pattern to enable GuardDuty for the organization and want to disable GuardDuty, in the repository `root` folder, run the following command to start the **cleanup-gd.sh** script. `bash scripts/cleanup-gd.sh` This script disables GuardDuty in the target organization, removes any deployed resources, and restores the organization to its previous state before using Terraform to enable GuardDuty. **Note** This script does not remove the Terraform state files or lock files from the local and remote backends. If you need	DevOps engineer, General AWS, Terraform, Python

Task	Description	Skills required
	to do so, you must perform these actions manually. Also, this script does not delete the imported organization or the accounts managed by it. Trusted access for GuardDuty isn't disabled as part of the clean-up script.	
Remove IAM roles.	Delete the stacks that were created with the **role-to-assume-for-role-creation.yaml** and **management-account-role.yaml** CloudFormation templates. For more information, see [Deleting a stack](#) (CloudFormation documentation).	DevOps engineer, General AWS

Related resources

AWS documentation

- Managing multiple accounts (GuardDuty documentation)
- Granting least privilege (IAM documentation)

AWS marketing

- Amazon GuardDuty
- AWS Organizations

Other resources

- Terraform
- Terraform CLI Documentation

Additional information

Clone the repository

Run the following command to clone the GitHub repository.

```
git clone https://github.com/aws-samples/amazon-guardduty-for-aws-organizations-with-terraform
```

Assume the management account IAM role

To assume the IAM role in the management account, run the following command. Replace `<IAM role ARN>` with the ARN of the IAM role.

```
export ROLE_CREDENTIALS=$(aws sts assume-role --role-arn <IAM role ARN> --role-session-name AWSCLI-Session --output json)
export AWS_ACCESS_KEY_ID=$(echo $ROLE_CREDENTIALS | jq .Credentials.AccessKeyId | sed 's/"//g')
export AWS_SECRET_ACCESS_KEY=$(echo $ROLE_CREDENTIALS | jq .Credentials.SecretAccessKey | sed 's/"//g')
export AWS_SESSION_TOKEN=$(echo $ROLE_CREDENTIALS | jq .Credentials.SessionToken | sed 's/"//g')
```

Verify operational best practices for PCI DSS 4.0 by using AWS Config

Tala Qraitem and Alex Goff, Amazon Web Services

Summary

The Payment Card Industry Data Security Standard (PCI DSS) outlines essential technical and operational protocols to help safeguard payment data. PCI DSS was developed to encourage and enhance data security for payment card accounts. It also facilitates the global adoption of consistent security measures. Although it's specifically designed for environments with payment card account data, you can use PCI DSS to help protect against threats and secure other elements in the payment ecosystem.

PCI DSS version 4.0 was released to address evolving requirements, provide clarification or additional guidance, and improve the structure and format of the standard. For more information about the changes, see Summary of changes from PCI DSS version 3.2.1 to 4.0.

An AWS Config conformance pack is a collection of AWS Config rules and remediation actions that help you create security, operational, or cost-optimization governance checks. You can deploy a conformance pack as a single entity in an AWS account and AWS Region, or you can deploy across an organization in AWS Organizations.

The conformance packs for PCI DSS version 4.0 augment and build upon the conformance pack for version 3.2.1. The rules in the conformance pack map to the rules in the standard. For more information, see the mapping provided in the *Attachments* section. You can choose between two versions of this conformance pack: one that includes global resource types and one that excludes them.

Important

Conformance packs are not designed to fully ensure compliance with a specific governance or compliance standard. You are responsible for making your own assessment of whether usage meets applicable legal and regulatory requirements.

Prerequisites and limitations

Prerequisites

- Have an active AWS account.
- Set up AWS Config.
- Meet the prerequisites for conformance packs.
- Deploy the PCI DSS version 3.2.1 conformance pack.
- Have permissions to access AWS Config and manage conformance packs. For an example policy, see the Additional information section of this pattern.

Limitations

- Your AWS account has default quotas, formerly referred to as *limits*, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, but not all quotas can be increased. Make sure that you are familiar with the AWS Config service limits, including the limits for single account conformance packs and organization conformance packs.
- The version of this conformance pack that includes global resource types is intended for deployment only in the `us-east-1` Region.
- The version of this conformance pack that excludes global resources types is intended for deployment only in the following Regions:
  - `ap-east-1`
  - `ap-south-1`
  - `ap-northeast-2`
  - `ap-southeast-1`
  - `ap-southeast-2`
  - `ap-northeast-1`
  - `ca-central-1`
  - `eu-central-1`
  - `eu-west-1`
  - `eu-west-2`
  - `eu-west-3`
  - `eu-north-1`
  - `sa-east-1`
  - `us-east-2`

- `us-west-1`
- `us-west-2`

Tools

AWS services

- AWS Config provides a detailed view of the resources in your AWS account and how they're configured. It helps you identify how resources are related to one another and how their configurations have changed over time.
- AWS Systems Manager helps you manage your applications and infrastructure running in the AWS Cloud. It simplifies application and resource management, shortens the time to detect and resolve operational problems, and helps you manage your AWS resources securely at scale.

Code repository

The conformance packs are located in the AWS Config conformance packs GitHub repository. This repository contains the following templates related to PCI DSS version 4.0:

- Operational-Best-Practices-for-PCI-DSS-v4.0-including-global-resourcetypes.yaml
- Operational-Best-Practices-for-PCI-DSS-v4.0-excluding-global-resourcetypes.yaml

Epics

Deploy and manage the conformance pack

Task	Description	Skills required
Download the conformance pack.	If you're deploying the conformance pack in the `us-east-1` Region, download the Operational-Best-Practices-for-PCI-DSS-v4.0-including-global-resourcetypes.yaml template.	DevOps engineer

Task	Description	Skills required
	If you're deploying the conformance pack in a different Region, download the Operational-Best-Practices-for-PCI-DSS-v4.0-excluding-global-resourcetypes.yaml template.	
(Optional) Modify the conformance pack.	You can modify the conformance pack template for the unique needs of your organization. For example, you can create custom remediation actions. For more information about how to create and modify templates, see Creating templates for custom conformance packs in the AWS Config documentation.	General AWS

Task	Description	Skills required
Deploy the conformance pack.	If you're deploying in a target AWS account or AWS Region, follow the instructions in [Deploying conformance packs](#) in the AWS Config documenta tion. You can use the AWS Management Console or the AWS Command Line Interface (AWS CLI). If you're deploying the conformance pack across an organization in AWS Organizations, follow the instructions in [Deploy AWS Config conformance pack using Quick Setup](#) in the AWS Systems Manager documenta tion.	General AWS
(Optional) Edit the conforman ce pack.	If you want to edit the conformance pack, follow the instructions in [Editing conformance packs](#) in the AWS Config documenta tion. You can use the AWS Management Console or the AWS CLI.	General AWS

Task	Description	Skills required
(Optional) Delete the conformance pack.	If you want to delete the conformance pack, follow the instructions in Deleting conformance packs in the AWS Config documenta tion. You can use the AWS Management Console or the AWS CLI.	General AWS

Related resources

AWS resources

- Conformance packs for AWS Config (AWS Config documentation)
- Deploy AWS Config conformance pack using Quick Setup (Systems Manager documentation)
- PCI DSS compliance on AWS (AWS website)
- PCI DSS version 4.0 on AWS (Compliance guide)

PCI DSS resources

- PCI DSS version 4.0 Resource Hub
- PCI Security Standards Council Document Library
- Summary of changes from PCI DSS version 3.2.1 to 4.0

Additional information

The following is a sample AWS Identity and Access Management (IAM) policy that allows the user to access AWS Config and manage conformance packs:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "config:PutConfigRule",
                "config:PutConformancePack",
                "config:DeleteConfigRule",
                "config:DeleteRemediationConfiguration",
                "config:DeleteConformancePack",
                "config:PutRemediationConfigurations",
                "config:BatchGetAggregateResourceConfig",
                "config:BatchGetResourceConfig",
                "config:Get*",
                "config:Describe*",
                "config:Deliver*",
                "config:List*",
                "config:Select*"
            ],
            "Resource": "*"
        }
    ]
}
```

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Verify that new Amazon Redshift clusters have required SSL endpoints

Priyanka Chaudhary, Amazon Web Services

Summary

This pattern provides an Amazon Web Services (AWS) CloudFormation template that automatically notifies you when a new Amazon Redshift cluster is launched without Secure Sockets Layer (SSL) endpoints.

Amazon Redshift is a fully managed, petabyte-scale, cloud-based data warehouse service. It is designed for large-scale dataset storage and analysis. It is also used to perform large-scale database migrations. For security, Amazon Redshift supports SSL to encrypt the connection between the user's SQL Server client application and the Amazon Redshift cluster. To configure your cluster to require an SSL connection, you set the `require_SSL` parameter to `true` in the parameter group that is associated with the cluster during launch.

The security control provided with this pattern monitors Amazon Redshift API calls in AWS CloudTrail logs and initiates an Amazon CloudWatch Events event for the CreateCluster, ModifyCluster, RestoreFromClusterSnapshot, CreateClusterParameterGroup, and ModifyClusterParameterGroup APIs. When the event detects one of these APIs, it calls AWS Lambda, which runs a Python script. The Python function analyzes the CloudWatch event for the listed CloudTrail events. When an Amazon Redshift cluster is created, modified, or restored from an existing snapshot, a new parameter group is created for the cluster, or an existed parameter group is modified, the function checks the `require_SSL` parameter for the cluster. If the parameter value is `false`, the function sends an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information: the Amazon Redshift cluster name, AWS Region, AWS account, and Amazon Resource Name (ARN) for Lambda that this notification is sourced from.

Prerequisites and limitations

Prerequisites

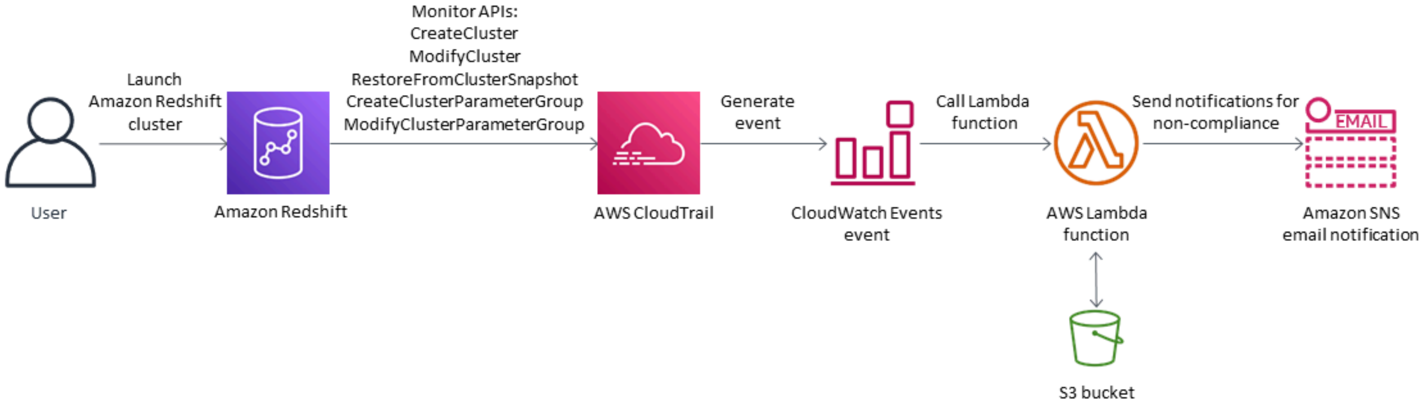
- An active AWS account.
- A virtual private cloud (VPC) with a cluster subnet group, and an associated security group.

Limitations

- This security control is regional. You must deploy it in each AWS Region you want to monitor.

Architecture

Target architecture



Automation and scale

- If you are using [AWS Organizations](), you can use [AWS Cloudformation StackSets]() to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- [AWS CloudFormation]() – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually.
- [Amazon CloudWatch Events]() – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- [AWS Lambda]() – AWS Lambda is a compute service that supports running code without provisioning or managing servers.

- Amazon Redshift – Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud.
- Amazon S3 – Amazon Simple Storage Service (Amazon S3) is an object storage service. You can use Amazon S3 to store and retrieve any amount of data at any time, from anywhere on the web.
- Amazon SNS – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

Code

This pattern includes the following attachments:

- `RedshiftSSLEndpointsRequired.zip` – The Lambda code for the security control.
- `RedshiftSSLEndpointsRequired.yml` – The CloudFormation template that sets up the event and Lambda function.

Epics

Set up the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	On the Amazon S3 console, choose or create an S3 bucket to host the Lambda code .zip file. This S3 bucket must be in the same AWS Region as the Amazon Redshift cluster you want to monitor. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.	Cloud architect

Task	Description	Skills required
Upload the Lambda code.	Upload the Lambda code .zip file provided in the *Attachments* section to the S3 bucket.	Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Launch the AWS CloudFormation template.	Open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/) in the same AWS Region as your S3 bucket and deploy the attached template `RedshiftSSLEndpointsRequired.yml` . For more information about deploying AWS CloudFormation templates, see [Creating a stack on the AWS CloudFormation console](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html) in the CloudFormation documentation.	Cloud architect
Complete the parameters in the template.	When you launch the template, you'll be prompted for the following information: • **S3 bucket**: Specify the bucket that you created or selected in the first epic. This is where you uploaded the attached Lambda code (.zip file).	Cloud architect

Task	Description	Skills required
	• **S3 key**: Specify the location of the Lambda .zip file in your S3 bucket (for example, *filename*.zip or controls/*filename*.zip). Do not include leading slashes. • **Notification email**: Provide an active email address where you want to receive Amazon SNS notifications. • **Lamba logging level**: Specify the logging level and frequency for the Lambda function. Use **Info** to log detailed informati onal messages on progress, **Error** for error events that would still allow the deployment to continue, and **Warning** for potential ly harmful situations.	

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the CloudFormation template deploys successfu lly, it sends a subscription email to the email address you provided. You must confirm this email subscript	Cloud architect

Task	Description	Skills required
	ion to start receiving violation notifications.	

Related resources

- Creating an S3 bucket (Amazon S3 documentation)
- Uploading files to an S3 bucket (Amazon S3 documentation)
- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail (AWS CloudTrail documentation)
- Creating an Amazon Redshift cluster (Amazon Redshift documentation)
- Configuring security options for connections (Amazon Redshift documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

Verify that new Amazon Redshift clusters launch in a VPC

Priyanka Chaudhary, Amazon Web Services

Summary

This pattern provides an Amazon Web Services (AWS) CloudFormation template that automatically notifies you when an Amazon Redshift cluster is launched outside a virtual private cloud (VPC).

Amazon Redshift is a fully managed, petabyte-scale, cloud-based data warehouse product. It is designed for large-scale dataset storage and analysis. It is also used to perform large-scale database migrations. Amazon Virtual Private Cloud (Amazon VPC) lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources such as Amazon Redshift clusters in a virtual network that you define.

The security control provided with this pattern monitors Amazon Redshift API calls in AWS CloudTrail logs, and initiates an Amazon CloudWatch Events event for the CreateCluster and RestoreFromClusterSnapshot APIs. When the event detects one of these APIs, it calls AWS Lambda, which runs a Python script. The Python function analyzes the CloudWatch event. If an Amazon Redshift cluster is created or restored from a snapshot and appears outside the Amazon VPC network, the function sends an Amazon Simple Notification Service (Amazon SNS) notification to the user with the relevant information: the Amazon Redshift cluster name, AWS Region, AWS account, and Amazon Resource Name (ARN) for Lambda that this notification is sourced from.

Prerequisites and limitations

Prerequisites

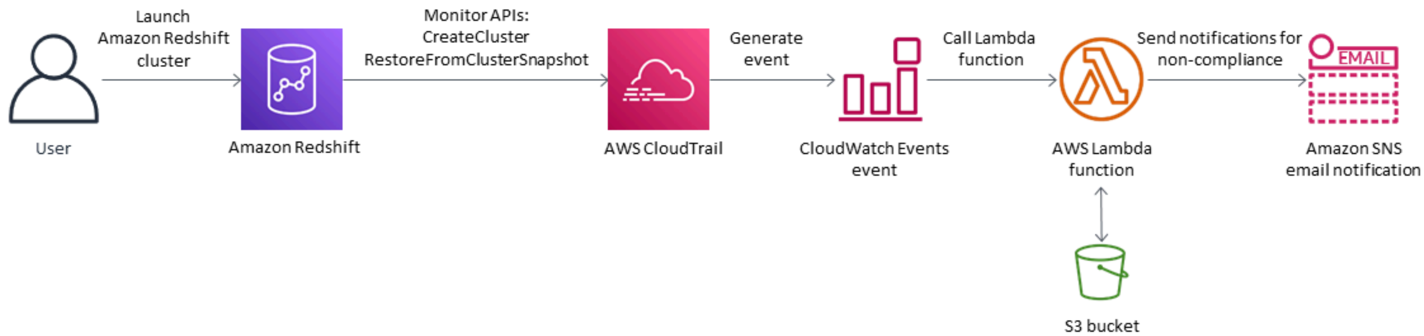
- An active AWS account.
- A VPC with a cluster subnet group, and an associated security group.

Limitations

- The AWS CloudFormation template supports the CreateCluster and RestoreFromClusterSnapshot actions (new clusters) only. It does not detect existing Amazon Redshift clusters that were created outside a VPC.
- This security control is regional. You must deploy it in each AWS Region you want to monitor.

Architecture

Target architecture



Automation and scale

If you are using [AWS Organizations](), you can use [AWS Cloudformation StackSets]() to deploy this template in multiple accounts that you want to monitor.

Tools

AWS services

- [AWS CloudFormation]() – AWS CloudFormation helps you model and set up your AWS resources, provision them quickly and consistently, and manage them throughout their lifecycle. You can use a template to describe your resources and their dependencies, and launch and configure them together as a stack, instead of managing resources individually.
- [AWS CloudTrail]() – AWS CloudTrail helps you implement governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.
- [Amazon CloudWatch Events]() – Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in AWS resources.
- [AWS Lambda]() – AWS Lambda is a compute service that supports running code without provisioning or managing servers. AWS Lambda runs your code only when needed and scales automatically, from a few requests per day to thousands per second.
- [Amazon Redshift]() – Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. Amazon Redshift is integrated with your data lake, which enables you to use your data to acquire new insights for your business and customers.

- [Amazon S3](#) – Amazon Simple Storage Service (Amazon S3)is a highly scalable object storage service that you can use for a wide range of storage solutions, including websites, mobile applications, backups, and data lakes.
- [Amazon SNS](#) – Amazon Simple Notification Service (Amazon SNS) coordinates and manages the delivery or sending of messages between publishers and clients, including web servers and email addresses.

Code

This pattern includes the following attachments:

- `RedshiftMustBeInVPC.zip` – The Lambda code for the security control.
- `RedshiftMustBeInVPC.yml` – The CloudFormation template that sets up the event and Lambda function.

To use these files, follow the instructions in the next section.

Epics

Set up the S3 bucket

Task	Description	Skills required
Define the S3 bucket.	On the Amazon S3 console, choose or create an S3 bucket to host the Lambda code .zip file. This S3 bucket must be in the same AWS Region as the Amazon Redshift cluster that you want to monitor. An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. The S3 bucket name cannot include leading slashes.	Cloud architect

Task	Description	Skills required
Upload the Lambda code.	Upload the Lambda code (`RedshiftMustBeInVPC.zip` file) provided in the *Attachments* section to the S3 bucket.	Cloud architect

Deploy the CloudFormation template

Task	Description	Skills required
Launch the CloudFormation template.	Open the [AWS CloudFormation console](https://console.aws.amazon.com/cloudformation/) in the same AWS Region as your S3 bucket and deploy the attached template (`RedshiftMustBeInVPC.yml`). For more information about deploying AWS CloudFormation templates, see [Creating a stack on the AWS CloudFormation console](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-console-create-stack.html) in the CloudFormation documentation.	Cloud architect
Complete the parameters in the template.	When you launch the template, you'll be prompted for the following information: • **S3 bucket**: Specify the bucket that you created or selected in the first epic. This is where you uploaded the attached Lambda code (.zip file).	Cloud architect

Task	Description	Skills required
	• **S3 key**: Specify the location of the Lambda .zip file in your S3 bucket (for example, *filename*.zip or controls/*filename*.zip). Do not include leading slashes. • **Notification email**: Provide an active email address where you want to receive Amazon SNS notifications. • **Lamba logging level**: Specify the logging level and frequency for the Lambda function. Use **Info** to log detailed informati onal messages on progress, **Error** for error events that would still allow the deployment to continue, and **Warning** for potential ly harmful situations.	

Confirm the subscription

Task	Description	Skills required
Confirm the subscription.	When the CloudFormation template deploys successfu lly, it sends a subscription email to the email address you provided. You must confirm this email subscript	Cloud architect

Task	Description	Skills required
	ion to start receiving violation notifications.	

Related resources

- Creating an S3 bucket (Amazon S3 documentation)
- Uploading files to an S3 bucket (Amazon S3 documentation)
- Creating a stack on the AWS CloudFormation console (AWS CloudFormation documentation)
- Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail (AWS CloudTrail documentation)
- Creating an Amazon Redshift cluster (Amazon Redshift documentation)

Attachments

To access additional content that is associated with this document, unzip the following file: attachment.zip

More patterns

- Access a bastion host by using Session Manager and Amazon EC2 Instance Connect
- Access container applications privately on Amazon ECS by using AWS Fargate, AWS PrivateLink, and a Network Load Balancer
- Access container applications privately on Amazon ECS by using AWS PrivateLink and a Network Load Balancer
- Access container applications privately on Amazon EKS using AWS PrivateLink and a Network Load Balancer
- Allow EC2 instances write access to S3 buckets in AMS accounts
- Associate an AWS CodeCommit repository in one AWS account with Amazon SageMaker AI Studio Classic in another account
- Authenticate existing React application users by using Amazon Cognito and AWS Amplify UI
- Automate adding or updating Windows registry entries using AWS Systems Manager
- Automate encryption enforcement in AWS Glue using an AWS CloudFormation template
- Automatically attach an AWS managed policy for Systems Manager to EC2 instance profiles using Cloud Custodian and AWS CDK
- Automatically encrypt existing and new Amazon EBS volumes
- Block public access to Amazon RDS by using Cloud Custodian
- Centralize DNS resolution by using AWS Managed Microsoft AD and on-premises Microsoft Active Directory
- Implement centralized custom Checkov scanning to enforce policy before deploying AWS infrastructure
- Check AWS CDK applications or CloudFormation templates for best practices by using cdk-nag rule packs
- Check EC2 instances for mandatory tags at launch
- Configure cross-account access to Amazon DynamoDB
- Configure HTTPS encryption for Oracle JD Edwards EnterpriseOne on Oracle WebLogic by using an Application Load Balancer
- Configure logging and monitoring for security events in an AWS IoT environment
- Configure mutual TLS authentication for applications running on Amazon EKS
- Connect by using an SSH tunnel in pgAdmin

- Create a React app by using AWS Amplify and add authentication with Amazon Cognito
- Create a report of Network Access Analyzer findings for inbound internet access in multiple AWS accounts
- Customize Amazon CloudWatch alerts for AWS Network Firewall
- Deploy a ChatOps solution to manage SAST scan results by using Amazon Q Developer in chat applications custom actions and AWS CloudFormation
- Document your AWS landing zone design
- Enable encrypted connections for PostgreSQL DB instances in Amazon RDS
- Encrypt an existing Amazon RDS for PostgreSQL DB instance
- Enforce automatic tagging of Amazon RDS databases at launch
- Enforce tagging of Amazon EMR clusters at launch
- Ensure Amazon EMR logging to Amazon S3 is enabled at launch
- Generate an AWS CloudFormation template containing AWS Config managed rules using Troposphere
- Get Amazon SNS notifications when the key state of an AWS KMS key changes
- Help enforce DynamoDB tagging
- Identify and alert when Amazon Data Firehose resources are not encrypted with an AWS KMS key
- Implement Microsoft Entra ID-based authentication in an AWS Blu Age modernized mainframe application
- Implement SAML 2.0 authentication for Amazon WorkSpaces by using Auth0 and AWS Managed Microsoft AD
- Implement SHA1 hashing for PII data when migrating from SQL Server to PostgreSQL
- Improve operational performance by enabling Amazon DevOps Guru across multiple AWS Regions, accounts, and OUs with the AWS CDK
- Ingest and migrate EC2 Windows instances into an AWS Managed Services account
- Migrate Amazon RDS for Oracle to Amazon RDS for PostgreSQL in SSL mode by using AWS DMS
- Migrate an ELK Stack to Elastic Cloud on AWS
- Migrate an F5 BIG-IP workload to F5 BIG-IP VE on the AWS Cloud
- Monitor Amazon Aurora for instances without encryption
- Provision least-privilege IAM roles by deploying a role vending machine solution
- Rotate database credentials without restarting containers

- Secure and streamline user access in a Db2 federation database on AWS by using trusted contexts
- Send AWS WAF logs to Splunk by using AWS Firewall Manager and Amazon Data Firehose
- Serve static content in an Amazon S3 bucket through a VPC by using Amazon CloudFront
- Set up end-to-end encryption for applications on Amazon EKS using cert-manager and Let's Encrypt
- Use user IDs in IAM policies for access control and automation
- Verify that ELB load balancers require TLS termination
- View AWS Network Firewall logs and metrics by using Splunk
- Visualize IAM credential reports for all AWS accounts using Amazon QuickSight